In *Proceedings of the Nineteenth Conference on Innovative Applications of AI*, July 24-26, 2007.

Supporting Feedback and Assessment of Digital Ink Answers to In-Class Exercises

Kimberle Koile, Kevin Chevalier, Michel Rbeiz, Adam Rogal,
David Singer, Jordan Sorensen, Amanda Smith, Kah Seng Tay, Kenneth Wu

MIT Computer Science and Artificial Intelligence Laboratory
32 Vassar Street, 32-221
Cambridge, MA 02139 USA
kkoile@csail.mit.edu
{kcheval, arogal, singerd, jsorense, kahseng, zstrif}@mit.edu
{mrbeiz, amandacs}@alum.mit.edu

Abstract

Effective teaching involves treating the presentation of new material and the assessment of students' mastery of this material as part of a seamless and continuous feedback cycle. We have developed a computer system, called Classroom Learning Partner (CLP), that supports this methodology, and we have used it in teaching an introductory computer science course at MIT over the past year. Through evaluation of controlled classroom experiments, we have demonstrated that this approach reaches students who would have otherwise been left behind, and that it leads to greater attentiveness in class, greater student satisfaction, and better interactions between the instructor and student. The current CLP system consists of a network of Tablet PCs, and software for posing questions to students, interpreting their handwritten answers, and aggregating those answers into equivalence classes, each of which represents a particular level of understanding or misconception of the material. The current system supports a useful set of recognizers for specific types of answers, and employs AI techniques in the knowledge representation and reasoning necessary to support interpretation and aggregation of digital ink answers.

Introduction

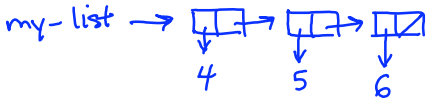
Effective and timely feedback is widely acknowledged to be a powerful means of improving learning (e.g., Angelo and Cross 1993, Bransford 1999, Steadman 1998). This technique is applied easily when working with individuals or small groups. Applying this technique in the classroom is another matter, however. How can classroom instructors determine which concepts have been understood in a class of 25 or more, what kinds of misunderstandings exist, and how prevalent they are? How can instructors make these determinations quickly and effectively enough to respond on the spot? A variety of approaches have been used over the years, ranging from straightforward queries ("Who doesn't understand?"), to structured questions ("Who thinks the answer is …?"), to more elaborate approaches of the sort enabled by wireless polling systems (e.g., Draper 2004).

All of these techniques offer some benefit, but come with drawbacks. No one likes to admit that they don't understand, the number of volunteered answers to in-class questions is notoriously low, and while registering an answer electronically with a "clicker" provides helpful anonymity, the interaction currently must be structured in terms of some variety of multiple choice.

We are designing, building, and evaluating a system called Classroom Learning Partner (CLP) to enable interaction between students and instructors that offers anonymity in replies and the ability to aggregate student answers, but that provides a much wider range of interaction styles than is currently available with "clickers"—styles more in keeping with how people naturally work together.

As one example of our vision, consider a question we often ask in our introductory computer science course: Draw the data structure that results from evaluating a particular Scheme[1] expression. To assess understanding for an expression, e.g., (define my-list (list 4 5 6)), we use a variety of techniques, such as asking for volunteers, writing the correct answer on the board and asking who got it, or using two-part "carbon" paper and collecting one of the parts. All of these techniques, however, are too elaborate, labor intensive, and slow for the small but important point being made.

What if each student instead could write an answer on his or her tablet computer, sketching something such as:



then submit the reply anonymously, and have software at the receiving system interpret each student's combination of drawing and handwriting as a data structure; classify the result as either correct, as one of a number of known misunderstandings, or as an answer type not previously encountered; and immediately present to the instructor a

[1] Scheme is a dialect of Lisp.

histogram indicating the classification results. This functionality would enable students to assess their understanding quickly and instructors to repair any misconceptions immediately.

We have constructed and tested in the classroom a system capable of interpreting and aggregating classroom answers written as numbers or character strings, or sets or sequences of numbers or character strings. We have employed AI techniques in the knowledge representation and reasoning necessary to support interpretation and aggregation of digital ink answers: an ontology centered around exercises and answers, a semantic representation for interpreted digital ink, a method for using answer type information to improve ink interpretation rates, and similarity measures for aggregating interpreted ink answers. We have evaluated the system in two ways: the performance of the system's ink interpretation and aggregation components, and the effect on student learning of real-time feedback and assessment of wirelessly submitted digital ink answers to in-class exercises. In the following sections of this paper, we discuss related work, and describe our system architecture, use of AI techniques, and our evaluations.

Related Work

Classroom Learning Partner uses a wireless distributed presentation system, Classroom Presenter (Anderson et al. 2004), as its underlying infrastructure. CLP's goals are similar to Classroom Presenter's: to increase instructor-student interaction and learning. CLP has two additional goals: to explicitly support real-time feedback and in-class assessment, and to do so in classes larger than is currently possible with Classroom Presenter.

Ubiquitous Presenter (Wilkerson, et al. 2005) is a web-based version of Classroom Presenter that extends student input devices to include a keyboard, and radio-button polling plus aggregation for fixed-answer questions. DyKnow (Berque 2004) supports wireless transmission of ink between instructor and student machines, though does not interpret or aggregate student answers. Other systems support student question-asking using small handheld devices in class: ActiveClass (Ratto, et al. 2003) allows students to send typed questions directly to an instructor during class; with eFuzion (Wentling et al. 2003) students can post typed questions and answers to a group website.

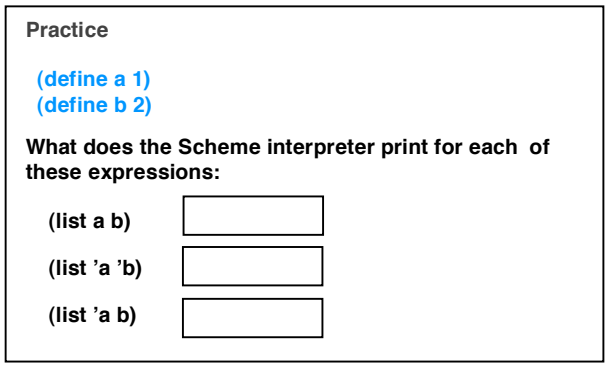
Systems that come closest to CLP's aggregation idea are wireless polling systems: Students use a transmitter, aka "clicker", to submit anonymous answers to multiple-choice, true and false, or matching questions. The results are tabulated and displayed on the instructor's computer in the form of a histogram. Such a polling system provides a way for students to communicate their misunderstandings to an instructor. Instructors, however, are limited to asking questions with predefined sets of answers—close-ended questions, which do not foster the critical thinking skills that open-ended questions do (Bloom 1956). CLP's aim is to facilitate aggregation of both close-ended and open-ended questions.

Classroom Learning Partner Architecture

CLP's system architecture is shown in Figure 1. The major components are an instructor authoring tool, an ink interpreter, an aggregator, and a results displayer. The components communicate via a central database, which embodies our ontology. Most of CLP is written in C#, the implementation language for the underlying Classroom Presenter infrastructure.

Instructor Authoring Tool

CLP is organized around the idea of exercises, so the instructor needs a way to create exercises for a classroom presentation. Using the Instructor Authoring Tool (IAT), an instructor creates a presentation in Microsoft PowerPoint, then adds exercise information to the presentation. The IAT allows an instructor to add two key pieces of information: the location of a student's ink answer, and the expected type for that answer. Providing the answer location allows students to take notes anywhere on a slide and not have to "lasso" an answer for submission. More importantly, the location information allows the instructor to include more than one exercise per slide: the ink interpreter knows where to find answers for particular exercises. Consider the following slide:



Each box represents an exercise answer, so that three exercises can be worked and discussed together, giving students more practice and saving the instructor time. The associated expected type information is used by the ink interpreter to increase interpretation rates. Consider the following ink answer: S. If the expected type is a number, the interpreter can prefer a hypothesis for 5 rather than one for S. The aggregator uses the expected type information to select appropriate similarity measures for its groupings. The exercise information is stored in the presentation, for ease of access by the interpreter, and in the central database, for ease of access by the aggregator.

In our current system, the instructor does not need to provide answers; the aggregator clusters student answers rather than matching student answers to instructor answers.

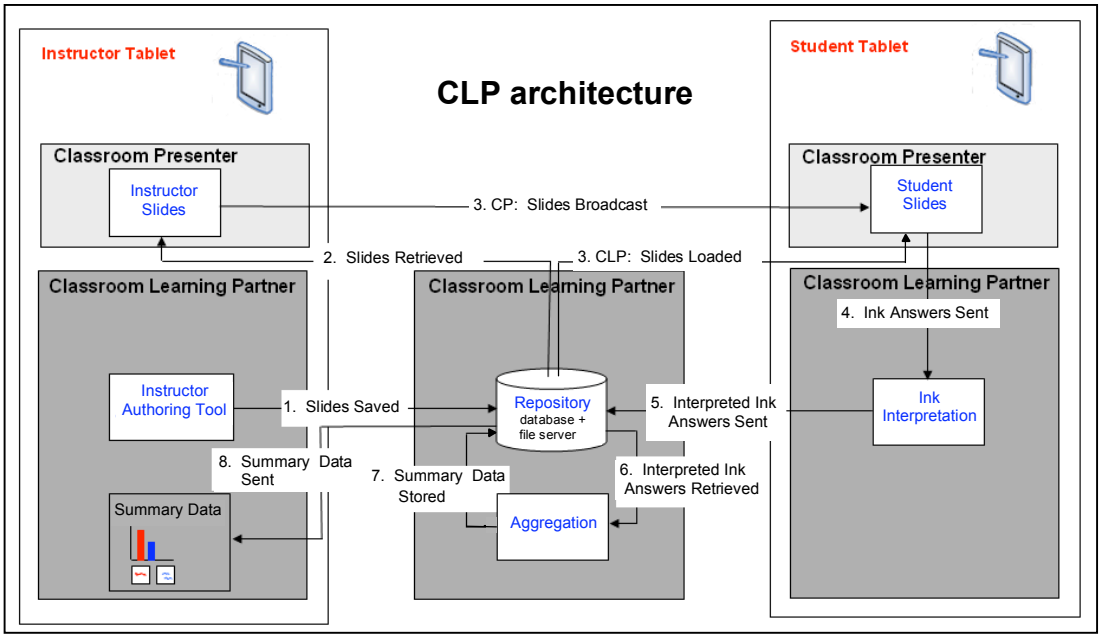


Figure 1. CLP architecture

Before class:

1. Instructor creates presentation and exercises using IAT; exercises are stored in database, slides on file server.

During class:

2. Instructor retrieves presentation from database (or it is resident on her machine already).
3. In Classroom Presenter, presentation slides are broadcast to student machines; in CLP slides are automatically loaded onto student machines when they log in to their Tablet PCs.
4. When a slide with an exercise is displayed, student enters ink answer, which is interpreted on his or her machine.
5. Each student's ink answer and interpretation are transmitted to database via wireless peer-to-peer (P2P) network.
6. When instructor indicates end of student exercise period (e.g., orally), she clicks on aggregator icon, which causes aggregator to retrieve the interpreted ink answers, aggregate them, and produce summary data.
7. Summary data is transmitted to the database via wireless P2P network.
8. Summary data is displayed on instructor's machine.

Ink Interpreter

The ink interpreter (MRbeiz 2006) runs on the student Tablet PCs. We had hoped to use the built-in Microsoft recognition software, but found that for our domain its error rate of 27% was too high. Instead, we designed an architecture that combined new components with some of the components of the built-in software. One of the challenges of our architecture was to interpret intermixed handwritten text and arrows, since recognizers typically are able to recognize text or sketches, but not both in the same sequence of ink. The CLP ink interpreter uses a two-tiered architecture similar to that of MathPad: (LaViola, et. al 2005): (1) An interpreter performs recognition and semantic information extraction from digital ink; (2) a renderer renders and displays the semantic representation of digital ink. Unlike MathPad, however, our interpretation happens synchronously, i.e., after the user inputs ink. While asynchronous interpretation is usually faster, it requires the use of an event-based system that would have required major modifications to the underlying infrastructure. The difference in speed of synchronous vs asynchronous interpretation is negligible for the small size of our typical answers, so we deemed synchronous interpretation sufficient. CLP also differs from MathPad in that students do not see recognition results (though the developers do when debugging), as we do not want them to get distracted worrying about interpretation accuracy; we want the lesson to proceed without interruptions. Some ink misinterpretations will not affect the use of CLP in the classroom: The CLP aggregator (Smith 2006) takes into account interpretation errors by acknowledging the recognition confidence provided by the interpreter. Furthermore, CLP is designed to give the instructor an overall view of the understanding in the classroom, rather than an exact count of correct and incorrect answers.

Shown in Figure 2 is the CLP ink interpreter architecture. The current version does not segment text and sketches; it passes ink directly to our handwriting recognizer. The handwriting recognizer, which distinguishes text from arrows, produces a semantic representation for the ink. The ink interpreter currently under development will segment text and sketches, and pass corresponding strokes to the handwriting recognizer or the sketch recognizer.

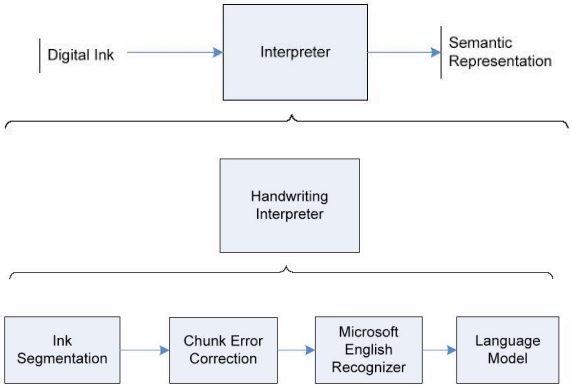


Figure 2. Ink interpreter architecture (Rbeiz 2006)

The handwriting recognizer works in the following way:

- The ink segmentation module, derived from Microsoft's Ink Analyzer, segments ink into individual chunks. Chunks are elementary units that in our current implementation are words or arrows.
- Our chunk error correction module attempts to fix errors common to handwriting recognizers: splitting a word into two words, or combining two words into one.
- The strokes of each chunk are then passed to the Microsoft English recognizer which outputs several hypotheses, ranked by a qualitative confidence score.
- The hypotheses are sent to the language model module, which uses a domain-specific dictionary and knowledge of expected exercise answer type in order to choose a best hypothesis. Supported answer types are: number, string, set, sequence, Scheme expression. Multiple-choice and true-false are subclasses of the string type. Scheme expression is a subclass of sequence; it uses a dictionary of Scheme terms.

Figure 4 shows CLP's ink interpretation for a student answer: Ink is segmented, passed to the English recognizer, then through the language model. An XML-like semantic representation is the result.

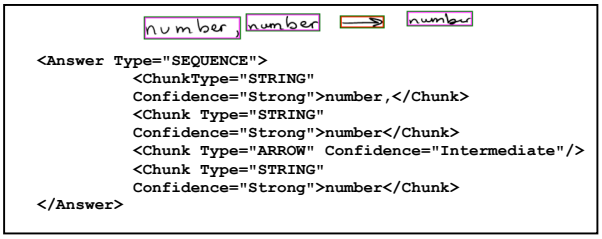


Figure 4. Segmentation and semantic representation

Recognition accuracy traditionally has been measured with a word error rate. In our research it was more appropriate to test the distance between the input and recognized strings in order to measure partial accuracy. If the input string is “caar”, for example, and the subsequent recognition results are “cr” and “car”, we want to take into account the similarity between the expected answer and the hypothesized one, something a word error rate cannot do. The Levenshtein distance (Atallah 1998), or edit distance, measures the distance between two strings, defined as the minimum number of operations needed to transform one string into the other, where an operation is an insertion, deletion, or substitution of a single character. In our example, the edit distance between “caar” and “cr” is 2 while the distance between “caar” and “car” is 1. In a controlled experiment, users were asked to ink 21 representative answers, and 167 inked answers were collected from several users. The inked answers were dynamically interpreted, stripped of spaces, and converted to lower case. They were then compared to the input. The current version of the handwriting recognizer performed with 249 characters errors out of 1820 characters, approximately 13% error rate. (See (Rbeiz 2006) more details.) The error rate for numbers was 5%.

Aggregator

The aggregator, which runs on the instructor's machine, is started by the instructor after students have submitted exercise answers. The aggregator retrieves student answers from the central database and, using the semantic representations produced by the ink interpreter, knowledge of expected answer type, and similarity measures associated with the type, groups the most similar answers together into equivalence classes. It outputs the classes, class sizes, and two example members for each class. The aggregator results displayer then displays a histogram on the instructor’s machine, with one histogram bar per class. (The displayer is discussed in the next section.) The aggregator is implemented in Lisp.

To place the student answers in meaningful groups, the aggregator compares student answers using similarity measures based on answer type. It is being designed to take two approaches: a top-down method and a bottom-up method. The top-down method first places all of the answers in a single group, then searches for the most logical split by picking two new group "centers" and dividing the rest of the answers based on which center is the most similar. The smaller groups are then split in a similar fashion until an instructor-determined number of groups has been created. The bottom-up method begins with each answer in its own one-member group and searches for the most logical way to merge two of the groups. The groups are merged until the instructor-specified number of groups remains. If the instructor has provided answers for an exercise, the aggregator uses these answers to determine the choice of splits and merges. If answers are not provided, the similarity measures determine the splits and merges.

An early design decision put most of the "smarts" in the aggregator rather than the ink interpreter. Consider the following student answers to a true-false question:

true True T t #t

The reasoning for equating all of these with "true" could happen in either the ink interpreter or the aggregator: The interpreter could have canonical representations for particular strings or characters and include those in its semantic representation; or the ink interpreter could leave

the ink as written, and the aggregator could contain synonym tables that it used when clustering student answers. We opted for the second approach, putting the intelligence in the aggregator since it was already reasoning about answer similarity.

Our current prototype, which employs the top-down algorithm, can aggregate numbers, strings, sets, sequences, and answers to true-false or multiple-choice questions (since these answers are treated as strings). It has been designed such that other question types may be integrated easily by writing the similarity measure for each new question type and by adding a small amount of dispatch code. The current similarity measure for numbers is the absolute value of the difference between the two numbers. Our string similarity measure uses a modified version of Levenshtein distance that assigns different costs to insertion, deletion, and substitution. The similarity measure for sequences employs a similar algorithm: The sequence itself is considered a string, with word-by-word comparisons made in the same fashion as character-by-character comparisons are made in strings. Sets are compared in the same fashion, but without penalty for differences in element order. For each similarity measure, the aggregator has a threshold for determining whether an answer is similar enough to be placed with other members in an equivalence class. This parameter is user-settable and currently has a value of 10%, meaning that answers must be at least 90% alike to be placed in the same class.

The aggregator has been tested using student answers from online problem sets for the introductory computer science course (XTutor). Its choice of grouping was examined for "reasonableness" and also compared against grouping performed by "human aggregators". Four questions, representative of those asked in class, were chosen from the online problem sets; two hundred student answers were selected for each question. One of the questions is shown below. (See (Smith 2006) more details.)

```
Question: Assume that we have evaluated the  following
 definition:
    (define fizz
      (lambda (a b) (* (/ a b) (/ b a))))
 Now, we evaluate the expression:
    (fizz (+ 1 -1) 1)
 What is the first step of the substitution model?

 Answer: (fizz 0 1)
```

Figure 5. Aggregator test question

When the student answers were aggregated for the above example, the largest bin contained answers that included the terms "define" or "lambda", indicating that students did not understand the meaning of those terms. The second largest bin contained the terms "a" and "b", indicating that students did not understand that they were to substitute the argument values of 0 and 1 into the expression. The remaining two bins contained answers that represented the second and third steps in the model, respectively, indicating that students were not aware of what constituted a step.

To compare machine and human aggregator results, 25 answers—about the size of a typical class—were chosen for each question, printed on paper, and given to four former 6.001 instructors and teaching assistants. These human aggregators were asked to group the answers using any similarity measures they chose. To ensure the 25 answers were representative of the entire set, the original 200 answers were aggregated and answers were drawn from each bin in proportion to the relative size of the bins. After grouping the student answers, each human aggregator was asked to explain his or her groupings.

For two of the four questions, the groupings created by the human aggregators were very similar to those created by CLP. For the question above, for example, the human aggregators created three groupings rather than four, combining into one bin the answers that indicated missed steps. For one question, two of the human aggregators produced groupings similar to CLP's and two did not. One question produced very different groupings even among the human aggregators, with the instructors and teaching assistants grouping based on the presence or absence of particular terms, length of answer, perceived misconception, or result of evaluating the student answer. Finally, the human aggregators thought CLP's groupings for each of the four questions "reasonable".

The conclusion is an obvious one: There is not necessarily one correct set of groupings for a particular set of answers. Instructors can control the grouping should they choose, however, by specifying answers for exercises created with the IAT. These answers then serve as "bins" for the aggregator.

Results Displayer

When the aggregator has completed its grouping of student answers into equivalence classes, it calls the results displayer to put up a histogram on the instructor's machine.

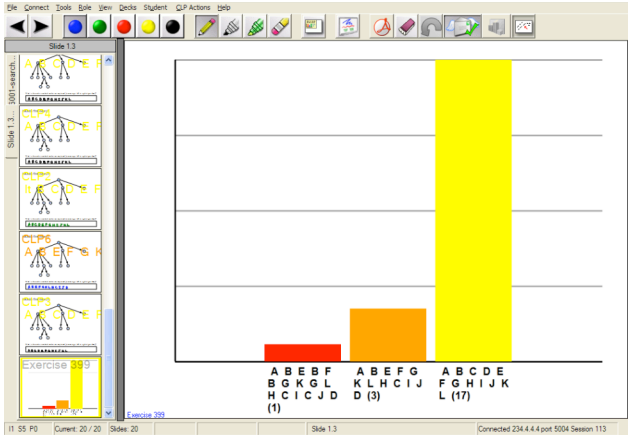


Figure 6. Aggregation results display

Each bar in the histogram, as shown above, is labeled with the number of items in the associated equivalence class and with a label derived from one of the elements in the class. In the current version of CLP, the items associated with each histogram bar are not directly

accessible via the bar. Instead, the instructor selects color-coded student answers that were sent to the instructor's machine using Classroom Presenter's underlying ink transmission mechanism. As shown in Figure 6, each answer is labeled with student machine name and ink interpretation, and each label is in the color of the bar to which the answer belongs. Because instructors can easily become overwhelmed with more than eight submissions (Anderson 2005a), it is critical that student submissions be stored not on the instructor's machine but in CLP's central database, and that only a small number of representative answers be sent to the instructor's machine. The version of the results displayer currently under development operates in this fashion.

Central Database

The central database, through which the various components of CLP communicate, embodies the ontology necessary to support interpretation and aggregation of in-class digital ink answers. The ontology centers around `Exercise` and `Answer` objects. The instructor creates exercise objects, which are stored in the database, using the Instructor Authoring Tool (IAT). Answer objects are created automatically in the database when students submit digital ink answers in class. The database is a relational SQL database, chosen for ease of integration with our C# code. We emulate an object-oriented database by representing object classes as different tables (Figure 7). Instances are represented as rows in tables.

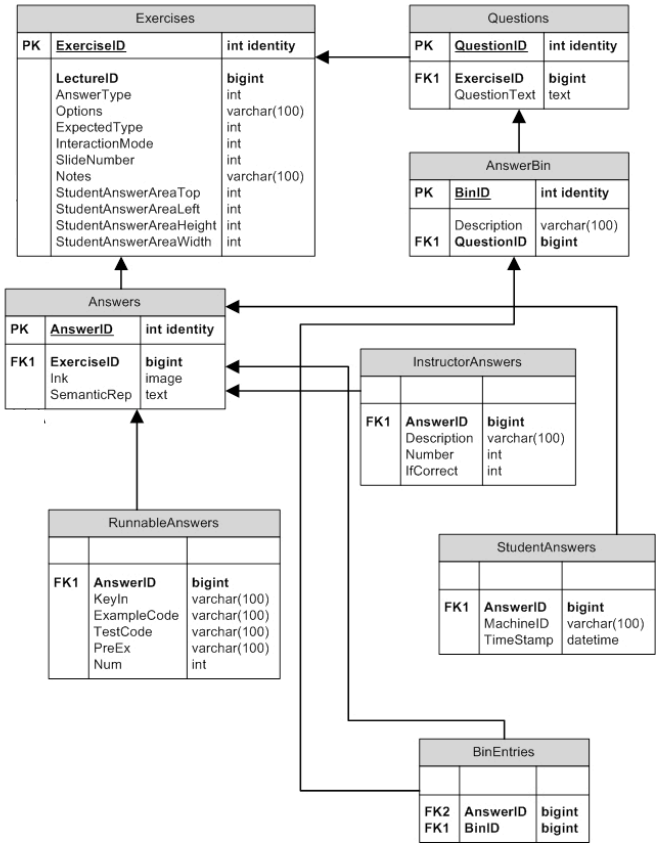


Figure 7. Portion of CLP database model

In our current implementation, the database runs on a dedicated Tablet PC, which also serves as a gateway to the Internet.[2] In an earlier implementation, the database ran on a remote server because it was deemed the option most reliable and secure. We had problems, however, with database submissions being lost due to unreliable network connectivity to the remote host. Since migrating to a dedicated local server, to which the instructor and student machines communicate via a wireless peer-to-peer network, we have not experienced loss of student submissions. The local server also has the advantage of allowing the instructor to be entirely self-sufficient in the classroom.

Classroom Evaluation

In parallel with our implementation efforts, we have been conducting studies to assess the impact on student learning of real-time Tablet-PC-based feedback and assessment of in-class exercises. As previously reported (Koile and Singer 2005, 2006), we have evaluated the hypothesis that the use of such a system increases student learning by: (1) increasing student focus and attentiveness in class, (2) providing immediate feedback to both students and instructor about student misunderstandings, (3) enabling the instructor to adjust course material in real-time based upon student answers to in-class exercises, (4) increasing student satisfaction. In three studies conducted in MIT's introductory computer science, we evaluated each of the above four parameters by means of classroom observation, surveys, and interviews. We also measured student performance using grades for exams, programming projects, problem sets, a final examination, and class participation. The performance results are summarized in this section.

Learning Studies. We have conducted studies in MIT's introductory computer science course, 6.001, during academic terms Fall 2005, Spring 2006, and Fall 2006. The course typically has an enrollment of 100 for Fall terms and 200 to 300 for Spring terms. Students taking the course meet in lecture for 50 minutes twice a week, in recitation classes of size 15 to 25 for 50 minutes twice a week, and in tutorial classes of size 5 to 7 for 50 minutes once a week. Our learning studies were conducted in recitation sections taught by the first author; classroom observation, surveys, and interviews were conducted by the fifth author. In each of the studies, Tablet PCs were deployed the sixth week of class—after the first exam—and used once or twice a week for the remaining nine weeks of the course. In Fall 2005, the first author taught one 6.001 recitation section in which Tablet PCs running Classroom Presenter software were deployed. In Spring 2006, the first author taught two 6.001 recitations, one that served as a control group in which she employed traditional teaching tools such as blackboard and overhead

[2] With Internet access, students are able to save class slides directly to their home directories.

projector; and one that served as the experimental group in which she used Tablet PCs running Classroom Presenter software. CLP was under development while these two studies were being carried out. Since the instructor used the same feedback and in-class assessment pedagogy as planned for CLP, we deemed the results of evaluating the use of Classroom Presenter to be relevant for CLP as well. In Fall 2006, the first author again taught two 6.001 recitations, one serving as a control group without Tablet PCs, and one serving as an experimental group with Tablet PCs and CLP software. In each of these three studies, the same real-time feedback and in-class assessment pedagogy was used in both control classes and experimental classes. The experimental classes, however, had the benefit of anonymous submission of student answers via Tablet PCs in place of traditional techniques such as calling on students to show their work on the blackboard. CLP's aggregator was used in Fall 2006 to aggregate confidence surveys that were given at the end of each class. It was not used for aggregating student answers to exercises because the ink interpretation rate was not high enough for meaningful results. (See next section for discussion of this issue.) Below is an example of a student answer.

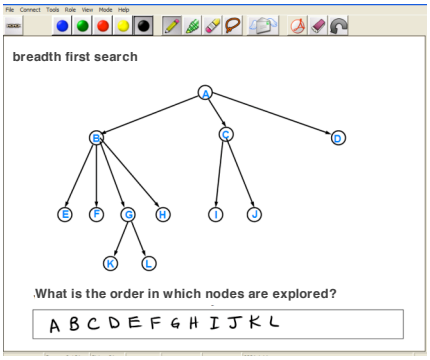


Figure 8. Student submission during a 6.001 class

Our findings in each of the studies have been consistent: In the classes using Tablet PCs and wireless transmission of answers to in-class exercises, fewer students than expected performed poorly (Koile and Singer 2005, 2006). Each of the following graphs illustrate the gap between the lowest performing students in the Tablet PC class and the non-Tablet-PC class. Note that in each study the lowest performing students were in the classes without Tablet PCs.

In Fall 2005, the non-Tablet-PC students were in other instructors' classes, so we were not able to control for teaching style. Nevertheless, the use of the Tablet PCs, plus submission of answers and immediate feedback pedagogy, seems to have benefited students at all levels of performance. When we controlled for teaching style, as in the Spring 2006 and Fall 2006 studies, there is a consistent pattern that the poorer performing students benefit most. The results are statistically significant for the Fall 2005 and Fall 2006 studies. The number of students participating in the Spring 2006 study was too small for statistical significance, though the pattern of performance in that study was consistent with the others.

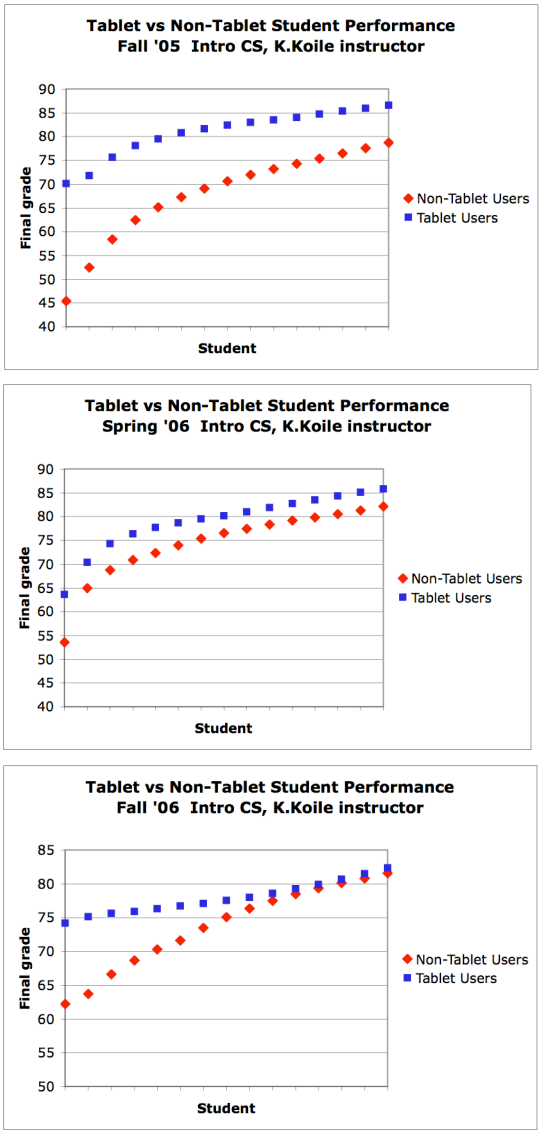


Figure 9. Graphs of student performance

In the Fall 2005 and Spring 2006 studies, we saw slight increases in the number of top performing students in the experimental class; we did not see this increase in the Fall 2006 study. This small or negligible increase indicates that either the top performing students are not benefiting from use of the technology (and would learn the material just as well without it), or that their benefits are not being measured. Top performing students in the experimental class, for example, could have learned the material more quickly than their counterparts in the control class.

We are repeating our learning study in Spring 2007, again with control and experimental groups in the first author's 6.001 recitation classes. We again will investigate the parameters mentioned earlier and the affect of Tablet PCs and CLP on student performance.

CLP in the Classroom. We deployed CLP in the Fall 2006 classroom study. As mentioned earlier, the first author used CLP's aggregator at the end of each class for aggregating confidence scores, numbers on a scale from one to seven that indicated a student's confidence in understanding particular concepts. The expected answer types were numbers, whose ink interpretation error rate as noted earlier is 5%. That rate was high enough for useful aggregation of student submissions. The instructor used the aggregated results at the end of each class to gauge the students' understanding and to guide lesson planning for the next class. Unfortunately, the ink interpretation was not accurate enough for the aggregator to be used for many other exercises. The `sequence` answer type, for example, was not specific enough to achieve good recognition for many answers. When interpreting the sequence of single characters shown in Figure 8, for example, the Microsoft English recognizer returned a sequence of multi-character strings rather than individual characters, presumably because it was trained to search for words.

Current Work

We are continuing our two lines of research: further development of CLP, and conducting another learning study this term. We again plan to deploy Tablet PCs in one of the first author's two 6.001 recitations. We will deploy a new version of our ink interpreter, one that we expect will have improved recognition rates because it includes more specific answer types. Such new types, e.g., `sequence of characters`, will improve interpretation rates by supplying more information for the interpreter to use when choosing among hypotheses. We also will add to the IAT a mechanism for storing instructor-specified answers, to be used by the aggregator as clustering "centers". We will compare the equivalence classes produced using the instructor-specified centers with classes created using aggregator-chosen centers.

In addition, we are working on the interpretation and aggregation of sketched answers, such as the box-and-pointer diagram shown in the introduction. We also are working on the interpretation and aggregation of what we are calling marked answers, such as a circle around a multiple-choice answer or on a map. These marks differ from sketches in that their semantics is dependent on an associated background image.

In summary, the goal of Classroom Learning Partner is to increase student-instructor interaction and student learning by supporting real-time feedback and assessment of in-class exercises. The Tablet PC has proven a very effective platform for this pedagogy, enabling facile and natural interaction for both instructors and students. Through our continued research and deployments we hope to continue to demonstrate the benefits of improved student learning, especially in classes currently too large to take advantage of real-time feedback and in-class assessment.

Acknowledgments

The authors thank MIT iCampus (http://icampus.mit.edu) for funding this project, and Hewlett Packard and MIT Academic Computing for generously donating Tablet PCs. We thank Howard Shrobe for advice on this research.

References

Anderson, R., Anderson, R., Simon, B., Wolfman, S., VanDeGrift, T., and Yasuhara, K. Experiences with a tablet-pc-based lecture presentation system in computer science courses. In *Proc. of SIGCSE '04*.

Anderson, R. Personal communication. 2005.

Angelo, T.A. and Cross, K.P. *Classroom Assessment Techniques: A Handbook for College Teachers*. Jossey-Bass Publishers, San Francisco, 1993.

Atallah, M. J. (Editor), *Algorithms and Theory of Computation Handbook*, "Levenshtein Distance (13-5)", CRC Press, 1998.

Berque, D., Bonewrite, T., Whitesell, M. Using pen-based computers across the computer science curriculum. In *Proc of SIGCSE '04*, pp. 61-65.

Bloom B.S. *Taxonomy of Educational Objectives, Handbook I: The Cognitive Domain*. New York: David McKay Co. Inc., New York, 1956.

Bransford, J.D., Brown, A.L., and Cocking, R.R., Eds. *How People Learn: Brain, Mind, Experience, and School.* National Academy Press, Washington, D.C., 1999.

Draper, S.W. From active learning to interactive teaching: Individual activity and interpersonal interaction. In *Proc. of Teaching and Learning Symposium: Teaching Innovations, 2004*, The Hong Kong University of Science and Technology.

Koile, K. and Singer, D. A. Development of a tablet-pc-based system to increase instructor-student classroom interactions and student learning. In Impact of Pen-based Technology on Education: Vignettes, Evaluation, and Future Directions. D. Berque, J. Prey, and R. Reed (eds). Purdue Univ Press. 2005.

Koile, K. and Singer, D.A. Improving learning in CS1 via tablet-pc-based in-class assessment. In *Proceedings of ICER 2006,* September 9-10, 2006, University of Kent, Canterbury, UK.

LaViola, J. and Zeleznik, R., *MathPad: A System for the Creation and Exploration of Mathematical Sketches,* Brown University, 2005.

Ratto, M., Shapiro, R.B., Truong, T.M., and Griswold, W. The ActiveClass project: experiments in encouraging classroom participation. In *Proc. of CSCL '03*.

Rbeiz, M. A. Semantic Representation of Digital Ink in the Classroom Learning Partner. MIT EECS Master of Engineering thesis. May, 2006.

Smith, A. C. Aggregation of Student Answers in a Classroom Setting. MIT EECS Master of Engineering thesis. August, 2006.

Steadman, M. Using classroom assessment to change both teaching and learning. *New Directions for Teaching and Learning*, 75, 1998, pp. 23-35.

Wilkerson, M., Griswold, W., and Simon, B. Ubiquitous presenter: increasing student access and control in a digital lecturing environment. *In Proc of SIGCSE '05*, pp. 116-120.

XTutor. http://icampus.mit.edu/XTutor/