GREWPtool: a System for Studying Online Collaborative Learning

Svetlena S. Taneva, Richard Alterman, Kenroy G. Granville, Michael R. Head, Timothy J. Hickey
Computer Science Department, Brandeis University, Waltham MA 02454
Tel: 1-781-736-2729, Fax: 1-781-736-2741
Email: svet@cs.brandeis.edu

Abstract: Collaborative learning continues to be a hot topic in education. There are many advantages in creating assignments, which promote collaboration. Progress towards the creation of a detailed model of student collaboration, that can be used to derive effective methods for scaffolding, has been hampered by the lack of precise data. In this paper, we explore online computer-mediated cooperative work as a method of investigating the process of learning. We have developed an experimental platform, GREWPtool, that supports collaboration amongst non-collocated students as they construct simple websites and programs. GREWPtool records a precise history of their interaction, including all keystrokes. In VCR mode, GREWPtool is able to replay the transcript of a group session at variable speeds (both forward and reverse), as well as to search for specific types of events (e.g. a coding event) and tally relevant statistics. In this paper we describe the tool, expound on the advantages and possible applications of this type of technology for other education studies, and discuss how it is being used in a large-scale study of cooperative programming in pre-novices.

Keywords: collaboration, assessment, analysis, tools

Introduction: Collaborative Learning and Online Collaboration

Peer cooperative learning has recently become a very active domain among researchers studying Computer Science Education (DeClue, 2003; McDowell, Werner, Bullock, & Fernald, 2003; Nagappan et al., 2003). Evidence for the positive effects of cooperative learning is found in several studies of Computer Science learners. However, effects of collaboration in Computer Science education vary depending on the way the collaboration is accomplished – i.e. in a collocated, computer-mediated, synchronous, or asynchronous setting. A recent large-scale study of students in Introductory Computer Science courses demonstrated that those working in pairs performed significantly better on programming projects than students working alone, although their scores on exams were not radically improved (McDowell et al., 2002). Another study found that cooperating students showed considerably greater improvement pre- to post-test, and rated the course higher (Sabin, R. & Sabin, E., 1994). Most interestingly from an educational perspective, Chase and Okie (2000) discovered that introducing peer instruction and cooperative learning to the curriculum of their Introductory Computer Science courses decreased the combined rate of withdrawal and failure from 56% to 33%. The research of online collaboration, however, has revealed somewhat mixed results, and the effectiveness of a groupware system has been shown to be strongly dependent on the context in which it is deployed (Guzdial, Ludovice, Realiff, Morley, & Caroll, 2002).

Studies like these demonstrate that peer learning can be effective, but they do not elucidate the underlying mechanisms. Precise studies of group interaction that would provide an understanding of the mechanisms in successful group learning are methodologically complex. Videotaped sessions capture much of the subtlety, but require painstaking, labor intensive analysis and are, by their nature, non-anonymous (Hay & Pea, 2002). In this paper, we present a suite of tools that both support a wide range of classroom activities and enable a more detailed and less costly ethnographic analysis of student collaboration than the one achieved through video-taping.

From a methodological point of view, the use of groupware to study collaborative learning has many advantages. Chief among them is that all interaction between the students is mediated through the computer, and hence by logging it one maintains a complete record of the interaction – from each person's perspective, and its chronological progression. Another advantage of such an approach is that it allows for a detailed, machine-readable record of students' interaction with the reference materials that they utilized. Finally, the recent rise of instant messaging (IM) communication among students of all ages has opened the possibility of building peer learning groupware systems, which are likely to be well-received by the current generation of students.

Aspiring to develop software that enables the educational researcher to do a more detailed analysis of student collaboration, we built a suite of tools that supports several kinds of classroom activities for a Computer Science course. These include instruction (the teacher can collaboratively write and debug code with students in the classroom), coding (students can write code together online), and TA assistance (students can dock their code and receive help in debugging it from a TA). A key feature of the system is that all online interactions are recorded, and are replayable for further analysis.

In this paper we will demonstrate that access to data at this fine level of granularity can be used to do more detailed kinds of analyses on a wider range of topics, than was previously possible with video data. We provide a case study of how this system is being used in a large-scale study of beginning programmers. The description of our experiment, currently in progress, helps highlight the ways in which it can be used to study online collaborative learning. We will not present experimental results, rather we will present an example of the kind of detailed analysis that is possible by evaluating transcripts of system use.

Background

For several years, we have been exploring the role of collaboration in the classroom. We have used collaboration in programming and writing assignments, as well as in seminars where pairs of students present a technical paper.

There are several benefits to giving students homework that they can work on collaboratively – it allows them to undertake more complicated problems; the work the students do to explain and discuss the material helps them to reach a greater understanding of the material (Roschelle, 1992; Chi, 1989) and the task itself (Perret-Clermont & Brossard, 1985); they explore a larger set of hypotheses (Okada & Simon, 1997). Other advantages accrue from the differences between students – for example, students will frequently have complementary skills. This is especially true in interdisciplinary courses. Even within a homogenous Computer Science class, one student may be better at analysis and another at programming; thus a carefully designed collaborative assignment enables students to learn from each other and see how significant work results from a mix of analysis and programming. Such assignments both replicate work situations in the "real world" and afford the students the opportunity to do more textured projects that require a mix of methodologies to accomplish. The "socializing" that the students do on collaborative projects is yet another important aspect of the collaboration – it directly addresses the alienation and isolation that many Computer Science students feel (Postner, 2003). Further, even from the teacher's point of view, collaborative assignments are advantageous: they enable the instructor to assign more complicated homework assignments for roughly the same grading cost.

Having had a number of positive experiences with assigning collaborative work in the classroom, we chose to develop a technology to explore online collaboration as an approach to education in Computer Science. We had several goals:

1. Develop a suite of collaborative tools for the classroom that would support all aspects of classroom activity, including instruction, collaborative programming, code debugging with TA's, and distance learning.
2. Allow students the ability to review their online work, by replaying any of the sessions in which the system had been used. This feature would make it possible for the student to replay how the teacher constructed code in the classroom or how a TA helped to debug code. The work of Tang (1990) has shown that a representation as a final product is less meaningful and communicative than seeing how that representation is produced.
3. Provide ease of use. One of the classes that we are interested in using the tool for was an introductory course in Computer Science where students write HTML and simple programs; many of the undergraduates in this class are non-scientists. Because relatively unsophisticated users would be amongst our target population, it was very important that the suite of tools be user friendly, utilizing interface concepts with which novices are likely to be familiar.
4. The tools should provide a basis for experimentation.

The last point is the focus of this paper.

There are many open questions in the study of collaboration. Do some kinds of pairing work better than others? For example, is it better to pair a novice with a more expert student or pair two experts? In what way does

the division of labor affect learning? For example, are there advantages to having one student comment another student's code? Are collaborators better at learning the semantics of a programming language than individuals working alone? Are individuals better at learning the syntax of a programming language than pairs? As one can see, there are a large number of issues that need clarification in order to more effectively design collaborative assignments.

One hindrance to the advancement of science in the area of online collaboration is that it is very difficult to obtain precise data. For offline collaboration there is a wealth of techniques for analyzing collaborative interactions (Suchman & Trigg, 1991; Hutchins, 1995). Unfortunately, offline analysis depends on video technology, which has several high costs: it is costly to transcribe parts of the taping, especially the dialogue; movement through the tape is sequential - if the analyst wants to look at particular kinds of activity, he/she has to go through the tape sequentially, tagging interesting material for further analysis; a single tape of an offline interaction is incomplete - it either follows the efforts of a single subject, or the use of a single representation, or a single task within an organization. Multiple tapes may correct the problem of incompleteness, but at the high cost of having to align the different tapes. By and large, the problem is that the data collected from this kind of technology is difficult to work with.

GREWPtool

In one of our projects, we developed a technology to support online collaboration, where the groupware system automatically produces a transcript of use that can be replayed using a replay tool (Landsman & Alterman, 2003). Many of the problems with video technology are alleviated by this approach to collecting data – there is no transcription problem; all of the interactions and tasks are simultaneously captured; the transcript can be viewed sequentially, but it is also possible for the analyst to specify certain kinds of events as stopping points, e.g., the next chat event in a session, or the next coding event in a session, or the next time somebody evaluates the code. We felt that this kind of technology would be especially useful in the analysis of student collaborations.

In this project we are studying the group dynamics among pre-novices (VanLehn, 1989) who are learning to program for the first time (Hickey, Alterman, & Langton, 2002; Langton, Alterman, & Hickey, 2003). As part of this research we built a groupware system, which consists of three parts – (1) a chat window where students communicate using an IM-like interface, (2) a collaborative editor that allows one or more students to simultaneously edit the same document, and (3) a pair of browser windows where students can navigate through help pages and watch the other students' webpage views. All user interaction with the tool is logged and we developed a playback mechanism as part of the system, dubbed the VCR, which allows one to analyze the resulting transcript in great detail. The combined tool is called GREWPtool.

Figure 1 (below) shows a snapshot of the GREWPtool as seen by the students. When a student opens a GREWPtool session he/she must specify a username, which is associated with a unique color (selectable by the user). The color and username form an identification mechanism used throughout the four frames. When a document is created it becomes a "document group" that can be joined by other users. A student can create or connect to multiple groups, switching from one to another via the tabs at the top of the window. To endorse maximum individual customization of screen real estate and to mitigate the issues that arise from a lack of sufficient display space, we chose not to implement a strict WYSIWIS tool (Stefik, Bobrow, Foster, Lanning, & Tatar, 1997). Instead, we encapsulated the major software pieces in elastic windows (Kandogan and Shneiderman, 1997) that allow the students to resize them to their liking, and according to their particular activity needs.

The chat frame on the top of the system screen serves as the main communication resource for the students. GREWPtool's identification mechanism allows for perceptually easy identity recognition on the part of the users – every message is prefixed by the sender's username and highlighted with his/her unique color (Figure 1).

The tool comes equipped with a pair of primitive browsers that display HTML-only webpages. Subjects have access to both public browsers, whose view is shared by all participants, and a private browser. Typically, these browsers are used to access the assignment or review online reference materials.

The collaborative editor, residing in the center, is the heart of the tool. It allows simultaneous editing by everyone in a given document group. A user is represented in this frame by a trailing highlighted cursor (with the user's unique color) that covers the last twenty characters entered by that individual. For the cases where two or more people have intersecting input positions (they are editing the document at an overlapping place), the colors are

merged to form a new highlight. The editor's generic features were kept to a minimum - it loads and saves local files, and allows for keyboard copying and pasting of text. Most importantly, the editor has watch-mode and edit-mode features that are very useful for collaborative editing. In watch-mode a user's view follows the current key inputs of other users, to stay abreast of document changes. Edit-mode allows editing independently of others.

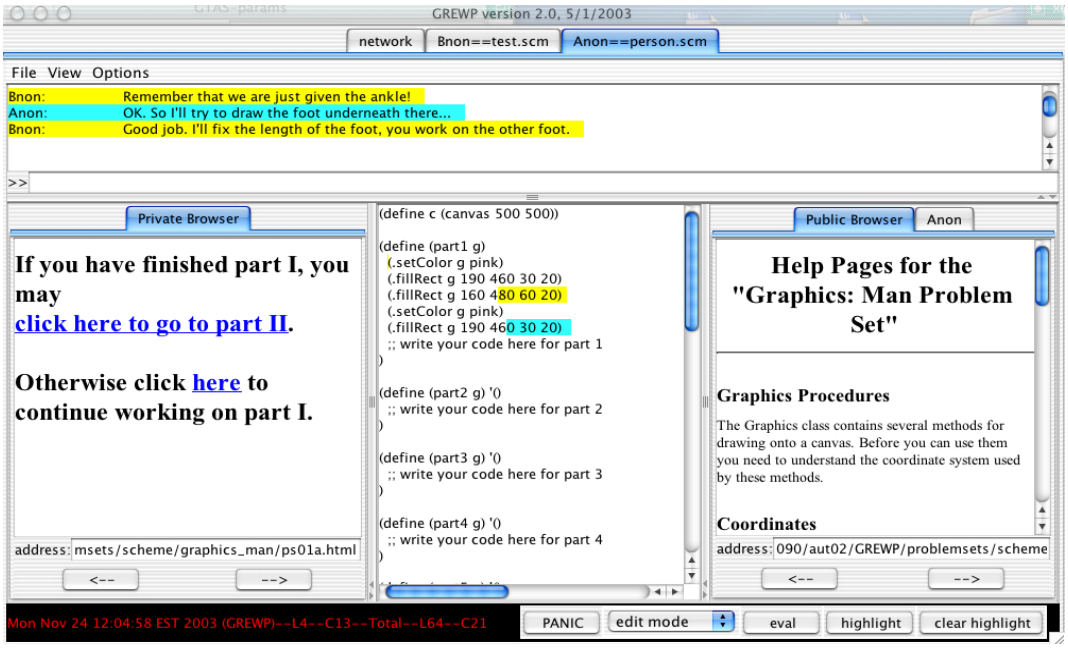

Figure 1. The GREWPtool as seen by students.

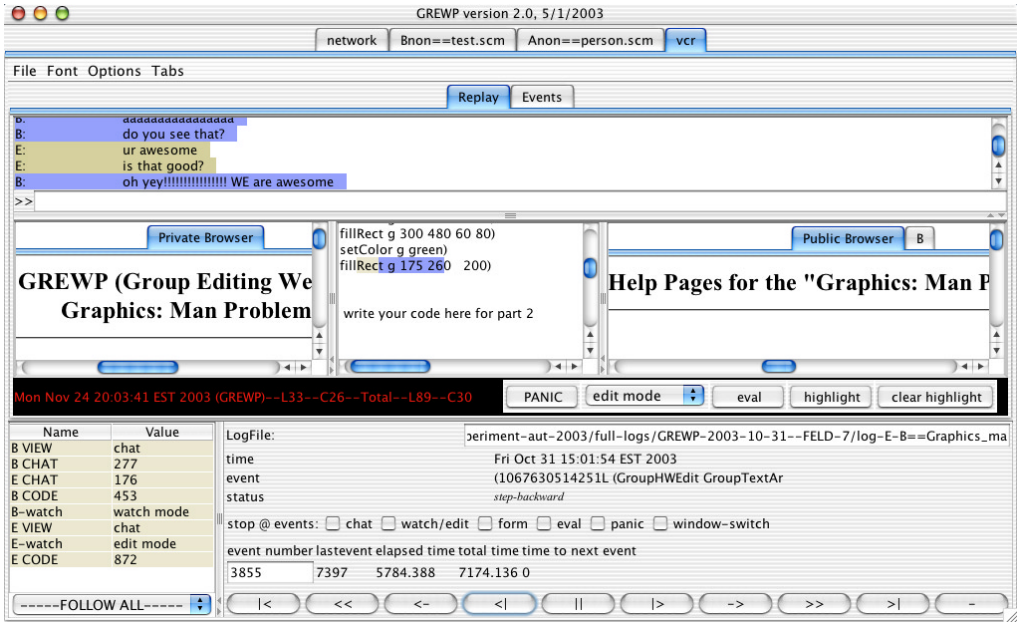

Figure 2. The GREWPtool in VCR mode as seen by analysts.

Figure 2 shows a snapshot of the GREWPtool in VCR mode, as used by the researcher. The analyst can view several log files at the same time, each one selected using the tabs at the top of the window. The VCR screen

view is identical to that seen by the students, with the exception of the VCR control and analysis pane at the bottom of the display. This pane is comprised of several useful widgets – control panel, search selectors, and event analysis pane.

The *control panel* contains the usual VCR controls - pause, single step forward/reverse, actual speed forward/reverse, fast forward/reverse, skip to beginning/end - which allow the analyst to easily browse through the log data. The region immediately above the controls lists the current elapsed time and the current event number. The VCR replays the events that the user saw and/or initiated with millisecond accuracy. As mentioned above, the various users are distinguished by using different colors to identify their chat contributions and edits. The 'color-identity' feature is extremely helpful to the analyst as it makes it perceptually easy to correlate chat actions with edits being made to the document.

The *search specifiers* are located above the control panel as well. These consist of checkboxes indicating the type of events that are of interest to the analyst. When the VCR encounters a selected event, in forward or reverse mode, it stops. The current list of events that can be selected are chat contribution, change to/from watch and edit mode, form (start of a pre/post test or survey), eval, panic, and window-switch.

On the left side of the VCR pane is a table, called the *event analysis*, that provides an instantaneous view of various statistics, including the number of chat messages each user has sent, the number of key strokes that each user has typed (including the delete key), and the current activity of each user (browsing help file, editing, chatting, watching). Additional statistics could easily be added to tailor the analysis to the specific data of interest.

The GREWPtool is disseminated as an open source project (available at http://groupscheme.sf.net). It is implemented in Java and has been tested on a wide variety of platforms including Windows, Linuix, and Mac (OS 10+).

Analyses supported by the GREWPtool VCR

The GREWPtool has been designed to allow the investigator to record and view a wide variety of learning measures at various levels of granularity. Some of this data is recorded and analyzed automatically and provides a quick, at-a-glance analysis – e.g. pre/post test scores, number of key strokes, number of chat messages; other data is automatically recorded but not pre-processed – pre/post surveys, final result of the document editing, actual interaction logs. In the current experiment (described below), we are using all these types of data to examine the fine detail of individual/group learning and group dynamics, as pertaining to the task that the participants solve.

Quantitative Measures

Quantitative measures of learning are provided by the pre/post test and survey data. Our current approach involves a self-perceived confidence-level indicator (from 1="just guessing" to 4="I'm positive its correct") for each answer on the tests, which allows for a more nuanced interpretation of the multiple-choice test results. The surveys were used to asses how students perceived their experience, especially whether it was enjoyable or not, and their level of learning. The logging mechanism records the student's activity as they fill out the survey and take the pre and post test. Replaying this data allows the analyst to watch the subject complete the tests and surveys. This has revealed interesting behavior such as changing answers or confidence levels to questions, and answering questions out of order. Thus, even for the strict quantitative measures, the log file provides a greater level of data completeness.

Qualitative Measures

The log files provide a very detailed record of the interaction. The data they contain is not only the final result of the interaction, but also the progression of the interactive activity from start to finish. In a matter of an instant one can find out how much of the problem was completed and how good the solution is. The logs describe precisely the way students got to the solution, and hence can be mined for much more information. Currently, this "mining" is done manually. However, some of it could be automated easily.

We have used the logs produced by the tool for various ethnographic-like observations – for example, we reviewed the degree to which students in groups engaged in pedagogically rich activities such as explaining concepts to one another, trial and error exploration of the problem space, hypothesis formation, proofreading of their partner's text, etc. In the programming domain, the number and types of errors that students make is a very

interesting measure of their programming ability and a change in these rates provides yet another unobtrusive measure of learning. Being able to observe the kinds of errors learners make, and the way they deal with them, allows the analyst a glimpse into the cognitive processes students are employing during the problem-solving activity.

Applications of GREWPtool

A modified version of the GREWPtool was integrated into the classroom. Students in various Computer Science courses connect from their residences to online class sessions. In addition, TA office hours have been held online through a modified version of the GREWPtool. Once the students and TA's became familiar with the tool, they found it easier and more productive to communicate via the system. The tool provided three primary benefits: the ability for the TA to remotely and synchronously manipulate and discuss the student's code, the ability for the TA to work with several students at once, and the ability for both the student and the TA to work from their respective preferred workplaces.

The GREWPtool has also been used to facilitate cooperation and group problem solving in the classroom (Langton, Granville, Alterman, Hickey, 2003) where it allows students to work collaboratively on written projects (usually web pages or programs). The instructor is able to unobtrusively join the student groups and chat/coedit/cobrowse with individual groups. The instructor can also display any particular group's work on the computer projector for discussion with the entire class. We have not yet begun collecting data on such uses, but have plans to do so in the near future. Our experience suggests that the GREWPtool could also be effective in joint writing assignments, or in facilitating student/tutor interactions for a number of courses where the students create text documents.

The Graph-man experiment

Our current major application of the system is for running a controlled experiments on learning to program. The participants in our study of consist of Brandeis University undergraduate students with backgrounds in various academic disciplines, none of whom had any prior programming experience, i.e. their expertise level was that of pre-novices (VanLehn, 1989). All participants volunteered to take part in the study and were compensated for their time. Students were randomly assigned to work in pairs or individually.

Throughout the Fall 2003 semester, we conducted experiment sessions that took place in one of our computer labs. Upon arrival, students would be briefed about the study and trained on using the tool. Pairs were placed to work on distant workstations (separated by cubicles), so as to prevent direct face-to-face communication during the session. After completing a pre-survey and a ten-minute diagnostic quiz, participants were instructed on the task – to write a basic program that draws an image of a man by actively using the scaffolds provided and following the example images. The task entailed writing code, which draws different parts of a man – feet, legs, torso, arms, head, and facial features. Participants were given ninety minutes to complete the task, after which they had to complete a post-test, which was identical to the pre-test, and a post-survey about their experience using the tool. We reasoned that having the same measure of students' performance before and after completing the task would most clearly indicate the change, if any, in their mastery of the subject matter. The tests were intended to assess three categories of coding knowledge – vocabulary, syntax, and semantics.

The experiment uses a between-subjects design - each group is exposed to one treatment level. The independent variable is cooperation, and the two experimental levels are its presence and absence. This experiment is still in the data collection phase, but we began examining the results of the pre- and post-tests. While almost all students' scores improved, there were some cases, which showed no improvement. Because we have the transcript of the students' entire interaction, we can do a fine-grained analysis and look at exactly what they did during the assignment – specifically the pairs.

A Finer-Grained Analysis

In one particular pair, neither of the students' post-test scores improved over the pre-test scores. We wanted to know why the test showed no learning. Was it because the students didn't learn anything or because of other factors? Since we have access to the full transcript, we can begin to answer this question by asking other finer-grained questions about their interaction, such as – did they collaborate and make progress on the assignment? If they gave up on the assignment and did not attempt to learn, then that would explain why they did not learn.

However, as demonstrated in Figure 3a, the students were not only collaborating, but also were progressing on the assignment. They appear to be enjoying the task and are mutually engaged in it.

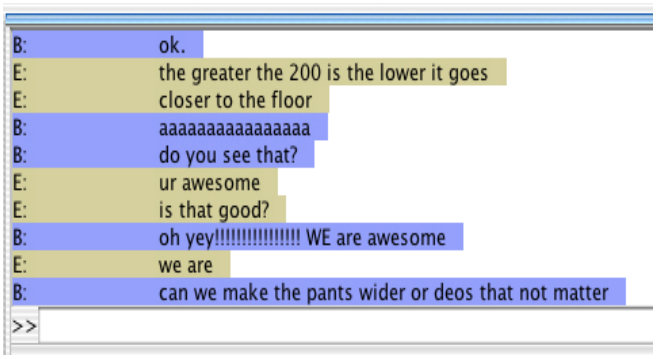

Figure 3a: Students congratulating each other.

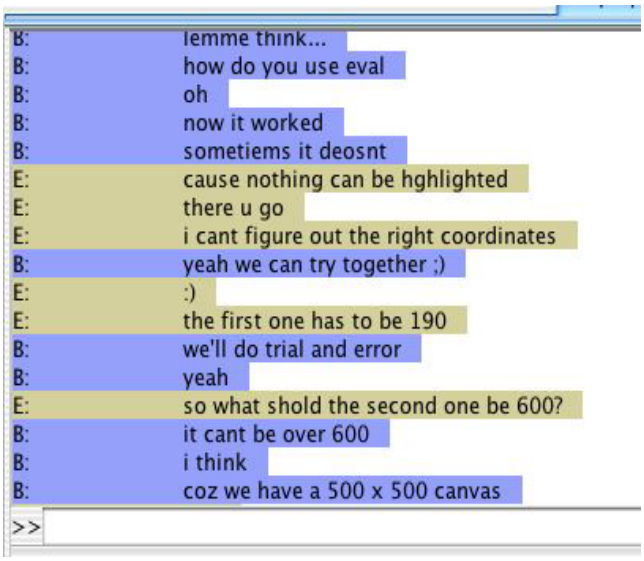

Figure 3b: Trial and error.

So if they were making progress together, why was this not reflected in their post-test scores? As we replayed their session we discovered that they only completed the first two parts of a five-part assignment. The two parts they had completed were in fact done correctly. However, these sections only required them to use a small subset of the concepts that they were tested on. We examined their answers and found that for those questions covering the concepts of the first two parts of the assignment, they both improved. Additionally, by reviewing the chat log shown in Figure 3b, we noticed that the students were still using trial and error to produce change and had not yet internalized the semantics enough at that point for it to reflect in that section of the test.

From a review of this transcript, we determined that this pair spent a larger portion of their allotted time learning how to use the tool than most other students. The questions on the test only ask about concepts in the assignment, not about the usage of the tool itself. This otherwise unavailable form of feedback indicates that we should allocate more tool-specific training time in our experimental protocol.

Concluding Remarks

Because our suite of tools automatically generates transcripts and provides a replay tool, we are able to examine the learning process in greater detail than looking only at final results. With this additional data, we can perform various automatic analyses, as well as selectively do detailed manual analysis on students' transcripts. We can extract any one part of the interaction and see how it correlates with performance on pre and post tests. Rather than asking students how much their partners contributed, we can replay and see exactly everyone's participation level. We can do a detailed manual analysis of the work process of any student or pair of interest. This helps us to understand parts of the learning process that might otherwise go unnoticed without transcripts with this level of granularity. Such understanding can help us improve the entire teaching process: from the software tool to the assignments, to the lectures.

References

Chase, J., & Okie, E. (2000). Combining cooperative learning and peer instruction in introductory computer science. *SIGCSE Bulletin*, 32(1), 372-376.

Chi, M. T. H., Bassok, M., Lewis, M. W., Reimann, P., & Glaser, R. (1989). Self-explanations: How students study and use examples in learning to solve problems. *Cognitive Science*, 13, 145-182.

DeClue, T. (2003). Pair programming and pair trading: Effect on learning and motivation in a CS2 course. *Journal of Computing in Small Colleges*, 18(5), 49-56.

Guzdial, M., Ludovice, P., Realiff, M., Morley, & T., Caroll, K. (2002 October). *When collaboration doesn't work*. Paper presented at ICLS, Seattle, WA.

Hay, K. and Pea, R. (2002 October). *Digital web-based video research: Mapping out the potential of a transformational research technology*. Special session presented at ICLS 2002, Seattle, WA.

Hickey, T., Alterman, R., & Langton, J. (2002). *TA Groupware* (Tech. Rep. CS-02-222). Waltham: Brandeis University, Computer Science Department.

Hutchins, E. (1995). *Cognition in the wild*. Cambridge, MA: MIT Press.

Kandogan, E., & Shneiderman, B., (1997). Elastic Windows: Evaluation of Multi-Window Operations. *Proceedings of SIGCHI: Human Factors in Computing Systems* (pp. 250-257). New York, NY: ACM Press.

Landsman, S. & Alterman, R. (2003). *Analyzing Usage of Groupware* (Tech. Rep. CS-02-230). Waltham: Brandeis University, Computer Science Department.

Langton, J., Alterman, R., T. Hickey (2003). *Integrating Tools and Resources: a case study in building educational groupware for collaborative programming* (Tech. Rep. CS-03-235). Waltham: Brandeis University, Computer Science Department.

Langton, J., Granville, K., Alterman, R., Hickey, T. (2003). *Enhancing CS Programming Lab Courses using Replayable Collaborative Editors* (Tech. Rep., in process), Waltham: Brandeis University, Computer Science Department.

Nagappan, N., L. Williams, M. Ferzli, E. Wiebe, K. Yang, C. Miller, & S. Balik (2003). Improving the CS1 experience with pair programming. *Proceedings of SIGCSE 2003*, (pp. 359-362). New York, NY: ACM Press.

McDowell, C., Werner, L., Bullock, H., & Fernald, J. (2002). The effects of pair-programming on performance in an introductory programming course. *Proceedings of SIGCSE 2003* (pp. 38-42). New York, NY: ACM Press.

Okada, T., & Simon, H.A. (1997). Collaborative discovery in a scientific domain. *Cognitive Science*, 21(2), 109-146.

Perret-Clermont, A. N. & Brossard, A. (1985). On the interdigitation of social and cognitive processes. In R. A. Hinde, A.-N. Perret-Clermont & J. Stevenson-Hinde (Eds.), *Social relationships and cognitive development*. Oxford: Clarendon Press.

Postner, L. (2003). *The challenges of learning to program*. Doctoral consortium session held at the SIGCSE, Northern Kentucky, KY.

Roschelle, J. (1992). Learning by collaborating: Conceptual change. *The Journal of the Learning Sciences*, 2, 235-276.

Sabin, R. E., & Sabin, E. P. (1994). Collaborative learning in an introductory Computer Science course. *SIGCSE Bulletin*, 26(1), 304-308.

Stefik, M., Bobrow, D., Foster, G., Lanning, S., & Tatar, D. (1997). WYSIWIS Revised: Early Experiences with multi-user interfaces. *ACM TOIS*, 5(2), 147-167.

Suchman, L.A., & Trigg, R.H. (1991). Understanding practice: Video as a medium for reflection and design. In J. Greenbaum & M. Kyng (Eds.), *Design at Work: Cooperative Design of Computer Systems*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Tang, J.C. (1991). Findings from observational studies of collaborative work. *International Journal of Man-Machine Studies*, 34(2), 143-160.

Tomasello, M., Kruger, A. C., & Ratner, H. H. (1993). Cultural learning. *Behavioral and Brain Sciences*, 16, 495-552.

VanLehn, K. (1989). Problem Solving and Cognitive Skill Acquisition. In M. I. Posner (Ed.), *Foundations of Cognitive Science*. Cambridge, MA: The MIT Press.

Acknowledgements

This work was supported by the National Science Foundation under Grant No. EIA-0082393.