UW CSE TR# 2006-10-02.

A Study of Digital Ink Student Artifacts to Inform the Scaling of a Classroom Interaction System

Richard J Anderson
Department of Computer Science and Engineering
University of Washington

Ruth Anderson
Department of Computer Science
University of Virginia

K M Davis
Department of Computer Science and Engineering
University of Washington

Craig Prince
Department of Computer Science and Engineering
University of Washington

Valentin Razmov
Department of Computer Science and Engineering
University of Washington

Beth Simon
Computer Science and Engineering Department
University of California, San Diego

ABSTRACT

This paper studies digital ink artifacts students produced in the classroom and how instructors could use these artifacts in support of classroom instruction. Currently, instructor use of student-produced artifacts is limited by the cognitive load of real-time review and analysis during class. The goal of the study is to evaluate, in the context of a TabletPC-based classroom interaction system, whether clustering techniques have the potential to assist instructors in this task. We examine student ink artifacts to determine whether they could be naturally grouped into categories that instructors find useful when discussing student work. We establish that grouping student artifacts plays a major role in how instructors use them in class, and that the artifacts often have an underlying grouping structure. This paper looks at the complexity of algorithms for grouping and also identifies several challenges that arise in analyzing student artifacts.

AUTHOR KEYWORDS

Educational Technology, Ink Recognition and Clustering, Pen Based Computing, Classroom Pedagogy

ACM Classification Keywords

H.5.3 [**Group and Organization Interfaces**]: Collaborative computing, synchronous interaction.

INTRODUCTION

Mobile computing and wireless networks are transforming many activities by allowing instantaneous transfer of rich data between users. Our domain of interest is the higher education classroom. By increasing the communication bandwidth in the classroom, we seek to improve the educational experience by:

- Making instructors more aware of the level of understanding of their students;
- Lowering the barriers to contributing so that all students can participate;
- Integrating student work into classroom discussions to promote peer learning.

Our approach is to deploy pen-based computers in the classroom, and use them to enable students to send digital ink artifacts to the instructor at specific times during the lecture. The idea is that the instructor quickly reviews these as they arrive during class – for instance, to determine the students' level of understanding – and then displays some of the student contributions to the class for further discussion. This raises the concern of overloading the instructor with data. One possible solution is to use ink recognition and clustering of student artifacts to make this data easier for the instructor to work with. Although such automatic techniques sound promising, prior to investigating them further, we conducted a background study to assess the applicability of these techniques to the types of data and activities we have collected from real classes.

Classroom Technology

Our goal in deploying technology in the classroom is to increase the level of interaction between the students and the instructor. There is a vast educational literature that addresses the difficulties of engaging students with the traditional university lecture [9, 27]. Common themes for addressing these difficulties include introducing student activities [10, 15] and giving the instructor feedback on student learning [7].

There is broad interest in using technology to improve the classroom experience. The idea behind much of the work on classroom technology has been to use technology to enhance or offload certain activities, so that students and instructors can be more effective in the classroom. Opportunities for this include capturing the classroom experience to reduce note taking demands [1, 21], improving presentation tools for the instructor [5], and creating new communication channels for student-student [17] and student-instructor [12, 23] communication.

Our Approach

To support our goals, we have developed Classroom Presenter [5, 26] – a distributed Tablet PC-based classroom interaction system. The system supports sharing of digital ink written on electronic slides. Using a digital pen, the instructor writes on top of a slide on a Tablet PC and the ink appears simultaneously on a public display. This allows the system to be used as a presentation tool that provides dynamicity to traditional PowerPoint-style lectures by enabling ink augmentation of slides. Classroom Presenter also supports sharing of information with student devices: the students' slides can be synchronized with the instructor's slides and receive the instructor's ink in real time. Students can also write on slides and send the result back to the instructor anonymously. We refer to this as the *student submission* scenario. The instructor can then choose to show some of the submissions on a public display.

Student submissions are central to the pedagogy we are developing around the use of interacting devices. The instructor develops a slide-based lecture and includes a number of activities on the lecture slides. When the instructor reaches an activity slide, students write their answers on the slide with digital ink, and send the slide and ink back to the instructor. The instructor can then review the submissions and selectively show some on the public display. This allows the instructor to bring in a diversity of ideas, show novel solutions, and discuss misconceptions arising from student answers. The use of a public display creates a focus of attention and provides a mechanism whereby student work can be integrated into the lecture discussion[1] – one of the most powerful aspects of the student submission process.

To make the workflow more concrete, Figure 1 shows the instructor interface, with the current slide, a filmstrip for slide navigation on the left, and ink controls across the top.

[1] The student work appears on the public display anonymously. The current implementation of the system shows the name of the machine that a submission came from on the instructor's view of the submitted slides.

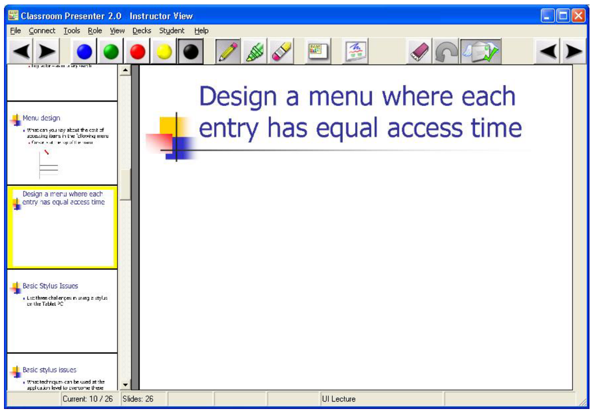

Classroom Presenter 2.0 Instructor View
File Connect Tools Role View Decks Student Help
Menu design
Design a menu where each entry has equal access time
Basic Stylus Issues
Basic stylus issues
Design a menu where each entry has equal access time
Current: 10 / 26
Slides: 26
UI Lecture

Figure 1. Instructor view, showing an exercise for students to work on.

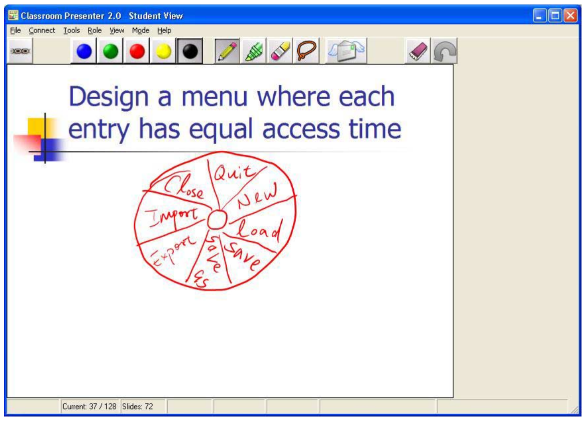

Classroom Presenter 2.0 Student View
File Connect Tools Role View Mode Help
Design a menu where each entry has equal access time
Quit
New
Load
Save
Save As
Export
Import
Close
Current: 37 / 128
Slides: 72

Figure 2. Student view after the student has completed the exercise.

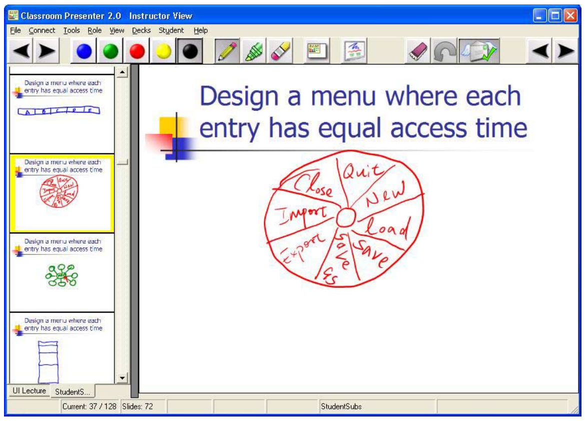

Classroom Presenter 2.0 Instructor View
File Connect Tools Role View Decks Student Help
Design a menu where each entry has equal access time
Quit
New
Load
Save
Save As
Export
Import
Close
UI Lecture
StudentS...
Current: 37 / 128
Slides: 72
StudentSubs

Figure 3. Instructor view after student responses have been received. Student submissions are in the film strip, while the selected slide shows in the main panel too.

At the start of the lecture, students receive the slides on their machines. Figure 2 shows one student's view after the student has responded to the given question. The student sends the answer back to the instructor with an explicit action. Figure 3 shows the instructor view after student submissions have been received. The filmstrip now contains the answers submitted by the students. The instructor can preview the submissions by scrolling the filmstrip, and also through a preview window. When the instructor selects one of the submissions from the filmstrip, it shows on the main panel of the interface, and also on the public display.

Benefits of Ink-Based Input

Classroom Presenter targets the Tablet PC platform. Using a pen-based device in a classroom environment brings significant advantages. Digital ink provides flexibility of expression that allows a wide range of "constructional" activities which would not be possible with "clicker" or keyboard-based devices. Pen-based input allows students to draw diagrams, use symbolic notations, and annotate on top of existing content. The naturalness of the act of writing also encourages useful "side channels" of student input. We have seen students use ink to display their initial stages of thought, to elaborate on an answer, and to express their personalities through handwriting and drawings. The rich expressiveness of ink is at the core of Classroom Presenter, yet using ink also creates challenges in recognition and analysis that are a theme of this paper.

Related Work

There are a number of other projects exploring the use of student devices in the classroom to support interaction. In terms of technology, the LiveNotes project [17] has taken a similar approach to Classroom Presenter by sharing ink and slides between Tablet PCs. However LiveNotes aims at a different scenario – that of supporting student group communication in the classroom. ActiveClass is a project where students use PDAs to deliver asynchronous feedback to the instructor [23]. Although not the focus of this paper, we have also investigated asynchronous feedback on slides in our Classroom Feedback System [6]. Many systems for distance or web-based education incorporate various polling and feedback facilities [14]. There has been a substantial amount of work studying the use of Classroom Response Systems [12, 20], which we discuss below.

The most mature work on supporting classroom pedagogy with mobile devices is the work on Classroom Networks [24, 25]. A classroom network consists of a collection of student devices which communicate with an instructor machine connected to a public display. The public display is used to show aggregated data coming from the student devices. Pedagogies such as Peer Instruction [20] have shown impressive gains in student achievement through use of such classroom networks. The key to successful use of a classroom network is to design instruction around activities that can then be discussed in the context of the displayed information [16]. Classroom networks to date have generally relied on discrete devices such as "clickers" [12] which support multiple choice questions and polling. Our work contrasts by providing a richer mechanism for student expression, and relying on the use of non-aggregated data. As a result, the types of activities supported by Classroom Presenter and Classroom Networks are very different and indeed complementary.

BASIC QUESTION

The successful use of this technology in the classroom depends both on the design of appropriate activities and on the instructor's ability to work effectively with student submissions. Instructors in our study have found analyzing student responses in "real-time" during lecture to be surprisingly challenging in some instances – even in classes with only 10-20 student devices. Often the majority of responses are submitted simultaneously and towards the end of the activity. Instructors are then under pressure to quickly determine overall class understanding and plan which individual submissions to display in order to make the desired points. A standard concern often raised is whether the instructor can cope with a large number of submissions as the system scales to larger deployments. Based on our experience the scaling issue is indeed a very real problem. We have seen comprehension issues arise in classes with 10-20 student devices and increase as more devices are added.

Direct evidence of the challenge of understanding submissions comes from comparing instructors' impressions during the class period with a careful post-class review of the submissions. In an evaluation of activities (described in a later section), instructors often commented that analyzing results in real time was more cognitively difficult than they had anticipated. There were frequent cases where instructors perceived a fairly different ratio of correct to incorrect answers from what the reality was, even when it was easy to quickly judge the correctness of individual answers. In addition to their inability to see the "big picture," instructors had trouble honing in on interesting details. Creative or otherwise outlying solutions often went unnoticed, prompting a reaction of "I wish I had seen that example" from the instructor during the post-class review. In other cases instructors struggled to locate an example of a specific type of answer they wanted to display to illustrate a particular (prepared) point.

There are obvious improvements to the user interface that could assist the instructor in working with student submissions. The current instructor interface (shown in Figure 3) shows submissions in a one-dimensional scrollable filmstrip. One idea for improvement is to dedicate the entire screen area (at least momentarily) for a two-dimensional view of student submissions. This would both increase the number of submissions that could be realistically viewed and allow improved organization of the displayed responses via the added dimension. Figure 4 shows an example of how submissions might appear when they are organized by the shape of the inked solution. For this sort of grouping to be useful to instructors during lecture, it would be necessary to have algorithmic methods for automatically computing the groups. There is a very large space of available approaches to recognition and grouping, and vast previous work to build on. Specific issues to consider include recognition technologies [2, 13, 22], algorithms for clustering [19], mechanisms for guiding group construction, handling of specific domains [3, 18], and techniques for visualizing the results [8]. Our hypothesis is that the key to developing a user interface that supports the effective use of student submissions in large classes is grouping submissions in a way that brings together submissions with the same meaning, or ones that could be used to illustrate the same point.

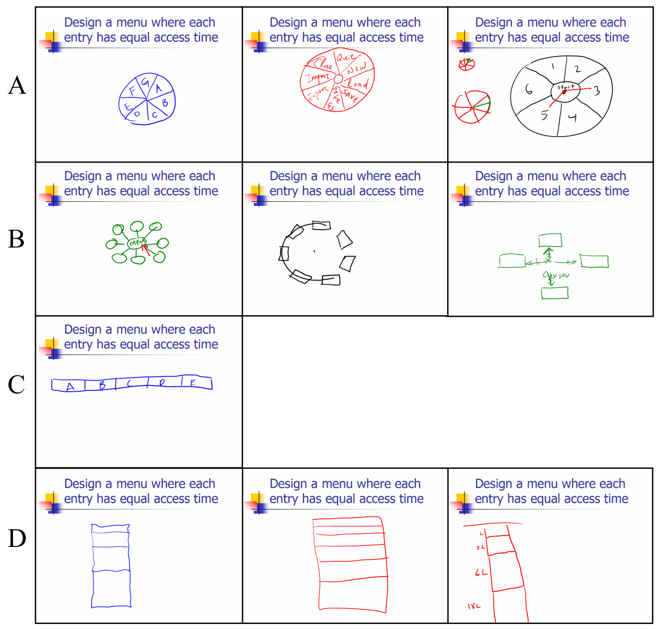


Figure 4. Grouping of student submissions in a two-dimensional grid with rows showing individual groups. In this activity, students were asked to draw a menu with equal access times for items according to Fitts' law predictions. Row D shows the type of answers the instructor was expecting.

This paper is about our investigation of the hypothesis that grouping has a central role in working with student submissions. We do this without attempting to implement any specific recognition and clustering algorithms. There are several reasons why we chose to investigate this problem before prototyping and evaluating specific such algorithms:

- The number of algorithms available for clustering and recognition is very large, so a better understanding of use could guide the choice of algorithms.
- The recognition problems are technically challenging, so prototype results could easily emphasize the quality of our implementations, and not illuminate the more interesting question of whether the general approach enhances the instructor's comprehension. At the same time, negative results could be attributed to not having good enough recognizers, rather than to a possible shortcoming of the approach.
- Because of the challenges of producing good recognizers, prototypes would have to be aimed at fairly narrow domains, which might not be broad enough to support a convincing evaluation in a real educational setting.
- It is possible that grouping does not play the key role that we hypothesize it does. If so, this would not be exposed by an implementation-based study.

The underlying question we investigate is whether it is possible to automatically group student submissions in a way that they are useful for the instructor. We break this question into three components – whether instructors actually depend on grouping, whether real data (i.e., student artifacts) has a grouping structure, and how hard it would be to compute useful groupings. Our analysis follows from falsifying the hypothesis and considering the main reasons why automatic grouping might *not* be useful in working with student submissions. First of all, it is possible that instructors do not rely heavily on constructing groupings of submissions, in which case grouping may not help. It is also possible that the type of data received from students does not exhibit a useful group structure – even if the instructor is looking for one. The final concern is that the algorithmic complexity of finding desired groupings may be too great so it is not feasible to compute them. In subsequent sections we examine these three aspects of grouping in light of actual classroom data.

FRAMEWORK OF THE STUDY

We have been piloting the system in several junior and senior level Computer Science courses at our institution on an experimental basis since December 2004. The courses were Data Structures, Software Engineering, Digital Design, and HCI. The purpose of these deployments has been to explore uses of the technology and the associated pedagogy it affords. The main classroom experience discussed in this paper comes from four instructors who have used the system in six different course offerings for a total of 28 class sessions. In these deployments the instructor lectured from a Tablet PC, connected to a public display, while all students could see and annotate the lecture slides on Tablet PCs on their desks. We usually had about 20 HP TC1100 Tablet PCs deployed in the classroom, communicating over an ad hoc or infrastructure-based wireless network. In courses larger than 20 students, we had students share devices. In addition to the courses discussed here, we have occasionally deployed the system in other courses and in numerous non-classroom demonstrations. In total, over 1100 student submission artifacts were collected from 87 different submission activities.

Our study centered on artifacts collected in the classroom[2]. Collecting the student submissions was an easy process, since the system allowed saving submissions from the in-

[2] For this paper, we only consider submissions collected from actual college classes. We have a large amount of data from demos too, but since the demo environment is very different from a classroom, we do not include activities from it here.

structor machine. This preserved all submissions made by the students (although it did not retain information on which students they came from or the timing of the submissions). The dataset was then cleaned by erasing ink drawn by the instructor on the submission and removing duplicate student submissions. A few activities were used in multiple course offerings; they were combined into single activities for the purposes of our study. This yielded a total of 87 activities. From this set, we selected 36 for more careful analysis. This subset was chosen to be representative of the courses included in our study and of the different types of activities in the full set (including ones that required textual responses and ones that asked for sketching diagrams). Later sections describe our analysis of the activities.

The data was obtained through real classroom usage during a time when the instructors were developing and refining the methodology of teaching with students submissions. The data was collected before we decided to conduct this study. Although there are ways to design activities to facilitate the automatic analysis of the resulting artifacts – for instance, by making it easy to identify individual answers to questions – the activities discussed here were not designed with that goal in mind, and consequently introduced additional challenges for the analysis.

Across all instructors involved and all different courses taught, we have seen considerable variation in the types of activities used and the types of responses received from the students. The most successful activities were well thought out to fit into the lecture and designed so that the range of likely answers could be easily understood by the instructor when he or she received them.

Overall, students responded very positively to the system. In one of the courses surveyed 43 out of 44 students thought the system had a positive effect on their learning experience. 40 out of 44 students felt that seeing other student solutions had a positive effect on their learning experience, although 20 of these students admitted that they would only have volunteered to show their answers to the class less than half the time if ever. In other courses students made observations such as "it gives equal voice to the quiet person and the one that talks a lot" and "The best thing about this system is it encourages the students to actually work on the problem . . . Knowing that my solution will appear on screen but will also remain anonymous encourages me to participate but at the same time reduces the worry of getting it wrong."

INSTRUCTORS' VIEW OF GROUPING

We started our investigation by examining the role that grouping plays for instructors who use student submissions. If grouping was not significant for them, it would not be worth pursuing methods to automatically group the data. Intuitively, it seems that grouping of student answers could be useful for the instructor – anyone who has graded homework recognizes that evaluating answers can often be done very quickly by placing similar answers together into bins. The higher education instructional literature also supports the idea that grouping is important in analyzing students' work. Angelo and Cross's text on Classroom Assessment Techniques (CATs) [7] is one that has greatly influenced our thinking on student submissions. Throughout their work they stress the role of breaking the student responses into groups, with suggestions such as:

> *Student responses to this type of CAT can be quickly sorted into three piles: correct/complete (or "on-target") responses, somewhat correct/complete (or "close") responses, and incorrect/incomplete ("off-target") responses. Then the number of responses in each pile can be counted, and the approximate percentage of the total class each represents can be calculated. Teachers also can look for particularly revealing or thoughtful responses among the on- and off-target groups. [p. 30]*

The basis of our study of instructor requirements was an in depth analysis of activities previously used in class. For each of the 36 activities studied, we asked the original instructor a set of questions in order to understand what their goals were for the activity, how they made use of student responses in class, and whether grouping responses would have helped them in this use.

For 33 of the 36 activities instructors indicated that having student responses placed into groups would have been useful during lecture, especially if the activity was used in a large class. This result held across a wide variety of activities and uses of student submissions. The preferred types of groups depended on the instructor's goals for the specific activity. For activities whose goal was to assess student understanding or point out misconceptions, groups such as "Correct", "Partially Correct", and "Incorrect" were often desired. For activities meant to generate a variety of artifacts of interest, the desired groupings resulted from partitioning by particular features (e.g., by shape: "linear", "star", "binary tree"; by coding approach: "recursive", "iterative", "other"; etc.). The maximum number of groups an instructor deemed useful for any one activity was 5 with 3 groups being most commonly considered an optimal number.

Instructors often described their use of student submissions during class in terms of displaying submissions from several different categories. One instructor noted: "I planned to show examples of the different types – and draw any additional examples that were necessary." Among the presentation strategies used, instructors were almost evenly split between planning on displaying only one solution (often a correct one), displaying several partially incorrect ones, culminating in showing a correct one, and displaying one solution from each expected type. One instructor employed the strategy of displaying every student submission, which was possible in smaller classes where no more than 15 Tablet PCs were used. Even this instructor agreed that for most of his activities, grouping would be useful in larger classes and deployments.

Based on the instructors' responses, we feel the evidence is strong that grouping could play a major role in how student submissions are used in the classroom.

DOES THE DATA FORM GROUPS?

In the previous section we argued that in many cases instructors looked for a grouping structure among student submissions. The next logical question is whether such an underlying grouping structure exists in the data.

To test this, we had six individuals participate as sorters and later analyzed how similar the groupings they produced were overall, as well as how similar they were to the original instructor's preferred grouping. All sorters had a Computer Science background and were familiar with the subject matter of the courses used in the study. Their task was to sort a collection of artifacts into groups in a manner that would be useful for an instructor. We did not specify ahead of time how the groups should be constructed or what the number of groups should be, so each sorter used the criteria they deemed best for the activities at hand.

Our decision to have humans do the sorting for this part of the study was motivated by the desire to sidestep the technicalities associated with handwriting and diagram recognition until a time when addressing such details would be more appropriate. Also, since humans have the ability to see the high-level semantic meaning behind an artifact – something an automated algorithm is unlikely to be able to infer – if humans failed to discern any useful group structure, automated algorithms would have little chance of success too.

Each of the 36 activities was sorted by three to five people. We used the *edit distance* metric [11] to evaluate the similarity between pairs of sorts. In the context of this study, the edit distance between two given sorts is the minimal number of student artifacts that would need to be moved between groups in order to convert one of the sorts into the other. Therefore, a low edit distance corresponds to a high degree of similarity between two sorts. Table 1 shows the results for a set of representative activities. For each activity, the table displays the average of the edit distances from the activity's original instructor to each sorter, as well as the average of the edit distances taken across all pairs of sorters (including the activity's original instructor). The edit distance is also shown as a percentage of the total number of artifacts submitted for that activity. The activities were taken from courses in Human Computer Interaction (HCI), Data Structures (DS), and Software Engineering (SE). The types of activities are diagram/tree/graph sketching (D), English text writing (T), or coding (C).

Activity	Course	Type	Average edit distance to instructor sort	Average edit distance across all sorts
EqAccess	HCI	D	1.4 (10%)	1.7 (12%)
FindCode	DS	C	1.0 (10%)	1.4 (14%)
BehindWriteIn	SE	T	2.8 (13%)	3.1 (15%)
ComGraph	SE	D	3.4 (17%)	3.7 (19%)
HuffmanTree	DS	D	4.5 (21%)	4.9 (23%)
FailReason	SE	T	18.6 (42%)	18.5 (42%)

Table 1. Activities and Edit Distances. The edit distances are expressed both as the absolute number of edits, and as a percentage of the total submissions for the activity

The results show, at least for some activities, significant similarities between groupings done by different sorters. This suggests that it is promising to look for algorithms to compute such groupings. In the next section we discuss in more detail each of these particular activities, along with the corresponding results from the table.

AUTOMATIC GROUPING

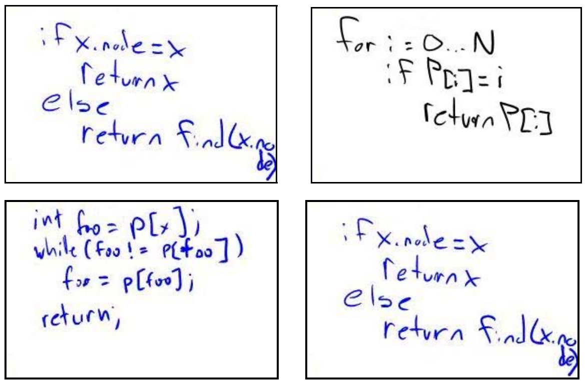
The final part of our study assesses the prospects for automatic grouping of student submissions by examining data collected from the classroom. Our method could be called "backwards analysis"; we analyze the student submissions to determine what type of algorithms could have been able to construct the types of groupings that were found by the human sorters. The goal of this analysis is to gain insight into how hard it would be to construct algorithms that would find groupings for certain classes of student submissions by looking at specific instances. This analysis can illuminate specific difficulties arising for student submissions and can provide evidence about the overall difficulty of the grouping problem.

It is widely known that there are many challenges in ink recognition, and that recognition becomes more challenging in informal domains [4]. While handwriting recognition is now relatively good, there are major challenges in diagram recognition [2, 18] and mathematical expression recognition. We observed student submissions that demonstrated challenges common to other informal domains such as sloppy handwriting, stroke segmentation difficulties, and multiple drawn strokes.

In the following subsections we examine in detail the potential for creating automatic grouping algorithms for student activities. We discuss specific examples of activities whose answers were expressed as code, text or diagrams, and illuminate possible approaches and potential difficulties.

Code

The *FindCode* activity (shown in) asked students to write code to traverse a tree from a node to the root. This



```
if x.node=x
   return x
else
   return find(x.node)
```

```
for i = 0...N
   if P[i]= i
      return P[i]
```

```
int foo = p[x];
while (foo != p[foo])
   foo = p[foo];
return;
```

```
if x.node=x
   return x
else
   return find(x.node)
```

Figure 5. Code written by students to traverse a tree from a node to the root in the FindCode activity. (A) shows a recursive solutions, (B) is an incorrect for-loop solution, and (C) and (D) show two iterative solutions.

activity was used to make a tie between data structures and implementation, and to allow the instructor to make the point that code for this operation can be very concise. In class, the instructor was surprised by the division of the submissions into groups: six were recursive, three were iterative, and one was an incorrect solution. This was an unanticipated grouping of the data (the instructor had only expected iterative solutions). During class it was easy for the instructor to distinguish between recursive and iterative solutions, but not to keep track of the percentage of each type or to notice the one incorrect solution. The instructor's preferred groupings were very close to the other sorters' groupings, with an average edit distance of 1, showing that, on the average, only a single submission was categorized differently.

In general, automatic analysis of handwritten code is a very difficult problem. Static analysis of code is a deep problem in software engineering, and there is also a hard recognition problem in going from handwritten code to text. The examples in show a number of syntactic irregularities which would make recognition more difficult. Despite the difficulty of the general problem, there is a simple heuristic algorithm which would have found the desired clustering in this case. Each of the groups could be easily identified by keywords. The algorithm would first recognize the programming language and then cluster based on the presence or absence of certain keywords. The recursive group could be identified by "if-else", the iterative group by "while" and the incorrect group by "for". The significance of this example is that it shows that there are cases where simple heuristics give the same results as a clustering based on a deeper understanding. Further investigation is needed to determine how broad this phenomenon is.

Text

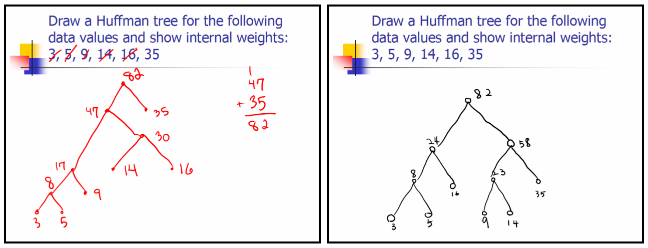
There is a broad class of activities where the answers are expressed in short phrases or sentences of text. Approximately 30 percent of the activities in our collection requested free-form textual answers. Instructors generally wanted a grouping "by the same meaning", although occasionally had a fixed collection of semantic categories. The number of groups found by instructors was generally significantly higher than for diagrammatic activities. The planned classroom usage was also different – aiming to show a greater number of answers, covering the range of available responses. The edit distances were higher for the manual clustering of textual submissions than for other types of activities. Average edit distances were as high as 40 percent. The edit distance measure is generally sensitive to increasing the number of groups, but this does show that there was substantial variation in how individuals sorted the data. Some submissions were consistently grouped together, but some outliers were ambiguous in meaning and therefore sorted differently. Sorts of the activity *FailReason* (not shown here) had an edit distance of 42%, indicating high variation among sorters. However, the instructor commented that the groupings generated by other sorters were "surprisingly useful," despite the fact that they differed significantly from his own sort. Thus, there may be value in generating automatic groupings even when there is no clear agreement on the ideal sort.

In *BehindWriteIn*, the instructor in a software engineering class asked students what the options were if a company fell behind on their schedule. Answers received included "hire more people", "bring in consultants", "move estimate back", "drop low priority features", and "work harder." The answers mostly consisted of short phrases. Although many algorithms already exist to cluster textual documents, including much of the work motivated by web searching [28], we were concerned that the shortness of the text in a student submission and the comparatively small sample size would limit the applicability of these algorithms to our domain. A very simple approach we examined was to first remove extraneous "stop words" and then begin combining elements into groups which either shared words or shared equivalent words. This algorithm might fail to bring together answers such as "hire more people" and "bring in consultants" if there was not a sufficient overlap to have them merged into the same group. (In our hand simulations, these two did end up in the same group, because the answer "bring more people onto team" overlapped with both.) This particular activity was one of the small text-based activities in our collection, with only 21 total submissions, each composed of no more than a single sentence. Yes, this was still an adequate amount of text to use for grouping. There were a handful of submissions that had no words in common with others, but that was the exception rather than the rule. Hand simulation of this algorithm produced results which were comparable in terms of edit distance to the groups produced by human sorters. In retrospect, this initial experiment with text clustering produced much more positive results than we had anticipated.

Diagrams

The *HuffmanTree* tree activity (shown in Figure 6) was used in a Data Structures course to evaluate whether or not students understood the standard algorithm for building Huffman trees. After the algorithm was introduced and the students saw an example of building a Huffman tree, they were given an activity that asked them to build such a tree from a given set of weights. The instructor planned to show several incorrect solutions to address misconceptions, followed by a correct solution. The desired grouping of solutions was a fairly common pattern: Correct, Almost Correct, and Wrong. The instructor's sort of this activity was Correct (13), Almost Correct (3) and Wrong (4). When the instructor did this activity in class, he had difficulty finding incorrect solutions to display. The sorters matched the instructor's groups quite well, with fairly close agreement on the Correct group. The average edit distance for the sorts, as shown in Table 1, is 4.5. The main contributor to this distance was the variety of ways sorters differentiated between almost correct and incorrect solutions.

The natural approach to automatically grouping this algorithm would be to apply a tree recognition algorithm, and then cluster on tree structure. The correct solutions would all be put into the same group, since they would all have the same structure. Based on the disagreements among the sorters on the incorrect solutions, it is unlikely the algorithm would do well on the Almost Correct / Wrong distinction – but determining the percentage of Correct answers and separating them from the incorrect ones would have been sufficient for this instructor's goals. There has been a substantial amount of work on recognizing domains such as trees [3]. It is still a challenging engineering task to develop a robust recognizer for such a domain that handles the natural variations in drawing. Although a tree recognizer could be a basis for grouping this type of activity, we had the hope that there might be a simpler approach – just as in the *FindCode* problem we could group by keywords instead of by algorithm understanding. Unfortunately, the simple approaches we looked at, such as clustering based on the number of edges, on the distribution of left and right edges, or on the length of the longest path, did not lead to useful groupings. We could not find any structural properties, other than computing the full tree that would result in the right grouping. Approaches based on looking at the weights on the nodes might work better, although in the data set we had, students had been very inconsistent in labeling nodes. This example points to needing fairly sophisticated recognizers in order to achieve decent results. We believe it is also likely that the same result would hold for many domains in other disciplines, such as chemistry diagrams and circuit drawings, where high quality recognizers would be required and would achieve good results.

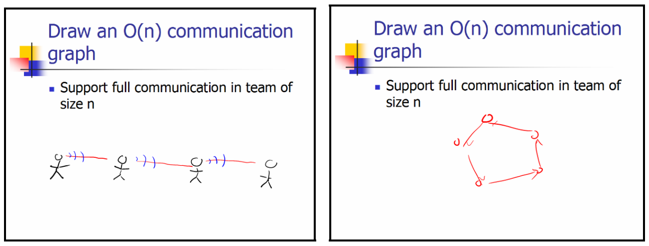

Draw a Huffman tree for the following data values and show internal weights: ~~3~~, ~~5~~, ~~9~~, ~~14~~, ~~16~~, 35

Draw a Huffman tree for the following data values and show internal weights: 3, 5, 9, 14, 16, 35

Figure 6. Correct and incorrect solutions to the Huffman tree activity.

The *ComGraph* activity, shown in Figure 7 and Figure 8, was used in a Software Engineering class during a lecture discussing teams. The students were asked to draw a graph of a team structure that has *O(n)* communication edges and allowed all team members to communicate. The instructor planned to use the results of the activity to make the point that there could be long chains of communication and bottlenecks in communication, but these could be avoided with a hierarchical communication structure. The instructor hoped to find a line graph, a star graph, and a balanced tree in the student submissions to illustrate these individual points. Thus, the instructor had a grouping structure in mind that covered the expected types of graphs. The breakdown of solutions was line (1), ring (5), star (7), tree (4) and other (3). In constructing a sort of the data after class, the instructor placed the line and the rings in the same group, because they could be used to make exactly the same pedagogical point: that there could be long communication distances between people with this structure. Table 1 shows that there was a fairly good agreement with the other sorters on this example – suggesting that the groups were well defined. The main differences in the groupings were that all of the other sorters separated the line from the rings (using the structural property – not the intended use), and the different handling of the other solutions.

Although *ComGraph* was similar to *HuffmanTree*, the data for *ComGraph* would be much more difficult to recognize. This is because it was a brainstorming exercise, where many students chose to augment their diagrams with writing or other art work. This writing served several purposes – in some cases it was used to provide additional information about the solution, and in others it was playful. This type of creative expression can have a positive impact on the class by providing richer examples for discussion and by highlighting individual expression. However, a recognizer would need to be very sophisticated to correctly classify the graphs which used faces for nodes (Figure 8) or had textual explanations (also Figure 8). Another technical challenge in this data is recognizing classes of graphs, and not instances of graphs, so for example, star graphs with different numbers of nodes should go in the same group, and it would also be necessary to recognize graphs with an indeterminate number of nodes.



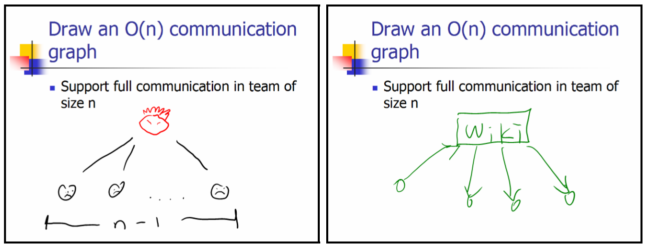
Draw an O(n) communication graph

- Support full communication in team of size n

Draw an O(n) communication graph

- Support full communication in team of size n

Figure 7. A line graph and a ring graph drawn to illustrate communication structures. Note the stick figures used for graph nodes.



Draw an O(n) communication graph

- Support full communication in team of size n

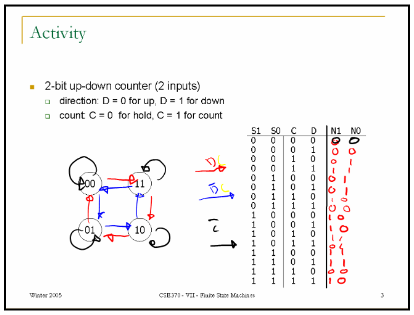
Draw an O(n) communication graph

- Support full communication in team of size n

Figure 8. Two different star graphs which were placed by all sorters in the same group. Challenges to automatically matching these include the different node shapes, the use of text explanation in a node label, and ellipses to indicate missing vertices.

Limitations

We now turn our attention to the examples where automatic grouping is not of interest, or does not appear to be feasible. In our examination of instructor requirements we determined that for 33 out of 36 activities, the instructors believed that grouping would be useful. The three activities where grouping was not thought to be useful were ones where either the instructor did not plan to show the results, or the results would be shown to support an *artifact discussion*. In an artifact discussion examples are shown to the class and discussed in some depth. Often there are many different points to be made from the examples, so they do not fall into any particular grouping structure. Figure 9 shows an example from a Digital Design class where the instructor displayed a variety of student work in order to describe key points that he looked for when grading homework. Although this activity looks similar to many where grouping was useful, it was the instructor's use of the activity that distinguishes it from the others.



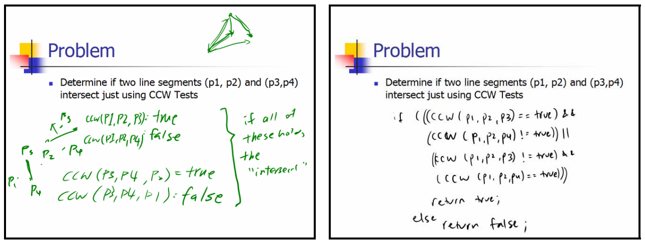
Activity

- 2-bit up-down counter (2 inputs)
  - direction: D = 0 for up, D = 1 for down
  - count: C = 0 for hold, C = 1 for count

S1	S0	C	D	N1	N0
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	0	1
0	0	1	1	1	1
0	1	0	0	0	1
0	1	0	1	0	1
0	1	1	0	1	0
0	1	1	1	0	0
1	0	0	0	1	0
1	0	0	1	1	0
1	0	1	0	1	1
1	0	1	1	0	1
1	1	0	0	1	1
1	1	0	1	1	1
1	1	1	0	0	0
1	1	1	1	1	0

Figure 9. An activity designing state machines where the instructor did not rely on grouping in presenting answers.

There are also cases where the recognition problems are likely to be too hard to expect useful groupings to be computable, even if domain experts are capable of identifying the groups. The examples from our set of activities that we would expect to be the most difficult to sort into groups have a number of features which combine to create a significant challenge: a substantial amount of writing, in a difficult domain, several different modalities used to express answers (text, diagrams, and symbols), and free form two-dimensional expression. It should be noted that these features also make these activities very hard to use in class (even if they could be sorted easily). Figure 10 shows one of the most difficult examples from our set of 87 activities. In class, the instructor just picked one example, and attempted to decipher it during the discussion. Very few activities showed this level of difficulty (maybe 3 others in our set), most likely because instructors recognized that they would be very difficult to use in class.



Problem

- Determine if two line segments (p1, p2) and (p3,p4) intersect just using CCW Tests

ccw(P1,P2,P3) = true
ccw(P1,P2,P4) = false
CCW(P3,P4,P2) = true
CCW(P3,P4,P1) = false
if all of these holds the "intersect"

Problem

- Determine if two line segments (p1, p2) and (p3,p4) intersect just using CCW Tests

```
if  (((ccw(p1, p2, p3) == true) &&
     (ccw(p1, p2, p4) != true)) ||
     ((ccw(p1, p2, p3) != true) &&
     (ccw(p1, p2, p4) == true)))
     return true;
else return false;
```

Figure 10. A geometric activity that was very difficult to analyze. The solution on the left gives the answer with symbols and diagrams, while the example on the right gives the answer in pseudo code.

CONCLUSIONS

This paper was motivated by an information overload problem: how an instructor may be able to integrate student work into discussions while delivering a lecture. This problem arises in the use of a Tablet PC-based classroom interaction system, where students submit digital ink artifacts to the instructor, and the instructor can selectively show some of these artifacts on a public display. We studied the feasibility of automatically grouping student work to support the instructor's use of student submissions in the classroom. The study was conducted by examining the collected student artifacts from activities used in class and by getting detailed feedback on those activities from the original instructors.

The results of our study are generally supportive of using automatic grouping of student submissions, although they do identify challenges that will arise as well. Here is a summary of those results:

- In the vast majority of cases, grouping was important for the instructor, both in planning for an activity and in working with student submissions.
- Student artifacts generally exhibited a grouping structure that could be identified by humans familiar with the domain.

- There is a broad range of challenges associated with the recognition of digital ink in this domain. Some of the strengths that digital ink brings to the classroom also contribute to the difficulty of recognition.
- There is a spectrum of difficulty of automatic grouping for different activities:
    - Groupings for some activities can be computed by using simple heuristics, even though the activities themselves may pertain to a difficult subject area.
    - It may be possible to get fairly good grouping results on text-based activities by building on existing text analysis work.
    - Domain-specific recognizers appear to be necessary for some types of activities, but will require substantial engineering efforts to build.
    - Diagrammatic activities allowing for creativity and individual expression may be very difficult to analyze automatically.
    - There are activities which are likely beyond the scope of automatic analysis, but these are also ones that are difficult to process manually.

Acknowledements
We thank Microsoft Research for support in this collaborative research.

REFERENCES

1. Abowd, G. Classroom 2000: An Experiment with the Instrumentation of a Living Educational Environment. In IBM Systems Journal, 38(4), 1999, pp. 508-530.
2. Alvarado, C., and Davis, R. Resolving Ambiguities to Create a Natural Sketch Based Interface. In IJCAI-2001
3. Alvarado, C. and Davis, R. SketchREAD: A Multi-domain Sketch Recognition Engine. In UIST 2004, pp. 23-32.
4. Anderson, R.J., Anderson, R.E., Hoyer, C., Prince, C., Su, J., Videon, F., and Wolfman, S. A Study of Diagrammatic Ink in Lecture. In Computers and Graphics, Volume 29, Number 4, 2005, pp. 480-489.
5. Anderson, R.J., Anderson R.E., Simon, B., Wolfman, S., VanDeGrift, T., and Yasuhara, K. Experiences with a Tablet PC Based Lecture Presentation System in Computer Science Courses. In SIGCSE 2004, pp. 56-60.
6. Anderson, R.J., Anderson, R.E., VanDeGrift, T., Wolfman, S., and Yasuhara, K. Promoting Interaction in Large Classes with Computer-Mediated Feedback. In CSCL 2003, pp. 119-123.
7. Angelo, T.A. and Cross, K.P. *Classroom Assessment Techniques: A Handbook for College Teachers*. Jossey-Bass Publishers, San Francisco, 1993.
8. Bederson, B. and Shneiderman, B. *The Craft of Information Visualization: Readings and Reflections*. Morgan Kaufmann, 2003.
9. Bligh, D.A. *What's the Use of Lectures?* Jossey-Bass Publishers, San Francisco, 2000.
10. Bransford, J.D., Brown, A.L., and Cocking R.R. (eds.) *How People Learn: Brain, Mind, Experience, and School (Expanded Edition)*. National Academy Press, Washington, D.C., 2000.
11. Deibel, K., Anderson, R.J., and Anderson, R.E. Using Edit Distance to Analyze Cardsorts. In Expert Systems, 22 (3), 2005, pp. 129-138.
12. Dufresne, R., Gerace, W., Leonard, W., Mestre, J., and Wenk, L. Classtalk: A Classroom Communication System for Active Learning. In Journal of Computing in Higher Education, 7, 1996, pp. 3-47.
13. Gross, M. D., and Do, E. Y.-L. Ambiguous Intentions: A Paper-like Interface for Creative Design. In UIST 1996, pp. 183-192.
14. Jancke, G., Grudin, J., and Gupta, A. Presenting to Local and Remote Audiences: Design and Use of the TELEP System. In CHI 2000, pp. 384-391.
15. Johnson, D., Johnson, R., and Smith, K. *Active Learning: Cooperation in the College Classroom*. Interaction Book Company, Minnesota, 1998.
16. Judson, E. and Sawada, D., Learning from Past and Present: Electronic Response Systems in College Lecture Halls. In Journal of Computers in Mathematics and Science Teaching, 21 (2), 2002, pp. 167-181.
17. Kam, M., Wang, J., Iles, A., Tse, E., Chiu, J., Glaser, D., Tarshish, O., and Canny, J. LiveNotes: A System for Cooperative and Augmented Note-Taking in Lectures. In CHI 2005, pp. 531-540.
18. Kara, L.B. and Stahovich, T. Hierarchical Parsing and Recognition of Hand-Sketched Diagrams. In UIST 2004, pp. 13-22.
19. Kaufman, L., Rousseeuw, P.J. Finding Groups in Data: An Introduction to Cluster Analysis. Wiley-Interscience, 1990.
20. Mazur, E. *Peer Instruction: A User's Manual*. Prentice Hall, New Jersey, 1997.
21. Muller, R. and Ottman, T. The "Authoring on the Fly" System for Automated Recording and Replay of (Tele)Presentations. In Multimedia Systems Journal, 8:3, 2000, pp. 158-176.
22. Plamondon, R., and Srihari, S., N. On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey, IEEE PAMI, 22(1), 2000, pp. 63-84.

23. Ratto, M., Shapiro, R.B., Truong, T.M., and Griswold, W.G. The ActiveClass Project: Experiments in Encouraging Classroom Participation. In CSCL 2003.

24. Roschelle, J. and Pea, R. A Walk on the WILD Side: How Wireless Handhelds May Change Computer-Supported Collaborative Learning. In International Journal of Cognition and Technology, 1(1), 2002, pp.145-168.

25. Roschelle, J., Penuel, W.R., and Abrahamson, L.A. The Networked Classroom. In Educational Leadership, 61(5), 2004, pp. 50-54.

26. Simon, B., Anderson, R.E., Hoyer, C., and Su, J. Preliminary Experiences with a Tablet PC Based System to Support Active Learning in Computer Science Courses. In ITiCSE 2004, pp. 213-217.

27. Stanley, C. and Porter, M.E. *Engaging Large Classes*. Akner Publishing Company, Bolton, MA, 2002.

28. Zamir, O. and Etzioni, O. Web Document Clustering: A Feasibility Demonstration. In SIGIR 1998.