Increasing Student Retention and Satisfaction in IT Introductory Programming Courses using Active Learning

Keith J. Whittington
Rochester Institute of Technology, Rochester, New York

kjw@it.rit.edu

Abstract

The emerging field of Information Technology is one of several fields that require students to learn computer programming. A large proportion of the students were having difficulty getting through the programming sequence and ultimately changed majors or dropped out of college. To deal with this problem, curricular reforms were implemented and active learning techniques were added to the classroom. The outcome of which was increased student retention, grades, and overall satisfaction. As a result of these encouraging results, an NSF CCLI grant was awarded to formally compare teaching techniques and to create active learning activities specifically designed for introductory computer programming courses. This paper discusses the preliminary work and results that led to the grant award and also summarizes the work that is currently underway.

This material is based upon work supported by the National Science Foundation under Award No. DUE-0442987.

Keywords: Active learning, introductory programming, student retention, curricular reform

Introduction

The Information Technology (IT) Department in the Golisano College of Computing and Information Sciences at Rochester Institute of Technology (RIT) began teaching its introductory Java programming sequence in academic year (AY) 2001-2. Over the past several years, several curricular and instructional modifications have been implemented with the goal of increasing student retention and satisfaction.

The changes included an alternative course sequence that provided more time though the programming sequence and instructional techniques designed to actively engage the students with the goal of deepening student understanding of object-oriented programming concepts.

The following sections describe the evolution of the programming sequence and the work currently underway on the NSF CCLI grant awarded to evaluate and compare traditional lecture instruction with active learning techniques in IT computer programming courses.

Alternative Programming Sequence

The IT department developed a three-course sequence in Java programming in academic year (AY) 2001-2. As the students moved through the sequence, the concepts became increasingly more abstract (encapsulation, polymorphism, inheritance, OOP design, exceptions, and IO) and became more difficult for many of the students to grasp. The second course became the “gatekeeper” course where a significant number of students had difficulty passing the course. To deal with this problem, the existing second course was expanded into an alternative, double-length sequence (Whittington & Bills, 2004).

The decelerated option was implemented in AY 2002-3 and was specifically designed for students who experienced increasing difficulty in grasping the abstract programming concepts presented towards the end of the first course. The hypothesis was that these students needed more time to absorb abstract concepts. These new courses were targeted for students who earned low B’s and C’s in the first course and had a high risk of failing and leaving the program. Students with lower grades had to re-take the first course since it was felt that students needed to become proficient in basic constructs such as decisions and loops before progressing through the sequence.

The alternative two-course sequence covered the same material as the original course, but was extended over two courses. Great care was taken to ensure that these courses were not described as the “slow” sequence, but rather as an opportunity to build strong foundation skills in programming.

As reported by Whittington and Bills (2004), the decelerated sequence was extremely effective and resulted in 14.3% more students receiving a grade of ‘C’ or better through the second step in the introductory programming sequence, decreased the level of intimidation felt from their fellow students by 40%, and 89% of the students who took the alternative sequence had a strong positive opinion of this option.

Active Learning

One more change, and perhaps more significant, was the addition of active learning techniques into the sections taught by the author. Although significant improvements were achieved by slowing down the pace of the courses, a greater impact was seen in the active learning sections over the traditional sections where lectures were the predominant form of disseminating course content. Active learning activities were created to enhance student learning, increase self-confidence, and make a learner-centered classroom. The author felt it was especially critical that these students grasp the fundamental concepts of object-oriented programming because they had begun to struggle with the first programming course as the concepts became more abstract. As course completion rates improved, it became apparent that active learning was a critical component to the success of these courses.

Meyers and Jones (1993) in their seminal book on active learning states that learning is by nature an active endeavor, and different people learn in different ways. Active learning enforces these assumptions through its opportunities for students to talk, listen, read, write, and reflect while using problem-solving exercises, informal small groups, simulations, case studies, role playing, and other activities. It also makes the students apply what they are learning.

A subset of active learning, cooperative learning, was the predominate form used for the active learning activities. These types of exercises use small groups of 3 or 4 to create genuine communities within the classroom and promote deep learning through well structured and orchestrated activities (Millis & Cottell, 1998). Cooperative learning activities were chosen because introductory programming courses are typically taken by freshman students, and it was felt that these students tended to lack the maturity and/or confidence to work collaboratively with other students on complex projects in which each student is assigned a different responsibility.

Some direct research shows that active learning techniques in computer programming are effective in reducing course attrition and improving success. Chase (2000) successfully used two strategies in an introductory computer science course: peer instruction and a cooperative learning environment. These two techniques reduced the overall number of D, F, or Withdrawals from 56% to 33% and showed even more improvement for female students (from 53% to 15%). In a similar study that used peer mentoring as a technique, D, F, or Withdrawals were reduced from 34% to 13% (Stephenson, 1996). Plus recently, Jeffrey McConnell (2005) has provided examples of materials using cooperative learning techniques that he has successfully used in various computer science courses.

Active Course Design

Active learning techniques were intergraded into the traditional classroom in a way that supplemented the traditional teaching methods. The course was designed to promote significant learning by using the following steps for each new topic:

1) Students were given a lecture on a topic
2) A paired-programming exercise was given immediately after the lecture where each pair of students worked on a simple programming exercise with step-by-step instructions. The purpose of this activity was to lead them into making common mistakes then have them analyze the mistakes, modify the code, and ultimately come up with a solution to the activity
3) A programming assignment was given where they had one week to complete it
4) After the above activities were completed, a cooperative learning activity was given that focused on the concepts and reasons for using the current programming constructs.

The purposes of this process were to provide multiple ways of learning and to place an emphasis on higher order learning and less emphasis on mimicry and memorization.

Cooperative Learning Activities

In these activities (Whittington, 2004), the students were divided into groups of 3 or 4 and asked to cooperatively work together on a common solution. Activities included posing questions that asked why particular constructs were used and what purpose they served, developing a code fragment, analyzing code fragments for errors and output, listing the steps required to perform an operation, and acting out a programming assignment where each group was a different object in the program.

Various techniques were used to elicit answers from the students. These methods utilized different group interaction models, such as selecting a best answer, iterative answer refinement, and answer deconstruction and synthesis. Answers were presented by the groups and critiqued by the rest of the students. Instructor led discussions regarding the answers and alternative solutions were also suggested. Another technique brought the students together in groups to develop a code fragment then the code design was discussed and the group presented their answers to the rest of the class. These activities were followed-up with paired programming exercises, described above, which were completed in-class. The weekly programming assignments were presented in a sequential, iterative manner where each project built upon the previous assignment. This allowed students more time to refine their previous solutions that had not worked properly.

Results

The initial two sections of the alternative two-course sequence had one section that used active learning while the other section used traditional teaching methods. These courses covered the same material and used the same tests, and homework assignments. The active learning section had an 8% D, F, W (withdrawal) rate as compared to 28% D, F, W rate in the traditional section. Although there were no further head-to-head comparisons, the D, F, and Withdrawal rates in the active learning sections in successive quarters were 7%, 9%, and 8%. The percentage of A/B grades was also greater for the active learning section (75% to 59%). Additionally, preliminary student satisfaction with the active learning techniques ranged from 71% to 92% approval rate. Although there were different instructors in each section, and it was not a fully realized experiment, the data did suggest that active learning techniques effectively reduce student course attrition and increase student satisfaction.

NSF Grant – Current Work

Based on the initial success of the courses that used active learning, an NSF CCLI grant entitled, Active Learning for Programming in Information Technology, was awarded in 2005.

While the techniques described above have apparently proven successful at RIT though informal course evaluations and anecdotal evidence, this grant provides the opportunity to systematically document and capture these techniques for dissemination, and to gather evaluation data that will help improve the techniques and measure their effectiveness.

The primary goal of this grant is to increase learning and reduce course attrition within introductory computer programming courses through the use of active learning techniques. This grant targets disciplines where programming skills are critical but not the predominant required skill, such as Information Technology. It also supports students who have previously been marginalized in the educational process and who are typically at-risk of leaving these fields, based on their lack of success with traditional instruction. The grant provides alternative instructional methodologies that can enable these students to achieve and succeed.

Evaluation Design and Analysis

The evaluation effort will reflect a quasi-experimental design (Cook & Campbell, 1979), using treatment and control groups of approximately 25 students per group, who are participating in Introduction to Computer Programming courses at Rochester Institute of Technology. Each group will be enrolled in different sections of the same course, but will have the different teachers. The quasi-experiment will be repeated twice, over two quarters. Those students in the treatment group will be taught using active learning techniques while students in the control group will not receive these techniques.

Prior to the collection of the above data, the evaluation team will assist course personnel with assessing the validity and reliability of the instruments used that include pre-tests and post-tests used for major topic portions of the courses. Further, all survey instruments and interview protocols developed by the evaluation team to gather data on usability and satisfaction with the course will undergo similar psychometric analyses prior to field use to establish validity and reliability.

Analysis

Descriptive data about the participants, in the form of entry skills, past successes and failures with programming, and gender will be gathered to help stratify the findings. In addition to general demographic data related to the participants, descriptive statistics will be calculated for both assessment and satisfaction data, and may be used in a formative manner. Inferential methods, (e.g., t-tests, ANOVAs, regression analyses) will be used to identify potential differences between the control and treatment groups. Further, student demographic data will be incorporated in the analyses to investigate the impact of active learning techniques on traditionally underserved populations. Observational data and notes will be shared with faculty for the purpose of refining and improving the techniques in practice.

Problems Addressed

Active and cooperative learning requires significant rethinking of a course and major adjustments for the faculty teaching it (Chase, 2000). One of the major problems that keeps faculty from incorporating active learning into the classroom is that it requires too much time to prepare the activities and the materials and resources are lacking (Mosely & Merritt, 1996). The culture of programming instruction is such that few instructors were ever taught using active techniques, and therefore, they tend to "teach the way I was taught" (Newcomer & Larson, 2001; Zywno, 2003). Instead, programming faculty will need to be explicitly taught how to use these active learning techniques.

Another goal of this work is to enable faculty to easily incorporate active learning exercises into their classrooms by providing specific activities and instructions on how to orchestrate the activities. This will be especially helpful to those who might otherwise be hesitant or unable to create their own active learning activities.

Dissemination

Several publishers will be contacted to publish the workbook of active learning techniques focused on computer programming. Given the current lack of such a document, we believe that it would be attractive to a publisher. Updating such a workbook would become part of a regular publishing cycle of the work. A web page will be created that can be used as a point of contact for interested faculty at other institutions. A series of workshops will also be given to provide instruction on how to implement these techniques in a classroom.

Conclusion

So far the work with active learning in introductory programming courses is promising. Student retention through the programming sequence has increased, and student satisfaction and grades have also improved. If the current work on the grant proves to be successful, it could change the way introductory programming courses are taught. Also, by providing materials and detailed instructions, faculty who have previously hesitated to incorporate active learning into their courses may be encouraged to try it in their classrooms.

References

Chase, J. D. & Okie. E. (2000). Combining cooperative and peer instruction in introductory computer science. *Proceedings of ACM SIGCSE*, Austin, Texas.

Cook, T.D. & Campbell, D.T. (1979). *Quasi-experimentation.* Houghton-Mifflin.

McConnell, J. J. (2005). Active and cooperative learning: Tips and tricks (Part 1). Inroads – *The SIGCSE Bulletin, 37*(2), 27-30.

Meyers, C. & Jones, T. B. (1993). *Promoting active learning: Strategies for the college classroom*. San Francisco, Jossey-Bass.

Millis, B. J. & Cottell, P. G. Jr. (1998). *Cooperative learning for higher education faculty*. Westport, CT, Oryx Press.

Mosely, M. (Ed.). (1996). Using active learning in college classes: A range of options for faculty. *New Directions for Teaching and Learning*. San Francisco, Jossey-Bass.

Newcomer, J. L. & Larson, K. L. (2001). Finding yourself in the classroom: Finding the classroom in your life. Proceedings of the *2001 American Society for Engineering Education Annual Conference & Exposition.*

Stephenson, S.D. (1996). Using a mentor program to reduce course attrition. Paper presented at the *38th Annual Conference of the International Military Testing Association*. Retrieved 5/15/04 from http://www.ijoa.org/imta96/toc.html

Whittington, K. J. (2004). Infusing active learning into introductory programming courses. *The Journal of Computing Sciences in Colleges, 19*(5): 249 – 259

Whittington, K. J., & Bills, D., (2004). Alternative pacing in an introductory Java sequence. *Proceedings of 5th Conference on Information Technology Curriculum*, SIGITE, ACM Press. 118-121

Zywno, M. S. (2003) Engineering faculty teaching styles and attitudes toward student-centered and technology-enabled teaching strategies. *Proceedings of the 2003 American Society for Engineering Education Annual Conference & Exposition.*

Biography

Keith J. Whittington is an Assistant Professor in the Information Technology Department at the Rochester Institute of Technology. His teaching area currently focuses on programming but has also taught in the multimedia, HCI, and networking areas. He spent over 20 years in industry as a computer programmer/systems analyst. His current research interest is teaching programming using active learning techniques, and is currently the PI for an NSF grant entitled “Active Learning for Programming in Information Technology”.