International Journal of Business, Human and Social Sciences
ISSN: 2517-9411
Vol:8, No:5, 2014

A Redesigned Pedagogy in Introductory Programming Reduces Failure and Withdrawal Rates by Half

Said C. Fares, Mary A. Fares

Abstract**—**It is well documented that introductory computer programming courses are difficult and that failure rates are high. The aim of this project was to reduce the high failure and withdrawal rates in learning to program. This paper presents a number of changes in module organization and instructional delivery system in teaching CS1. Daily out of class help sessions and tutoring services were applied, interactive lectures and laboratories, online resources, and timely feedback were introduced. Five years of data of 563 students in 21 sections was collected and analyzed. The primary results show that the failure and withdrawal rates were cut by more than half. Student surveys indicate a positive evaluation of the modified instructional approach, overall satisfaction with the course and consequently, higher success and retention rates.

Keywords**—**Failure Rate, Interactive Learning, Student engagement, CS1.

I. Introduction

PROGRAMMING is a core skill in the computer science field, and therefore most computer science programs start with introductory programming courses. However, regardless of the recognized importance of programming, the results can be disappointing. Deficiencies in basic programming skills were reported by instructors of the upper-level courses. Another consequence of poor learning was high failure rates in introductory programming courses. Many colleges report dropout rates of 20 to 40 percent, even higher, for students in introductory programming courses [1]-[4]. Pedagogical approaches, which take advantage of learning theories and information technologies, have been proposed in the research literature to tackle the learning problems associated with introductory computer programming [5]-[8], [10]-[13]. However, there are very few evidence-based experiences, and the difficulties of learning how to program for novice students remain to be researched [6]. As blended learning becomes more and more pervasive in higher education as the most prominent delivery mechanism, expectations for learning benefits in computer programming are becoming greater. But, simply providing instruction with a mix of face-to-face learning and information technologies will not have the desired effect if the underlying blended learning model does not rely on learning theory and pedagogical principles [9].

To confront this problem, an initiative was developed in 2012 to revise one institution's traditional face-to-face lecturing approach in teaching Computer Science I (CS1). This interactive model was supported by the university Institutional Planning Council in an attempt to improve the overall student satisfaction, retention, and success rates. Five years of data were collected and analyzed. Three years (2009 through 2011) of data represent the previous traditional method of instruction and two years (2012 and 2013) of data represent the current interactive method of instruction. Seven different instructors taught the 21 sections of the programming course from 2009 to 2013. Introductory Java programming is taught to over 60 undergraduate students per semester, where more than 86% of them major in computer science or computer information systems. Course contents, pre-requisites, requirements, and program entrance criteria have remained virtually unchanged.

S. F. is with the Mathematics and Computer Science Department, Valdosta State University, Valdosta, Georgia 31698 USA (email: sfares@valdosta.edu).
M. F. is with the Physics, Astronomy, and Geosciences Department, Valdosta State University, Valdosta, Georgia 31698 USA (email: mfares@valdosta.edu).

II. Description of the Module

Until fall semester 2011, we used a traditional model to teach the introductory programming course. Regular lectures, occasional laboratory meeting, out of class programming projects, limited online resources and limited out of class support. Since spring semester 2012, daily out of class help sessions and tutoring services were available. Interactive lecturing and laboratory have been used, where students participated in writing codes and class discussions. Online resourced and self-paced practice exercises were provided and utilized by the majority of students.

After careful analysis of the traditional teaching method, the new module was developed with the following components described in the following subsections.

A. Help Sessions

Since programming concepts and skills are related, they are often prerequisites or co-requisites. Students who do not understand early in the semester the introductory programming concepts face a greater danger of not grasping the course materials and end up dropping or failing the course. In order to support students in this critical transitional phase, we offered students daily laboratory help sessions and tutoring services. Help sessions offered one on one assistance with programming problems as well as covered common course topics that students expressed difficulty in understanding. The help sessions presented timely support for serious students who faced a "brick wall" in their transitional period. Students' attendance at these voluntarily activities directly correlated to the effectiveness of the early intervention.

B. Interactive Face-to-Face Lectures

Modified regular classroom lectures were introduced to

present class materials, using slides presentations, board, live code developing, debugging, and running sample programs with students 'active involvement and participation.

C. Online Delivery System

Extensive online content was developed to support student learning and housed online in the learning management system (LMS) which was available 24/7. The online components of the module of the course included (1) a course web site, organized by Power Point Slides, assignments, sample codes, test data and executable versions of assigned programming projects, and (2) a large test bank of self-paced exercises with instant and automated feedback to students.

D. Laboratory

The class met in a computer laboratory twice a week. The laboratory was used for instruction, practice, and testing.

Table I highlights the main components and features of the traditional approach compared to the new interactive method of learning.

Individuals and teams shared their findings with the class by presenting solutions or demonstrating their work to the class using a projector. At the end of the period, sample solutions were posted on the course web site for all students to study, evaluate, and compare.

E. Interactive Programming and Practice

After a brief description by the instructor of the lab programming project, students worked individually or in groups to design and implement the solution of the programming task.

F. Quizzes

To encourage student learning, two types of weekly laboratory quizzes were given. The non-programming quizzes were given to encourage students to keep up with class materials while the programming quizzes were intended to give students enough programming practice to enforce learning. During the programming quizzes, students developed programs by using an IDE and submitted their work online by the end of laboratory period. To encourage students to complete the project and understand the concepts, late submissions were allowed but graded out of 50% of the total grade. More than 90% of the students submitted their work on time. Live coding quizzes were also given and allowed open book, online search, and open notes.

G. Progress Monitoring

Combinations of tools were applied to monitor students' performance and identify students-at-risk early enough to intervene. WebCT system tracking data of student activities, class attendance, and students' weekly progress report were used in our intervention. Post-questionnaire results showed that 95% of the students found the weekly performance report helpful.

TABLE I
TRADITIONAL VS. INTERACTIVE METHOD

	Component	Traditional Method	Interactive Method
1	Help Sessions	✔ Occasionally Individual Tutoring	✔ Daily
2	Face-to-face Lectures	✔	✔ With Student Engagement
3	Programming Assignments	✔	✔
4	Testing	✔	✔
5	Quizzes		✔ Weekly
6	Laboratory	✔ Occasionally	✔ Twice a Week
7	Online Resources		✔
8	Self-Paced Practice Exercises		
9	Tutoring Service	✔ Occasionally University Centralized Center	✔ Daily In House Laboratory
10	Progress Monitoring	Midterm grades only	✔ Weekly
11	Student-Instructor Interaction	Limited to class time and office hours	✔ Chatting and emailing

III. EVALUATION

A detailed evaluation of the interactive approach was conducted to study the effectiveness of the new learning environment and its major components. Two survey questionnaires were employed to study student's learning.

A. Pre-Questionnaire

Survey provided to the students after four weeks of the semester. This was designed to collect data related to the activities that were going well, issues that needed to be improved, instructor's feedback, students' overall satisfaction with the course, and students' suggestions for better learning.

B. Post-Questionnaire

An extensive survey was administered to students one week before the end of the semester. Technical evaluation issues addressed the extent to which in class lectures, laboratory, and online resources helped the students to learn programming. Also, pedagogical evaluation issues addressed the extent to which the daily out of class help sessions and tutoring services, interactive learning model provided support to the learning process.

The questions of the survey were grouped into four categories: (1) General student information, (2) Overall student satisfaction with the course, (3) Usefulness of the course resources, and (4) Effectiveness of student –instructor interaction and feedback.

Fig. 1 shows that Eighty-six percent (86%) of the students agreed or strongly agreed that the course advanced their learning to program. Eighty-one percent (81%) agreed or strongly agreed that the course increased their interest in the

computer science field, and they would recommend the course to other students.

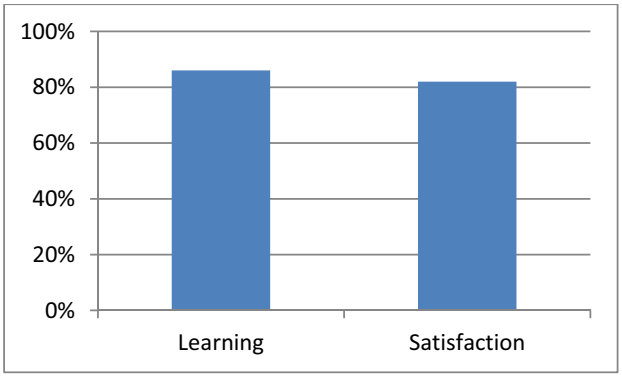

Fig. 1 Student learning and satisfaction

Fig. 2 shows that eighty-file percent (85%) of the students agreed or strongly agreed that the assignments and tests prompt grading and feedback benefited their learning. Ninety-five percent 95% of students agreed or strongly agreed that the updated announcements on Blazeview and the prompt instructor's responses via email kept them informed and connected and helped them understood the class materials by answering their questions.

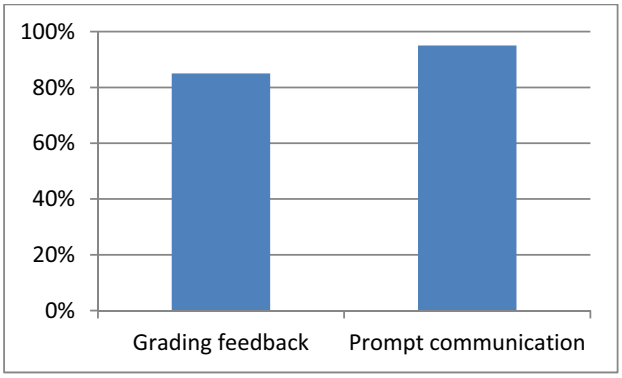

Fig. 2 Prompt grading and communication

Fig. 3 indicates that hundred percent (100%) of the students were satisfied with classroom discussions and live programming examples, eighty percent (80)% of them were satisfied with programming projects, 80% were satisfied with help sessions, but only thirty-six percent (36%) of the students were satisfied with Power Point presentations.

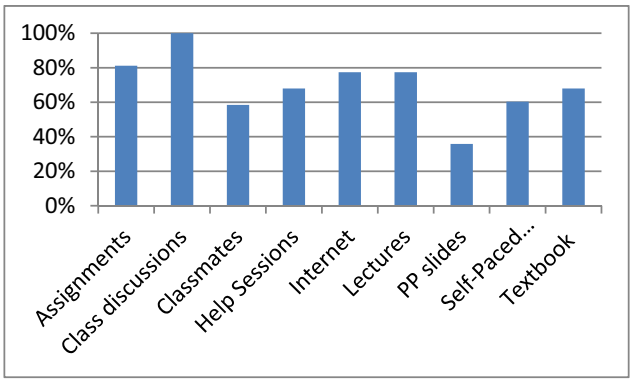

Fig. 3 Usefulness of course resources

Figs. 4 and 5 show that the passing rate of grades (A, B, and C) increased by about 25% using the new interactive method over the traditional teaching method. It is also useful to point out that the course requirements and subjects taught in the course remained virtually unchanged from the old model to the new approach.

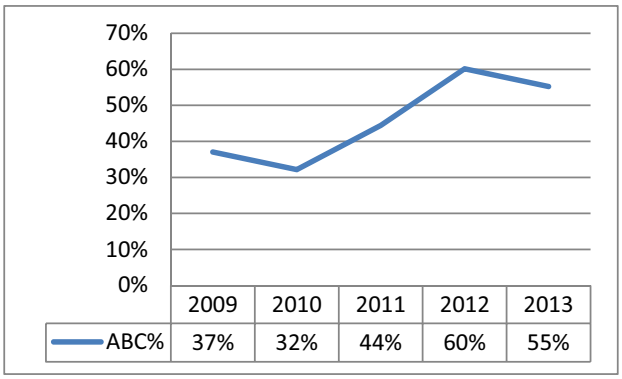

Fig. 4 Percentage of Passing Grades (A, B, and C) from 2009 to 2013

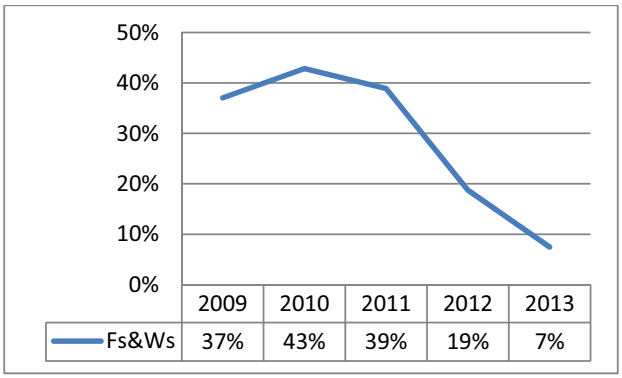

Fig. 5 Percentages of W and F Grades from 2009 to 2013

IV. Conclusions

This paper reported on an interactive learning approach to tackle the problem of a low pass rate in introductory programming course. The results showed significant improvements in decreasing the failure and withdrawal rate by

more than half and increasing the pass rate by about 25%. Evidence from increased student class and help sessions attendance and high utilization of the online resources increased student engagement and subsequently, increased the retention and passing rates.

REFERENCES

[1] Jens Bennedsen, Michael E. Caspersen. "Failure Rates in Introductory Programming". *SIGCSE* Bulletin, Vol. 39, No. 2. 2007.
[2] Bennedsen, J. and Caspersen, M.E. "An Investigation of Potential Success factors for an Introductory Model-Driven Programming Course". *Proceedings of the First International Workshop on Computing Education Research (ICER '05)*. ACM. 2005.
[3] Päivi Kinnunen, Lauri Malmi. "Why Students Drop Out CS1 Course?". *ICER*, Canterbury, UK, 2006.
[4] Rountree, N., Rountree, J., Robins, A. & Hannah, R. "Interacting Factors that Predict Success and Failure in a CS1 Course". *SIGCSE Bulletin*, Vol. 36, No. 4. 2004.
[5] Tom Boyle, Claire Bradley, Peter Chalk, Ray Jones & Poppy Pickard. "Using Blended Learning to Improve Student Success Rates in Learning to Program". *Journal of Education Media*, Vol. 28, 2003.
[6] Said Hadjerrouit. "Towards a Blended Learning Model for Teaching and Learning Computer Programming: A case Study". *Informatics in Education*, Vol. 7, No. 2, 181-210, 2008.
[7] Grandon Gill and Carolyn F. Holton. ADJERROUIT. "A Self-Paced Introductory Programming Course". *Journal of Information Technology Education*. Volume 5, 2006.
[8] Anthony Robins. Learning Edge Momentum: "A New Account of Outcomes in CS1". *Computer Science Education* Vol. 20, No. 1. 2010.
[9] Nocols, M. "A theory of e-learning". *Educational technology and Society*. Vol. 5, No. 2, 2003.
[10] Wilson, B.C. & Shrock, S. "Contributing to Success in an Introductory Computer Science Course: A Study of Twelve Factors". *ACM SIGCSE Bulletin*, Vol. 33, No. 1. 2006.
[11] Woszczynski, A., Haddad, H., & Zgambo, A.: "Towards a Model of Student Success in Programming Courses". *Proceedings of the 43rd Annual Southeast Regional Conference*. Vol. 1. ACM-SE 43), 2005.
[12] Porter, Leo, Simon, Beth. "Retaining Nearly One-Third more Majors with a Trio of Instructional Best Practices in CS1". *SISCSE*' 13, 2013.
[13] Zingaro, Daniel, Bailey Lee, Cynthia, Porter, Leo. "Peer Instruction in Computing: the Role of Reading Quizzes". *SISCSE*' 13, 2013.