STUDENT ASSESSMENT OF GROUP LABORATORIES IN A DATA STRUCTURES COURSE

Robert Pastel
Michigan Technological University
E-mail: rpastel@mtu.edu

ABSTRACT

In industry, large software projects require a cooperative work environment. The computer science curriculum has only recently fostered an environment of cooperative work to prepare the students for industry, but most data structures courses continue to rely solely on individual work for instruction and assessing student achievement. This paper studies a series of group laboratories progressing from several versions of programming in pairs to programming in groups of four students to encourage collaborative learning and to introduce students to cooperative work. Student assessment of the laboratories illuminates that the partnership in pair programming naturally organizes itself and is effective at peer instruction. Group laboratories with more than two students provide an environment of cooperative work, and require more laboratory time and structure for the groups to organize.

1. INTRODUCTION

The author/instructor has often considered how a data structures course can prepare its students for cooperative work. Science curriculums, such as physics, chemistry, and biology, traditionally have laboratories which are performed by groups of students. The students naturally assume roles during the laboratory. Group programming laboratories in a data structures course could possibly fulfill the same role, and the students may naturally divide work among themselves. But computer science laboratories have additional constraints that do not exist in the science laboratories. Foremost is the physical constraints imposed by the computer; only one student at a time can type at the keyboard, and it is difficult for more than two students to view the computer monitor simultaneously. The technique of pair programming in the extreme programming methodology exploits this constraint and
establishes material interdependence [3] of the partnership. Assigning the role of driver, controlling the keyboard, to one student and navigator, watching for input errors [5, 7], to the other student ensures partnership interdependence [3]. Beck et al. [2] have generalized the technique of role assignment to larger groups of students by assigning specific roles to students, such as variable manager, program reader, method executor, and facilitator. Although these roles express the multiple aspects of programming, they are not indicative of the roles that software engineers assume in industry. Possibly more beneficial to students would be group laboratories that encourage the students to divide the work into responsibilities that are more similar to industry.

McKinney and Denton [6] have proposed a semester project and LeJeune [4] has proposed multi-week projects conducted by 5-6 student teams combined with pair programming. The structured projects mimic the group structure that students will experience in industry and emphasize program specification and design. They require a large investment in both instructional and laboratory time. Laboratories for a data structures course should emphasize interfaces and algorithms associated with the data structures.

Assiter [1] developed a series of one week course assignments for balancing depth and breadth in a data structures course and encouraged the students to program in pairs. The structured programming experience in closed laboratories can also provide depth. Laboratories that require students to implement interfaces and extend classes expose students to examples of good coding style and encourage students to adopt the style. Closed laboratories also provide the instructor the opportunity to observe students’ difficulties and insights into their learning process.

This paper searches for alternatives to team projects for transitioning students to the cooperative work environment in their upper level projects by studying a series of closed laboratories progressing from programming in pairs to programming in groups of four. As in Assiter [1] and VanDeGrift [9] studies, surveys are used to assess the effectiveness of the laboratories. Student assessment using both likert scale surveys and open-ended questions can validate the instructor’s assumptions about the success of laboratories and students’ participation in the laboratories. In this study, student assessment has provided valuable feedback on ways to improve the laboratories. Students are aware of how they learn, and surveys can provide insights to the students’ learning processes and the pedagogical effectiveness of the laboratories.

2. DATA STRUCTURES COURSE

Approximately 75 students attend our data structures course in one of two sections every semester. Less than 7% of the students are female. The course is required for computer science, computer engineering, and bio-informatics majors. Students generally take the course sometime in their freshman or sophomore year. The students have had two semesters of introductory Java programming. The course goals are the implementation and use of fundamental data structures including cost analysis, and strengthening the students’ programming skills including object-oriented principles. During prior years, students’ work for the data structures course consisted of individually writing 5 programs and passing two exams. Programming assignments early in the course illustrated implementation of simple data structures and later programming assignments emphasized using multiple data structures.
in combination. This year students participated in 4 group laboratories, individually wrote 3 programming assignments, and had two exams.

Throughout the students’ academic career, the university and the department foster an atmosphere of cooperative work and group projects. In their introductory Java course, students program a larger project in pairs. Computer science students are required to take the team software course their junior year, a semester long course in which groups of 4 or 5 students complete a software project of their own design. Several upper level computer science courses rely upon the students’ ability to work and organize into groups, including introduction to software engineering, software quality assurance, human-computer interactions, and senior design project. In prior years the data structures course has not participated or helped prepare the students for the cooperative environment that exist in industry or in their upper level courses. The data structures course had enforced a strict policy of individual work on programming assignments.

3. GROUP LABORATORIES

Group laboratories can provide more than cooperative work experience; they can also facilitate collaborative learning. The peer instruction that naturally occurs in groups can facilitate learning difficult concepts. The four types of group laboratories, described below, attempt to teach multiple software engineering and data structures concepts. The general format is the same for all the laboratories. Each student completes a pre-laboratory consisting of answering a few questions or writing Java or pseudo code. Students were randomly assigned to groups the evening of the closed laboratory, except for the final laboratory when students were assigned to groups two weeks prior to the laboratory. The closed laboratory lasted for two hours. At the conclusion of the laboratory, the group submitted the product of the laboratory, a program or written report, and each student filled out a short survey, which attempted to elicit the students’ thoughts about the laboratory assignment.

The laboratories were developed and conducted during the 2004-2005 academic year. All the laboratory assignments, surveys, and survey summaries are available on line [8].

3.1 Round-robin Laboratories

The first laboratory, the round-robin laboratory, was conducted the second week of the semester. Because the survey results during the Fall semester suggested ways to improve the laboratory a second version was developed and conducted in the Spring. In both versions pairs of students finished the implementation of three interfaces. Successive interfaces extended the prior interface. The initial interface implemented a link list, and the second and third interfaces extended the link list to different applications. The version in the Fall used the link list to represent a polynomial, and the version in the Spring used the link list to implement a stack. After finishing the implementation of an interface, the partnership would separate, with one member of the partnership keeping the code to extend to the next interface with a new partner, and the second member joining with a new partner and extending unfamiliar student written code. All students participated in three partnerships, so that each student would experience working with unfamiliar code.

This is a “get acquainted” laboratory; students become acquainted with other class members, programming to an interface, and using code written by a stranger. Although
students are introduced to link lists in their introductory Java course, the laboratory provides students an opportunity to hone their programming and collaborative learning skills early in the semester.

3.2 Pair Programming Laboratory

The second laboratory, pair programming, was conducted during the fifth week of the semester. Students, with a single partner, extended a linked binary search tree by adding tree traversals.

In prior years, many students have had difficulty with their second programming assignment, parsing and evaluating expressions using tree traversals. Although students are introduced to recursion in their introductory Java course, many students can benefit from the collaborative learning opportunity provided by pair programming to understand the multiple recursions in tree traversals. In this laboratory, partnerships last a sufficient time for students to discover and establish the roles defined by the extreme programming methodology, driver controlling the keyboard, and navigator, watching for input errors and navigator.

3.3 Triplet Code Design Laboratory

The third laboratory, triplet code design, did not require any programming and was conducted approximately the twelfth week of the semester. Several days before the laboratory each student read the laboratory description, which explained the greedy algorithm and wrote a pseudo code solution to a scheduling problem. During the closed laboratory the students discussed their solutions in groups of three, and submitted a final version.

This laboratory transitions the students to working with more than a single partner. The laboratory also provided the students with an opportunity to exchange programming ideas using pseudo code. In addition students formally learn the greedy algorithm, which will be used in several advance algorithms in the final weeks of the course.

3.4 Group Programming Laboratories

The fourth laboratory, programming in groups, was conducted during the thirteenth and fourteenth week of the semester. Two versions of the laboratory were developed. The version for the Fall semester emphasized comparing cost analysis for different sorts and the version for the Spring semester emphasized programming to an interface. In both versions, students were assigned to groups of four two weeks before the closed laboratory. After formation of the groups, each student programmed part of the assignment at home alone. Students in the Fall semester implemented and ran sorts on their own. The groups meet during the closed laboratory to compare the cost of their sorts and write a summary. Students in the Spring semester implemented alone an interface representing a component of a game. During the closed laboratory they combined their code into a larger program, creating a game, which they played.

In both versions of this laboratory, groups were responsible on their own to organize the work within the group. It was hoped that students would gain some experience organizing groups before their team software course. In the sort comparison version, students
could learn together the intricacies involved in applying cost analysis to actual running programs. In the game version students gained experience using interfaces for organizing a large program.

4. LABORATORY ASSESSMENTS

Students assessed the laboratories by filling out an online survey after finishing the laboratory. The survey consisted of approximately 10 multiple choice questions many which were likert scales from 1 to 5, “where 1 is not very much, 3 is some, and 5 is a lot.” Other multiple choice questions elicited preference or agreement with 3 possibilities such as: “no,” “I am not sure,” or “yes.” In this paper, multiple choice responses are reported as a percentage for each possible answer. All students were required to answer all multiple choice questions. Following the multiple choice questions, students could choose to type answers to open-ended questions, including: “How did you manage your group?” “What did you like about this lab?” “What did you dislike about this lab?” “What suggestion do you have for improving the lab?” and “Do you have any other comments?” The students’ responses to the questions were tracked through all surveys. Except for laboratories with two versions, the round-robin laboratory and the group programming laboratory, the results reported in this paper are from the Spring semester.

Not all of the students could participate in the scheduled group laboratories because of scheduling conflict. At the beginning of the semesters, students could choose, for any reason, to do all the laboratories individually. Eighteen percent of the students in the Fall semester and 10% of the students in the Spring semester choose to do the laboratories individually.

4.1 Round-robin Laboratories

The original version of the round-robin, in the Fall, required the students to complete three interfaces each with a different partner in a single two hour session. The first interface, a singly link list, required the students to write 13 methods. The second interface, a list representation of a polynomial, required writing only three methods, but did depend on a correctly implemented link list. The third interface, which compared and added polynomials, only required writing two methods. Ninety two percent of the students responded that they did not have enough time to finish the laboratory. Only 14% of the students responded that they would prefer this lab to a programming assignment, 47% responded they’d prefer a programming assignment and 39% not sure. Students’ responses to the open ended questions included:

“I enjoyed the peer programming experience…Make part 1 a programming assignment, part 2 and 3 a group lab.”

“I liked working with different people…Maybe 2 different lab times to complete [the] whole assignment?”

The responses suggest two approaches to improving the laboratory, and these suggestions were explored in a subsequent survey. Students’ preference for improving the laboratory with a pre-laboratory programming assignment or meeting two nights to complete the lab is 62% and 38%, respectively; the distinction is not very significant, binomial test, \( p = 0.13 \). In order to determine the number of methods for the partnership to implement in
the revised version, students were asked, “How many Java methods do you believe you and your partner could program and debug in an hour?” Figure 1 shows the range of responses.

The revised version of the round-robin laboratory, used in the Spring semester, met two evenings on consecutive weeks. In the first 2 hour meeting, students worked with a single partner to finish a singly link list implementation, which required writing 5 methods. In the first hour of the second meeting, students formed new partnerships and implemented a stack using the link list, which required writing 5 methods. In the second hour of the third partnership tested an infix to prefix expression conversion, which used the stack.

Seventy eight percent of the students in the revised version of the round robin laboratory reported that they had enough time. Fifty six percent of the students reported preference for the laboratory, 15% for a programming assignment, and 29% were undecided. Students’ preference for the revised version of the round-robin laboratory has significantly improved from the first version, \( \chi^2 = 22.4, p < 0.0001 \). The primary purpose of the round-robin laboratory is to increase student awareness of using code written by others. Fifty six percent of the students responded that the lab acquainted them to using someone else’s code, 36% responded the lab acquainted them only somewhat, and 8% responded the lab did not acquaint them, so in general students felt that they become acquainted with others’ code.

Some of the Spring semester students were not introduced to stacks in their introductory Java course, so teaching stacks was a goal for the revised version. Many students reported learning about stacks from the laboratory, (12% reported learning a lot, 32% learning more than some, 31% some, 19% less than some, and 7% not much). More significant is the correlation between students’ responses of knowing stacks before the laboratory and learning about stacks from the laboratory. Figure 2, illustrates the correlation; the Spearman correlation, a nonparametric correlation that relies on only the ranking, is \( \rho = -0.6, S = 54888, p < 0.0001 \).
4.2 Pair Programming Laboratory

The instructor has found in prior years teaching the data structures course that some students have more difficulty understanding and implementing tree traversals than other students. Pair programming is a good technique for peer instruction. Many students reported learning about tree traversals from this laboratory, (12% learning a lot, 37% more than some, 30% some, 17% less than some, and 5% not learning much). Again, the correlation plot for reporting knowing tree traversals before the laboratory verses learning about tree traversals is similar to Figure 2, and the Spearman correlation is $\rho = -0.34$, $S = 48196$, $p = 0.008$. Although the correlation is low the significance is high.

Students that were engaged in the lab enjoyed teaching and learning, $\rho = 0.36$, $S = 22887$, $p = 0.0044$, illustrated in Figure 3. Again although the correlation is low the significance is high.

Another goal for the pair programming assignment was to ascertain how students work in pairs. Fifty seven percent of the students responded that they exclusively typed or checked code, and 40% of the students responded that the partnership traded off typing and checking code. This is consistent with the students’ report of their work in a separate question, (32% typing, 37% checking, and 32% trading off). Although students that only checked code were engaged in the laboratory, students who only typed were more engaged, Spearman correlation $\rho = -0.42$, $S = 51295$, $p = 0.0008$, Figure 4.

Students’ answers to the open ended question, “How did you manage your partnership?” are more explicit about how they work in pairs:

“We worked well as a team, I checked the code and looked up how to do something.”

“We traded off coding and typing as one partner came up with how to implement a method or solve an issue…”

“My partner pretty much took control and held it…. he would keep typing….rather than to try any of my ideas.”

The last comment testifies that not all partnerships were successful. Nevertheless 65% of the students preferred the laboratory to a programming assignment, 10% would have preferred a programming assignment, and 25% were undecided. Because of the preference of to this laboratory to a programming has the highest positive response, this laboratory and
the survey question, “How do you prefer this laboratory to a programming assignment?” is used throughout the rest of the paper as a standard for the other group laboratories.

4.3 Triplet Code Design Laboratory

Students did not spend much time preparing for this laboratory, which was writing pseudo code to share with the group, (45% report spending an hour or less, 36% between 1 and 2 hours, 13% between 2 and 3 hours, and 6% more than 3 hours). Fifty three percent of the students preferred this laboratory over a programming assignment, 28% would have preferred a programming assignment, and 19% were undecided. This preference is significantly down from the preference reported for the pair programming laboratory, $\chi^2 = 5.6, \ p = 0.06$.

4.4 Group Programming Laboratories

A goal of the group programming laboratory in the Fall was for students to experimentally measure and compare asymptotic cost between sorts on different machines. Many students reported learning asymptotic analysis and cost, (6% reported learning a lot, 30% more than some, 46% some, 10% less than some, and 8% not much). Eighty eight percent of the students reported having enough time for this laboratory, 8% almost enough time, and 4% not enough time. Forty two percent of the students reported that they preferred this laboratory to a programming assignment, 28% would have preferred a programming assignment, and 30% were undecided. This is significantly down from students response to the pair programming laboratory, $\chi^2 = 14.8, \ p = 0.02$. By analyzing the responses to open ended questions, we ascertained that most groups organized themselves using email, and writing and running the sorts individually. The group responsibility of this laboratory separated cleanly into individual programming, and the students interacted only to compare their results during the closed laboratory.

The Spring version of the group programming laboratory required a more intricate interaction between group members, because each student coded to an interface that was combined with the game engine to produce a game. Only 43% of the students reported having enough time for this laboratory, 25% almost enough time, and 31% not enough time. Thirty seven percent of the students reported preferring a programming assignment, 33% reported preferring the group laboratory, and 29% were undecided. Students’ preference for the Spring version of the laboratory is less than the pair programming laboratory, $\chi^2 = 14.7, \ p = 0.0006$. Students’ responses to opened ended questions are explicit about the difficulty of this lab and suggesting improvements:

“I liked working with other people. We could not get all the bugs ironed out….it should be…two lab sessions…”

“I liked working in a group and using the interfaces. [I suggest] Make it a two lab project.”

“[I like] That it was a game. It is a very fun concept, and one which I enjoy. [I dislike] … having so little time…”

The students enjoyed that the group laboratory was making a game, but the groups did not have enough time to establish themselves or finish the laboratory. A revised version of
this laboratory could simplify the interfaces and the algorithm for the game engine, and include an additional meeting time for groups to integrate and debug their code.

5. DISCUSSION AND CONCLUSION

The percentages of students that choose to participate in the group laboratories, 80% in the Fall and 90% in the Spring, suggest that our computer science students are eager to program in groups. The single most important criterion for success of the laboratory is for the groups to have enough time to finish the laboratory. Extending the round-robin laboratory to two sessions provided sufficient time for the partnerships to establish themselves, and in less than an hour teach themselves about stacks. In a final survey, 40% of the students in the Fall and 47% in the Spring reported that the pair programming laboratory was their favorite laboratory. The laboratory was focused at teaching only tree traversals. Even though two lectures covered tree traversal before the laboratory, students who felt that they did not know tree traversal after the lectures reported learning tree traversals from the laboratory. The instructor can anecdotically report that unlike prior years students did not have questions about the individual programming assignment requiring parsing and traversing expression trees that followed the laboratory. Partnerships are particularly adroit at peer instruction. Because of the constraints of a single machine, partnerships readily organize themselves to driver and navigator roles either for the duration of the lab or trading off during the laboratory. The instructor believes that these roles are suitable for peer instruction, but there is concern that students who type are more engaged in the laboratory. Instructors may consider enforcing the exchange of roles mid way through the laboratory.

The laboratories with groups larger than two students compared favorably to individual programming assignments, but significantly less than the partnership laboratories. Johnson and Johnson [3] propose that “the rule of thumb about group size is, the smaller the better.” Larger groups require more skilled members and more time to organize themselves. Thirty two percent of the students in the Spring report in the final survey that the group laboratory programming a game was their favorite, while 42% reported that the laboratory was there least favorite. This contradictory result can be explained. Students are highly motivated to program applications that interest them such as games, but the groups did not have enough time to organize themselves. A large group implementing a large project should not solely rely on emails for organization, and the instructor of a lower level course should provide time and structure for the group to organize. An instructor of a data structures course must weigh the benefits of students learning to manage larger groups with the cost in time to organize the groups. Compared with pair programming laboratories, which partnerships quickly organize themselves and are particular adroit at peer instruction, larger group laboratories may not be appropriate for a data structures course.

This paper is only a first study of group laboratories in a data structures course. More studies of group laboratories could find suitable structures for larger groups. A longitudinal study tracking students through the team software course and into the upper level project courses could illustrate if structured group laboratories are successful at providing the skills required to manage group projects.
6. ACKNOWLEDGEMENTS

The author wishes to thank his Chair, Dr. Linda Ott, for providing support for this work, both laboratory space and enthusiasm. The author also thanks the students for their participation in the laboratory and most especially the reviewers for their thoughtful suggestions.


