Effect of Instructional Design and Pair Programming on Student Performance in an Introductory Programming Course

Terence C. Ahern
College of Human Resources and Education,
West Virginia University, Morgantown, WV 26506, terence.ahern@gmail.com

Abstract – California State University Monterey Bay (CSUMB) uses an outcomes-based delivery philosophy and focuses on measurable student learning. Because of the required outcome it was determined that the students needed more scaffolding or “metacognitive support” than is typically afforded in the introductory programming courses. The students were simply not capable of making the jump from simple programming concepts to being able to deliver a satisfactory completed portfolio piece. This paper reports on a study that investigated use of pair programming as a delivery methodology for an introductory Java programming course.

Index terms. Cooperative learning, Extreme Programming, instructional design

INSTRUCTIONAL DESIGN

Historically [1, 2] introductory computer science courses have been delivered using a lecture/demonstration in conjunction with a separate laboratory section. The lecture/demonstration or the direct instruction model is very efficient in conveying factual or procedural information. Over the past few years there has been a tremendous amount of research aimed at improving course delivery that have incorporated elements such as group problem solving and animation [2]. These improvements created better learning experiences resulting in better student retention.

Initially the introduction to Java programming was designed using the lecture/demonstration/assessment fashion expecting a highly motivated CS-type student as the target audience and would be able to quickly understand object oriented programming as it applied to Java.

Half way through this course it was determined that the students needed more scaffolding or “metacognitive support” than is typically afforded in the direct instruction format. Metacognitive scaffolding essentially teaches the student how to learn by strengthening the student’s ability to "select, evaluate, and adjust faulty strategies when they are not working effectively" [3, p. 537]. The goal was to provide scaffolding so that individual students would be supported in understanding the correct strategies in how to design and deploy the required term project. Anecdotal results from the first implementation using the whole class simulation resulted in marked improvement in the final portfolio piece. The resulting portfolio piece was of better quality than the year before but the conceptual understanding of the students was superficial. These mixed results presented a problem, which led to an investigation into a different scaffolding approach.

COOPERATIVE LEARNING

In contrast to the whole class method, cooperative-learning groups “consists of students working together, for one class period or several weeks, to achieve shared learning goals and complete specific tasks and assignments” [4]. The size of the groups can vary from simple pairs to groups of 5 or more. The teacher needs 1) provide pre-instructional coaching, 2) explain the task, 3) monitor group progress and finally 4) assess performance both as a group and individually.

McDowell [5,6] and others have done numerous studies on the efficacy of pair programming for instruction and have confirmed the results that pair programming is an effective lab experience.

EXPERIMENT

CST 336 Internet Programming does not have a formal lab section attached to the course. During the first 6 weeks of the course, lecture/demonstrations were used along with weekly coding to illustrate the concepts. The course uses an objects first approach so concepts such as classes, objects and methods are introduced as early as the first week of the semester.

Initially, 16 students enrolled in CST 336. Before the fourth meeting, six students had dropped the course resulting in 10 remaining students. There were three female students along with seven males. Four of the students in CST 336 would be classified as technical students who would have been exposed to more technical subjects such as Unix programming, C++, network architecture, and so on. In contrast, the remaining six students were design students who may have had a web-based scripting class but were more oriented to visual or graphic design subjects.

After the midterm, the students were instructed in the nature of pair programming. For this experiment, each student was allowed to self-selected the group they wanted to belong too. The class broke into self-selected groups creating a total of four pairs consisting of three groups of two. One group ended up with three members because a student dropped the
course after the midterm. The students formed homogenous pairs aligned along their degree tracks, design or technical. The students who performed the lowest on the midterm were generally the design students.

RESULTS

The results of the midterm indicated a bifurcated distribution based on the student’s categorization of their major. Those students described as being on the more technical track scored well above the mean whereas the students described as being on the design track were typically scored below the mean. This resulted in a much larger variance (see Table 1) as well as a greater standard deviation for the midterm scores.

<table>
<thead>
<tr>
<th>TABLE 1: MIDTERM RESULTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Score</td>
</tr>
<tr>
<td>SD:</td>
</tr>
<tr>
<td>Variance:</td>
</tr>
</tbody>
</table>

The type of questions on the midterm was multiple choice, fill-in, and short answer. The short answer typically referenced a code fragment or the students had to write some simple code to answer the question. The final was constructed from the same question pool. On the final all of the students performed much better than on the midterm. (See Table 3)

<table>
<thead>
<tr>
<th>TABLE 2: FINAL RESULTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Score</td>
</tr>
<tr>
<td>SD</td>
</tr>
<tr>
<td>Variance</td>
</tr>
</tbody>
</table>

The results as shown in Table 3 shows a significant gain (p. = .048) from the midterm scores.

<table>
<thead>
<tr>
<th>TABLE 3: PAIRED STUDENT’S T-TEST</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
</tr>
<tr>
<td>Df</td>
</tr>
<tr>
<td>P</td>
</tr>
</tbody>
</table>

Overall, the class made significant gains in their conceptual understanding. Further, the students who scored poorly on the midterm also showed remarkable individual improvement. Even though the self-selection of the groups led to a homogenous distribution, it did not retard the conceptual improvement in the individual student. On the contrary it can be argued that the pairing provided both emotional and metacognitive support.

Anecdotal data, from observing the groups in action, confirm the psychosocial benefit to the pairs. The groups interacted so that together they would try to understand what the code was doing. There was a real sense that they were in this together. One of the design students remarked that the Thursday programming sessions were fun and that Java was now beginning to make sense.

DISCUSSION

The use of pair programming as a scaffolding delivery methodology was successful in improving the conceptual understanding of the students. However, the implementation of the pair programming was not without cost. In contrast to previous years the quality of the final portfolio piece was less than what had been expected, especially for the design students.

Pair programming is simply another tool in the toolbox. The whole class simulation method maintained the heterogeneous nature of the class. Research has shown that this can result in a rich and more fertile sharing of ideas across groups. According to Swain, Monk and Johnson, “Whole class discussions produced more ideas per class (20) than was the case per pair (9.8) or for individuals working alone (6.9)” [7, p. 392]. Changing from a whole class simulation to the collaborative pair had a severe impact on the ability of the design students to learn from the more technical and experienced programmers within the class. Contrast that with the fact that the homogenous pairs were able to improve on a conceptual basis.

The results of this study demonstrate that combining the direct instruction paradigm with scaffolding can have significant effect on student performance. However, the results also indicate that the type of scaffolding used is not trivial and can have a serve impact on the quality of that performance. This is a difficult question given the constraints of the content, time and intended outcome. Further research is necessary to investigate these issues.

REFERENCES