2004

# ILMDA: Intelligent Learning Materials Delivery Agents

Leen-Kiat Soh
*University of Nebraska*, lsoh2@unl.edu

L.D. Miller
*University of Nebraska*, lmille@cse.unl.edu

Todd Blank
*University of Nebraska*, tblank@cse.unl.edu

Suzette Person
*University of Nebraska - Lincoln*, sperson@cse.unl.edu

University of Nebraska-Lincoln, Computer Science and Engineering
Technical Report # TR-UNL-CSE-2004-0005

# ILMDA: Intelligent Learning Materials Delivery Agents

**Leen-Kiat Soh, L.D. Miller, Todd Blank, and Suzette Person**

University of Nebraska
Computer Science and Engineering, 115 Ferguson Hall, Lincoln, NE 68588-0115 USA

{ lksoh, lmille, tblank, sperson } @cse.unl.edu

## Abstract

In this paper, we describe an intelligent agent that delivers learning materials adaptively to different students, factoring in the usage history of the learning materials, the student static background profile, and the student dynamic activity profile. Our assumption is that through the interaction of a student going through a learning material (i.e., a topical tutorial, a set of examples, and a set of problems), our agent will be able to capture and utilize the student's activity as the primer to select the appropriate example or problem to administer to the student. Even if the agent fails to do so, it is able to recover to provide a more appropriate example of problem based on what it learns from its failure. In addition, our agent monitors the usage history of the learning materials and derives empirical observations that improve its performance. We have built an end-to-end ILMDA infrastructure, with a GUI front-end, an agent powered by case-based reasoning (CBR), and a mySQL multi-database backend. We have also built a comprehensive simulator for our experiments. Preliminary experiments show the feasibility, correctness, and learning capability of ILMDA.

## 1. Introduction

Traditionally, learning materials are delivered in a textual format and on paper. For example, a learning module on a topic may include a description (or a tutorial) of the topic, a few examples illustrating the topic, and one or more exercise problems to gauge how well the students have achieved the expected understanding of the topic. The delivery mechanism of these learning materials has traditionally been via textbooks and/or instructions provided by a teacher. A teacher, for example, may provide a few pages of notes about a topic, explain the topic for a few minutes, discuss a couple of examples, and then give some exercise problems as homework. During the delivery, students ask questions and the teacher attempt to answer the questions accordingly. Thus, the delivery is interactive: the teacher learns how well the students have mastered the topic, and the students clarify their understanding of the topic. In a traditional classroom of a relatively small size, the above scenario is feasible. However, when E-learning approaches such as distance learning and asynchronous learning are involved, or in the case of a large class size, the traditional delivery mechanism is often not feasible.

In this paper, we propose an intelligent agent that delivers learning materials based on the usage history of the learning materials, the student static background profile (such as GPA, majors, interests, and courses taken), and the student dynamic activity profile (based on their inter-

actions with the agent). The agent uses the profiles to decide, through case-based reasoning (CBR) (Kolodner 1993), which learning modules (examples and problems) to present to the students. Our CBR treats the input situation as a problem, and the solution is basically the specification of an appropriate example or problem. Our agent also uses the usage history of each learning material to adjust the appropriateness of the examples and problems in a particular situation. We have built an end-to-end ILMDA infrastructure, with an interactive GUI front-end that is active, an agent powered by CBR and capable of learning, and a mySQL multi-database backend.

In the following, we first discuss some related work in the area of intelligent tutoring systems. Then, we present our Intelligent Learning Materials Deliver Agent (ILMDA) project, its goals and framework. Subsequently, we propose the CBR methodology and design. We then describe on its implementation and a comprehensive simulation. Next we report on our preliminary experiments. Finally, we conclude.

## 2. Related Work

Research strongly supports the user of technology as a catalyst for improving the learning environment (Sivin-Kachala and Bialo 1998). Educational technology has been shown to stimulate more interactive teaching, effective grouping of students, and cooperative learning. A few studies, which estimated the cost effectiveness, reported time saving of about 30%. At first, professors can be expected to struggle with the change brought about by technology. However, they will adopt, adapt, and eventually learn to use technology effortlessly and creatively (Kadiyala and Crynes 1998).

Kulik and Kulik (1991), Bangert-Drowns et al. (1985), and Baxter (1990) defined several types of computer-aided education systems. In Computer-Assisted Instruction, the system provides drill and practice exercises and tutorial instruction. In Computer-Managed Instruction, the system evaluates and stores student performance and guides students to appropriate instructional resources. In Computer-Enriched Instruction, the system satisfies student requests such as solving a mathematical equation, generating data, and executing programs.

As summarized in Graesser et al. (2001), intelligent tutoring systems (ITSs) are clearly one of the successful enterprises in AI. There is a long list of ITSs that have been tested on humans and have proven to facilitate learning. These ITSs use a variety of computational modules that are familiar to those of us in AI: production systems, Bayesian networks, schema templates, theorem proving, and explanatory reasoning. Graesser et al. (2001) also pointed out the weaknesses of the current state of tutoring systems: First, it is possible for students to guess and find an answer and such shallow learning will not be detected by the system. Second, ITSs do not involve students in conversations so students might not learn the domain's language. Third, to understand the students' thinking, the GUI of the ITSs discourages students to tend to focus on the details instead of the overall picture of a solution.

There have been successful ITSs such as as PACT (Koedinger et al. 1997), ANDES (Gertner and VanLehn 2000), AutoTutor (Graesser et al. 2001), and SAM (Cassell et al. 2000), but without machine learning capabilities. These systems do not generally adapt to new circumstance, do not self-evaluate and self-configure their own strategies, and do not monitor the usage history of the learning materials being delivered or presented to the students. In our research, we aim to build intelligent tutoring agents that are able to learn how to deliver appropriate different

learning materials to different types of students and to monitor and evaluate how the learning materials are received by the students.

## 3. Project Framework

In the ILMDA project we aim to design an intelligent agent to deliver learning materials. Each learning material consists of three components: (1) a tutorial, (2) a set of related examples, and (3) a set of exercise problems to assess the student's understanding of the topic. Based on how a student progresses through the learning material and based on his or her profile, an ILMDA will choose the appropriate examples and exercise problems for the student. In this manner, the ILMDA will customize the learning material. Most software tutors or learning delivery mechanisms are able to customize the learning material for different students, with or without agent-based technology. Our design has a modular design of the course content and delivery mechanism, utilizes true agent intelligence where an agent is able to learn how to deliver its learning materials better, and self-evaluates its own learning materials.

The underlying assumptions behind the design of our agent are the following. First, a student's behavior in viewing an online tutorial, and how he or she interacts with the tutorial, the examples, and the exercises, is a good indicator of how well the student is understanding the topic in question, and this behavior is observable and quantifiable. Second, different students exhibit different behaviors for different topics such that it is possible to show a student's understanding of a topic, say, T1, with an example E1, and at the same time to show the same student's lack of understanding of the same topic T1 with another E2, and this differentiation is known and can be implemented.

Further, we want to develop an integrated, flexible, easy-to-use database of courseware and ILMDA system, including operational items such as student profiles, ILMDA success rates, etc., and educational items such as learner model, domain expertise, and course content. This will allow teachers and educators to monitor and track student progress, the quality of the learning materials, and the appropriateness of the materials for different student groups. Finally, we plan to build an agent capable of adapting its delivery of examples and exercise problems to students' real-time behavior and historical profile, learning useful delivery strategies, and performing self-monitoring and evaluation tasks. With the ability to self-monitor and evaluate, the agent can identify how best to deliver a learning module to a particular student type with distinctive behaviors. We see this as valuable knowledge to instructional designers and educational researchers as ILMDA not only is a testbed for testing hypotheses, but it is also an active decision maker that can expose knowledge or patterns that are previously unknown to researchers.

## 4. Methodology

Our ILMDA system is based on a three-tier methodology, as shown in Figure 1. It consists of a graphical user interface (GUI) front-end application, a database backend, and the ILMDA reasoning in between. A student user accesses the learning material through the GUI. The agent captures the student's interactions Acknowledgments with the GUI and provides the ILMDA reasoning module with a parametric profile of the student and environment. The ILMDA reasoning module performs case-based reasoning to obtain a search query (a vector of search keys) to retrieve and adapt the most appropriate example or problem from the database. The agent then delivers the example or problem in real-time back to the user through the interface. In our agent,

we use case-based reasoning (CBR) to select and adapt which examples and problems the student is given.

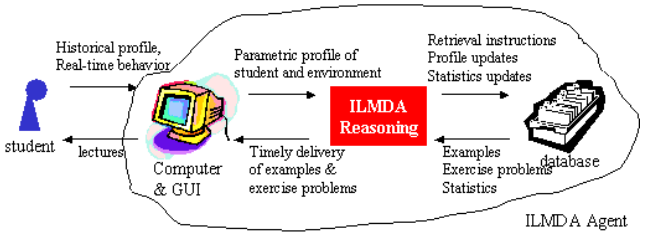


**Figure 1.** Overall methodology of the ILMDA system.

### 4.1. Overall Flow of Operations

When a student starts the ILMDA application, he or she is first asked to login. This associates the user with his or her profile information. The information is stored in two separate tables. All of their generally static information, such as name, major, interests, etc., is stored in one table, while the user's dynamic information (i.e., how much time, on average, they spend in each section; how many times they click the mouse in each section, etc.) is stored in another table. After a student is logged in, he or she selects a topic and then views the tutorial on that topic. Following the tutorial the agent looks at the student's static profile, as well as the dynamic actions the student took in the tutorial, and searches the database for a similar case. The agent then adapts the output of that similar case depending on how the cases differ, and uses the adapted output to search for a suitable example to give to the student. After the student is done looking at the examples, the same process is used to select an appropriate problem. Again, the agent takes into account how the student behaved during the example, as well as his or her background profile. After the student completes an example or problem, they may elect to be given another. If they do so, the agent notes that the last example or problem it gave the student was not a good choice for that student, and tries a different solution. Figure 2 shows the interaction steps between our ILMDA agent interacts and a student.

### 4.2. Learner Model

A learner model is one that tells us the metacognitive level of a student by looking at the student's behavior as he or she interacts with the learning materials. We achieve this by profiling a learner/student along two dimensions: student background and student activity. The background of a student stays relatively static and consists of the student's last name, first name, major, GPA, goals, affiliations, aptitudes, and competencies. The dynamic student profile captures the student's real-time behavior and patterns. It consists of the student's online interactions with the GUI module of the ILMDA agent including the number of attempts on the same learning material, number of different modules taken so far, average number of mouse clicks during the tutorial, average number of mouse clicks viewing the examples, average length of time spent during the tutorial, number of quits after tutorial, number of successes, and so on.

In our research, we incorporate the learner model into the case-based reasoning (CBR) module as part of the problem description of a case: Given the parametric behavior, the CBR module will

pick the best matching case and retrieve the set of solution parameters which will determine which examples to pick or which exercise problems to pick.

We are currently in consultation with educational researchers to incorporate more complex learner models such as self-efficacy and motivations.

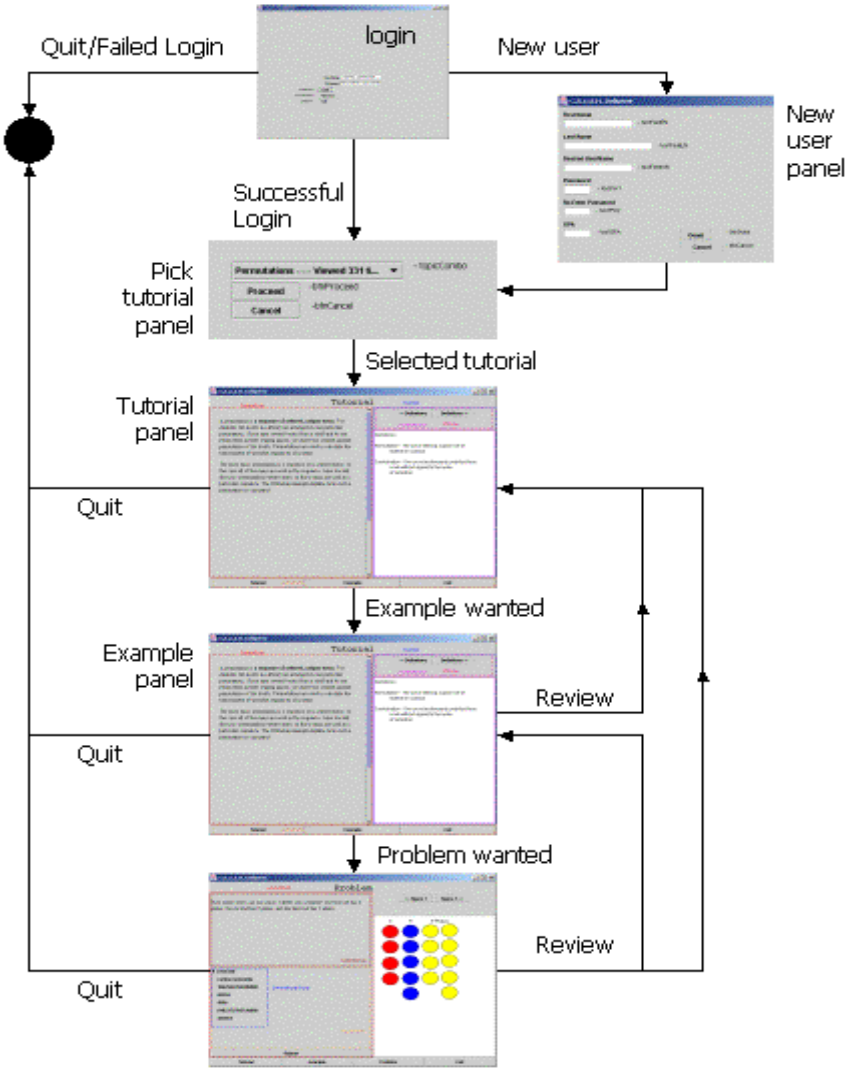


**Figure 2.** GUI and interactions between ILMDA and students.

### 4.3. Case-Based Reasoning

Each agent has a case-based reasoning (CBR) module. In our framework, the agent consults the CBR module to obtain the specifications of the appropriate types of examples or problems to administer to the users. The learner model discussed above and the profile of the learning materials constitute the problem description of a case. The solution is simply a set of search keys (Table 1) guiding the agent in its retrieval of either a problem or an exercise. Here, we briefly describe how the agent uses a rulebase of adaptation heuristics, simulated annealing, and graph traversal to support the adaptation and retrieval mechanisms of case-based reasoning.

| Solution Parameters | Description |
|---|---|
| *TimesViewed* | The number of times the material has been viewed |
| *DiffLevel* | The difficulty level of the case between 0 and 10 |
| *MinUseTime* | The shortest time, in milliseconds, a single student has viewed the material |
| *MaxUseTime* | The longest time, in milliseconds, a single student has viewed the material |
| *AveUseTime* | The average time, in milliseconds, the material has been viewed by students |
| *Bloom* | Bloom's Taxonomy number |
| *AveClick* | The average number of clicks the interface has recorded for this material |
| *Length* | The number of characters in the course content for this material |
| *Content* | The stored list of interests for this material |

**Table 1.** The solution parameters of a case in the ILMDA system. These parameters form the search query to retrieve an example or a problem from the database.

**Adaptation Heuristics.** The objective of the adaptation process is to adapt the solution parameters for the old case (Table1) based on the difference between the problem description of the new and old cases. Each adaptation heuristic is weighted and responsible for adapting one solution parameter. Our agent also adjusts the weights of the heuristics using a learning module. Finally, we implement the heuristics in a rulebase to add flexibility and modularity to our agent design—for example, different student age groups may have different adaptation rulebases.

**Simulated Annealing**. The situation may arise where the adaptation process is given an old case with undesirable solution parameters—a set that has consistently led to unsuccessful outcomes. To avoid repeating the same mistake, we use simulated annealing to obtain different and possibly satisfactory solutions. Because this simulated annealing process is costly compared to the normal adaptation process, an agent does this only when it encounters a case that is retrieved often and that has a low success rate.

**Graph Traversal**. We represent symbolic attributes such as *interests* and *majors* in content graphs to support the adaptation process. With this, our agent can adapt symbolic attributes by searching a weighted graph. Our agent has the ability to adjust the edge values of the content graph described above. For instance, at first, one may assume that an interest in cars and an interest in theater are unrelated (with a large weight for the edge between the two interests). However, later it is discovered that many students who are interested in cars perform in a similar way to students who are interested in theater. The graph-learning module would pick up on this and decrease the weight (distance) between the two interests.

**Case-Based Learning**

To improve its reasoning process, our agent learns the differences between good cases (cases with a good solution for its problem space) and bad cases (cases with a bad solution for its problem space). It also meta-learns adaptation heuristics, the significance of input features of the cases, and the weights of a content graph for symbolic feature values.

Our agent uses various weights when selecting a similar case, a similar example, a similar problem, or when it compares interests via the interest graph. By adjusting these weights we can improve our results, and hence, learn from our experiences. In order to improve our agent's independence, we want to have the agent adjust the weights without human intervention. To do

this, the agent uses simple methods to adjust the weights called learning modules. Adjusting the weights in this manner gives us a layered learning system (Stone 2000) because the changes that one module makes propagate through to the other modules. For instance, the changes we make to the similarity heuristics will alter which cases the other modules perceive as similar.

In our agent design, we have four CBL modules. First, there is a case learning module that learns about good and bad cases, through both traditional methods and by using simulated annealing. Second, we have a graph-weight learning module. To facilitate database retrieval, we relax some symbolic features (such as interest) so that the agent is able to always return an example or problem to the student. We use a content graph for each such feature, e.g., Interest. The weights connecting the vertices (different interests) in the graph denote the distance between two interests. Lastly, we have an adaptation learning module that learns about the importance or quality of the adaptation heuristics. The case-learning module uses traditional CBL and a small amount of reinforcement learning. The other three modules meta-learn how to better retrieve from the database, evaluate the similarity between two cases, and adapt the solution of the best case to the current problem. This layered learning allows our agent to learn how to better serve the students.

**Graphical User Interface**

The ILMDA front-end GUI application is written in Java, using the Sun Java Swing library. A student progresses through the tutorials, examples, and problems in the following manner. The student logs onto the system. If he or she is a new student, then a new profile is created. The student then selects a topic to study. The agent administers the tutorial associated with the topic to the student accordingly. The student studies the tutorial, occasionally clicking and scrolling, and browsing the embedded hyperlinks. Then when the student is ready to view an example, he or she clicks to proceed. Sensing this click, the agent immediately captures all the mouse activity and time spent in the tutorial and updates the student's activity profile. The student goes through a similar process and may choose to quit the system, indicating a failure of our example, or going back to the tutorial page for further clarification. If the student decides that he or she is ready for the problem, the agent will retrieve the most appropriate one based on the student's updated dynamic profile.

## 5. Implementation & Simulation

We have implemented an end-to-end prototype ILMDA system with a front-end, applet-based Graphical User Interface (GUI) that captures all user mouse activities, a CBR-powered agent, and a backend database system containing a set of learning materials. We have also implemented a Simulator to test the agent by running experiments on it with virtual students. The ILMDA system is written in Java (SDK version 1.4.2). Both integrated development environments were run under Windows 2000/XP operating systems. The backend database is a MySQL database (Version 3.23). The system connects to the database using the JDBC drivers for Java. We have also built a comprehensive simulator to provide us with *virtual students* to use ILMDA. This simulator will allow us to test the correctness and feasibility of the ILDMA, as well as to evaluate our learner and instructional models in the future. When we run our experiments, this simulator bypasses the GUI component and directly feeds the agent with simulated student background and activity profiles.

### 5.1. Simulator

Our simulator consists of two distinct modules: a *Student Generator* and an *Outcome Generator*. The first module generates nine different types of students based on their aptitudes and speeds (Table 2) in using online learning materials. This also guides the simulation of actual student activity. The second module simulates interactions and outcomes.

| **Aptitude/Speed** | **Fast** | **Medium** | **Slow** |
|---|---|---|---|
| **High** | 1 | 2 | 3 |
| **Average** | 4 | 5 | 6 |
| **Low** | 7 | 8 | 9 |

**Table 2.** Table of student types used by ILMDA Simulator. E.g., type 2 student has high aptitude and medium speed.

**Student Generator.** The Student Generator module creates the *virtual* students that ILMDA interacts with. This module generates (1) all student background values such as names, GPAs, and interests, and (2) the activity profile such as the average time spent on a session and average number of mouse clicks. The underlying approach we use is to choose a random value from a Gaussian distribution qualified by the student type. For aptitude, it has three Gaussian distributions. A random value to indicate where the student's aptitude is, given a particular aptitude type (high, average, or low). Then, given that aptitude value, it's probability is determined based on the Gaussian distribution for the type. Similarly, another random value is generated and processed for speed. These two values are then averaged using different weights to generate a single unit-less value that we call the *independent parameter value* (IPV). To obtain a value for a particular problem descriptor such as the average number of mouse clicks (*aveClicks*), the IPV is mapped proportionally to the bounds of *aveClicks*. The resultant mapping yields the simulated value for *aveClicks*.

**Outcome Generator**. The Outcome Generator simulates the *interaction* and *quit* behavior for each virtual student. The interaction behavior basically is simulated using the results of the Student Generator and modifying them with the content (tutorials, examples, and problems). To illustrate, the simulated time spent on an example (*ExampleTime*), the simulated number of clicks on an example (*ExampleClicks)*, and the simulated number of times that the student goes from the example back to the tutorial (*ExmpToTutorial)* are computed as follows:

$$ExampleTime = ((XL/AXL)*(XD/AXD))/2*SAET*RDM$$

$$ExampleClicks = (((XL/AXL)*(XD/AXD))/2*SAEC*RDM$$

$$ExmpToTutorial = (((XL/AXL)*(XD/AXD))/2*SAETT*RDM$$

where

$XL$ = the example's length

$AXL$ = the average example length for this topic

$XD$ = the example's difficulty level

*AXD* = the average example difficulty for this topic

*SAET* = the student's average example time

*SAEC* = the student's average number of example clicks

*SAETT* = the student's average example to tutorial

*RND* = a random number between .8 and 1.2

These formulae operate under the following assumptions: (1) the longer an example is, the more time, more clicks, and more example to tutorials there will be; (2) the more difficult an example is, the more time, more clicks, and more example to tutorials there will be; (3) the student's averages accurately reflect how they will act on future examples; and (4) RND gives us a small variation to the values.

We use the above formulae to generate the actual interaction behavior. With this and the student background, the simulator now has a complete problem description for the case-based reasoning to take place.

A student may choose to quit while reading the tutorial, or going through the exercise, or working on the problem, depending on the student's aptitude and the difficulty level of the example and problem he or she encounters. To simulate this, we have the following assumptions.

We start with a base probability that a given student type will quit any tutorial, example, or problem. We assume that high-aptitude students are less likely to quit than other students. Table 3 shows the base probabilities. The quitting rate for tutorials is assumed to be 5% lower than the base quitting rates for an example or a problem, as we assume that students are less likely to quit a tutorial.

| High | Average | Low |
|---|---|---|
| 10% (5%) | 25% (20%) | 40% (35%) |

**Table 3.** Base probabilities for quitting for each aptitude type. Numbers in parentheses indicate the values for tutorial.

To decide whether a student would quit a tutorial, we simply randomly (using a uniform distribution) make a student quit according to the above base probabilities.

To decide whether a student would quit an example, we modify the base probabilities by adding to it two values: a difficulty-based modification and a time-based modification. Our rationale is that a student is more likely to quit an example when the example is more difficult than appropriate for the student's aptitude, and when the example takes longer to go through than appropriate for the student's speed.

To decide whether a student would quit a problem, we use a similar modification as the above.

To decide whether a student would answer a problem correctly, it is based on the product of the final, modified probability and a factor we call the *motivation factor* (MF). We see that a student who has made it through the tutorial, the example, and the problem is more likely to answer the problem correctly.

### 5.2. Preliminary Experiments

The objective of our first experiments with our simulation was to determine the feasibility of ILMDA, its correctness, and the learning performance. Using the simulator discussed above, we generated 900 students, with 100 students in each student type. Particularly, we focused our experiments on type 2 students, with high aptitude and medium speed. We then conducted the simulation in three steps. The first step ran the simulator for 1000 iterations with all learning modules of the ILMDA disabled. That is, no new cases were added and no weights were adjusted. The second step ran the simulator for 100 iterations with the learning modules enabled. For this experiment, we also coded the system to always learn a new case. The last step ran the simulator for another 1000 iterations with the learning modules turned off. Our aim was to compare the results of the first and the third steps.

After the first and third steps, we collect the following parameters: (1) the average quitting point for students, (2) the average score of students who reach a problem, (3) the number of times each example/problem given, (4) the number of times students quit each example/problem, and (5) the number of times students answered each example/problem correctly. The average quitting point is based on the following scale: the agent receives 0 point if the student quits the tutorial, 1 point if during an example, and 2 points if during a problem. The agent receives 3 and 4 points for incorrect and correct answers, respectively.

Overall, the end-to-end behavior of our ILMDA is correct, performing as expected. Our agent is able to deliver different learning materials to different students adaptively. Our agent is also able to learn new cases and adjusts its weight. Based on the experiments, we observe that:

- The average quitting points for the first and third steps were 1.801 and 1.634, respectively. This indicates that after the agent was trained after step 2, students were more prone to quitting before reaching a problem. Indeed, only 59 students reached a problem in step 1, and 44 did in step 3. This indicates that the ILMDA failed to further the students' activity more often than before it was trained. Upon a closer look, we realize that the ILMDA was trained to provide difficult-enough examples and problems to the students. In the first step, the ILMDA identified a particular example that had a 100% success rate and learned to use another, more difficult example. As a result, students quit more often and failed to reach the problem session. We are currently addressing this issue to include instructional scaffolding and hints for each example.
- The average scores for the student who answered a problem were 0.407 and 0.568 for the first and third steps, respectively. This indicates that the ILMDA was able to give more suitable problems to the students after the 100-iteration training received in the second step.
- After learning, the ILMDA gave twice as many examples to the students, comparing the results of step 1 and step 3. This shows that the agent was able to learn to retrieve different examples to give to the students.
- After learning, the ILMDA gave exactly the same number of problems to the students. Coupling this with the second observation above, we see that the ILMDA was able to learn to apply more appropriate problems to different students even though it did not increase the number of problems used.

- The ILMDA identified a particular example, used successfully 67 times in the first step. As a result, our agent stopped delivering this example to the students in the third step. This shows how our agent is capable of adapting to the profile of a learning material. However, as observed previously, this also led to a decrease in the average quitting points.

## 6. Conclusions

We have described an intelligent agent that delivers learning materials adaptively to different students, factoring in the usage history of the learning materials, the student static background profile, and the student dynamic activity profile. We have built an ILMDA infrastructure, with a GUI front-end, an agent powered by case-based reasoning (CBR), and a mySQL multi-database backend. We have also built a comprehensive simulator for our experiments. Preliminary experiments demonstrate the correctness of the end-to-end behavior of the ILMDA agent, and show the feasibility of ILMDA and its learning capability. We are currently continuing with the experiments on all nine types of students and popularizing our database of learning materials. Future work includes incorporating complex learner and instructional models into the agent and conducting further experiments on each learning mechanism, and investigating how ILMDA adapts to a student's behavior, and how ILMDA adapts to different types of learning materials.

## 7. Acknowledgments

The research project is supported by the Great Plains Software Technology Initiative and the CSE Department at the University of Nebraska, Lincoln, NE.

## 8. References

Bangert-Drowns, R., Kulik, J., and C.-L. Kulik, C.-L. 1985. Effectiveness of Computer-Based Education in Secondary Schools, *J. Computer-Based instruction*, 12(3):59-68.

Baxter, E. 1990. Comparing Conventional and Resource Based Education in Chemical Engineering: Student Perceptions of a Teaching Innovation, *Higher Education*, 19:323-340.

Bradshaw, G. 1987. Learning about Speech Sounds: The NEXUS Project, in *Proc. 11th Int. Workshop on Machine Learning*, Irvine, CA, 1-11

Cassell, J., Annany, M., Basur, N., Bickmore, T., Chong, P., Mellis, D., Ryokai, K., Smith, J., Vilhjálmsson, H., and Yan, H. 2000. Shared Reality: Physical Collaboration with a Virtual Peer, *ACM SIGCHI Con. on Human Factors in Comp. Sys.*, April 1-6, The Hague, The Netherlands

Gertner, A. S. and VanLehn, K. 2000. ANDES: A Coached Problem-Solving Environment for Physics, in *Proc. ITS'2000*, 133-142.

Graesser, A. C., VanLehn, K., Rosé, C. P., Jordan, P. W., and Harter, D. 2001. Intelligent Tutoring Systems with Conversational Dialogue, *AI Magazine*, 22(4):39-51.

Kadiyala, M. and b. L. Crynes (1998). Where's the Proof? A Review of Literature on Effectiveness of Information Technology in Education, in *Proc. 1998 FIE Conf.*, 33-37.

Koedinger, K. R., Anderson, J. R., Hadley, W. H., and Mark, M. A. 1997. Intelligent Tutoring Goes to School in the Big City, *J. Artificial Intelligence in Ed.* 8(1):30-43.

Kolodner, J. 1993. *Case-Based Reasoning.* Morgan Kaufmann.

Kulik, C.-L. and Kulik, J. 1991. Effectiveness of Computer-Based Instruction: an Updated Analysis, *Computers in Human Behavior* 7:75-94.

Sivin-Kachala, J. and Bialo, E. 1998. *Report on the Effectiveness of Technology in Schools, 1990-1997*, Software Publishers Association

Stone, P. 2000. Layered Learning in Multiagent Systems. MIT Press.