Teaching Introductory Programming to IS Students: The Impact of Teaching Approaches on Learning Performance

Xihui Zhang
Department of Computer Science and Information Systems
College of Business
University of North Alabama
Florence, AL 35632, USA
xzhang6@una.edu

Chi Zhang
Department of Information Technology
School of Computing and Software Engineering
Southern Polytechnic State University
Marietta, GA 30060, USA
chizhang@spsu.edu

Thomas F. Stafford
Department of Management Information Systems
Fogelman College of Business and Economics
University of Memphis
Memphis, TN 38152, USA
tstaffor@memphis.edu

Ping Zhang
Department of Mathematical Sciences
College of Basic and Applied Sciences
Middle Tennessee State University
Murfreesboro, TN 37132, USA
pzhang@mtsu.edu

ABSTRACT

Introductory programming courses are typically required for undergraduate students majoring in Information Systems. Instructors use different approaches to teaching this course: some lecturing and assigning programming exercises, others only assigning programming exercises without lectures. This research compares the effects of these two teaching approaches on learning performance by collecting data from two sections of an introductory programming course in an urban public university. One section used lectures and assignments while the other used assignments only. Data analysis included tests within each dataset, tests across the two datasets, and tests of a simple model over the combined dataset. Results indicated that both approaches are effective, but the exercises-only approach is more effective than lectures combined with exercises. Further analysis indicated that students' current programming skills, prior programming experience, and grade expectations are significant antecedents of learning performance in the course. Results support the conclusion suggesting that when teaching introductory programming courses, instructors may want to consider choosing the student-centered active learning over the traditional lecture format in order to improve students' learning performance. This study contributes to the improvement of teaching and learning effectiveness as well as efficiency of introductory programming classes to the benefit of instructors and students, alike.

Keywords: Teaching approach, Learning performance, Active learning, Introductory programming course

1. INTRODUCTION

Undergraduate students majoring in information systems (IS) are typically required to complete an introductory programming course. In general, this course focuses on teaching students one of the several major high-level programming languages, including C, C++, Java, C#, and Visual Basic .NET. Students typically must pass this course with a letter grade of C or higher.

Unfortunately, students often struggle with introductory programming courses. Anecdotal evidence suggests that students taking this course are stressed and afraid of learning the material (Woszczynski et al., 2005a; Woszczynski et al., 2005b). Results from empirical studies indicate that more than one-third of the students taking this course are characterized as "DWF" (earning letter grades of D, withdrawals, or failures), and do not complete the course with the A, B, or C grade required (Beise et al., 2003; Gill and Holton, 2006).

Instructors of such courses use different approaches: some give lectures and assign programming exercises, as well, while others only assign programming exercises without giving lectures (Chou, 2001; Poindexter, 2003). Instructors using the former approach believe that lectures in addition to exercises help students better understand programming concepts and ultimately help improve their programming skills; instructors using the latter approach believe that actively engaged in coding to solve concrete business and computing problems best serves the understanding of programming concepts (Chou, 2001; Gill and Holton, 2006).

A question arises: which approach actually is more effective? At one level, each teaching approach is a matter of personal preference based on belief, but we believe that the most effective approach to teaching introductory programming courses will be indicated by student learning performance, which can be assessed by objective measures. To that end, this study purposely compares the learning outcomes for the two teaching approaches to introductory programming. Specifically, we address three research questions: (1) is teaching introductory programming using exercises only as effective as using exercises combined with lectures, or (2) is one approach more effective than the other? Lastly, (3) what specific factors predict the student learning performance?

Our search of the literature indicates that little research has been done in this area. Hence, empirically determining the answers to these questions is the purpose of this study. We think that determining the effectiveness of programming instruction combined with programming exercises as a teaching approach is critical to informing the effective instruction of introductory programming courses, and that identifying the most effective teaching approach for programming courses effectiveness will lead to increased learning outcomes among IS students.

This paper proceeds as follows: first we present a review of the literature from which we develop hypotheses. Following this, the research methodology is discussed, including details on course background, data collection, and data analysis. Results of hypothesis testing are presented, followed by a discussion of the findings and their implications as well as a discussion of limitations and directions for future research. We conclude the paper with our recommendations and discussion of the issues related to effectiveness in teaching introductory programming.

2. LITERATURE REVIEW AND HYPOTHESES DEVELOPMENT

2.1 Traditional Teaching Approach

The traditional approach to teaching is instructor-led and instructor-centered (Saulnier et al., 2008; Wilson, 1995). This approach suggests that instruction is the primary conduit through which knowledge is delivered in classrooms. Indeed, introductory programming courses are generally taught with lectures, only, or with lectures combined with experiential labs and discussion. In the typical programming classroom, this generally translates into teaching with PowerPoint-based lectures supplemented by audiovisual and other multimedia teaching materials (Schiller, 2009).

When the traditional instructor-led teaching approach is used in programming courses, the instructor generally reviews the content of a chapter, explaining the key terms and concepts followed by the assignment of programming exercises. Students complete the assignments and the instructor provides grades and feedback on their submissions. This traditional teaching approach, used for decades in programming courses, usually produces satisfactory results in terms of student learning performance on tests covering lecture concepts and programming assignments (Saulnier et al., 2008; Wilson, 1995). In this approach, to understanding programming concepts, students use low-level learning strategies to memorize course information and such memorization approaches typically leads to reasonable test performance. The reason students can perform well through the recall of relevant information on tests is because introductory programming courses typically focus on delivering programming concepts and definitions of programming in addition to basic programming language syntax and semantics (Pendergast, 2006). Thus, we hypothesize:

Hypothesis 1: The traditional teaching approach of lectures combined with assignments is effective for an introductory programming course.

2.2 Active Learning Approach

When the traditional instructional approach is used, students are passive recipients of information from the instructor (Prince, 2004); as such, they will often perceive their programming classes as "dry, boring, and tedious" (Lippert and Granger, 1997). In view of this common perception, some educators have begun to experiment with established instructional approaches, redesigning software development courses into more active learning experiences, using exercises and peer learning combined with mini-lectures, where necessary, as compared to the traditional extended lecture approach. Active learning is one of the alternatives to the traditional lecture-based mode of course delivery for

teaching programming courses (Poindexter, 2003). The active approach to learning emerged in the literature in the early 1990s, and involves instructional activities that lead students in "doing things and thinking about what they are doing" (Bonwell and Eison, 1991, p. 1).

The core element of active learning is student engagement through activities related to the course topics (Bakke and Faley, 2007; Schiller, 2009; Williams and Chinn, 2009). In this view, student engagement is critical to learning, whether they do their experiential work individually or in groups. In the active learning environment, students do not simply participate but generally commit to learning and understanding. Hence, learning is enhanced when students become directly engaged in the learning process (Bakke and Faley, 2007; Schiller, 2009; Williams and Chinn, 2009).

In addition to the benefits of direct engagement, student motivation and interpersonal skills are improved in the active learning process (Poindexter, 2003; Prince, 2004; Vernon and Blake, 1993). Both the active exercises and the experiential environment in which they are presented produce benefits for the students. For instance, studies show that active exercises accelerate the learning cycle and improve student problem-solving abilities; related benefits are the reduction of student boredom and the improvement of their course performance (Cordes and Parrish, 1996; Lippert and Granger, 1997; McConnell, 1996; Neufeld and Haggerty, 2001). As contrasted to the instructor-centered approach to teaching, active learning is considered to be learner-centered and has demonstrated substantial improvements in learning outcomes for E-commerce courses (Abrahams and Singh, 2010), database courses (Harris and Vaught, 2008), and MBA-level IS courses (Schiller, 2009). When active learning approaches are used in an introductory programming course, it is expected to improve students' attention, engagement, attitude, motivation, and problem-solving abilities. As such, we hypothesize:

Hypothesis 2: The active learning approach is effective for an introductory programming course.

2.3 Characteristics of Active Learning Approach

Active learning uses problem-based learning. Problem-based learning is an instructional method that presents information on the course topic followed by inviting the students to consider how they might use the information to solve related problems and whether they need to learn more in order to master such problems as well as how they might go about obtaining additional knowledge related to problems at hand (Prince, 2004; Williamson and Chang, 2009). The problem-based approach typically involves significant amounts of self-directed learning on the part of the students, which is likely to influence student attitudes and study habits positively (Prince, 2004).

Vernon and Blake (1993) conducted a meta-analysis spanning 35 studies on the problem-based learning approach, and their results indicated that student attitudes, class attendance, and student moods were consistently more positive for problem-based learning course as compared to course using the traditional instructor-centric approach. Other studies suggest that students will improve the long-term retention of knowledge and develop enhanced critical thinking and problem-solving skills when taught with the problem-based approach (Gallagher, 1997; Major and Palmer, 2001; Norman and Schmidt, 1992).

Active learning promotes student engagement. Active learning introduces experiential activities into the classroom and promotes student engagement in the course (Bakke and Faley, 2007; Schiller, 2009; Williams and Chinn, 2009). Bakke and Faley (2007) found that active learning keeps student interested and engaged while producing high quality learning outcomes. Their results indicate that with active learning students enjoy the classroom experience, have greater control over the learning process, and are able to master more difficult materials. One reason is that active learning is self-directed, hence self-motivated. Motivation research indicates that understanding of content is enhanced when students are committed to knowledge attainment through the use of deep learning strategies such as active learning (Blumenfeld et al., 2006). As such, motivation to learn sets the stage for cognitive engagement. When cognitive engagement is deep, students are able to relate new materials to prior knowledge, which has great benefits over superficial cognitive engagement approaches such as rote memorization (Fredricks et al., 2004).

It is worth noting that motivation alone is not sufficient for ensuring better achievement in the classroom. Cognitive engagement is a catalyst to learning and achievement. Students who value the subject matter and perceive that their needs have been met in the course are more likely to employ deep-level learning strategies (Blumenfeld et al., 2006). Hence, student motivation is enhanced when they have opportunities to decide what and how to analyze, interpret, and apply in the learning process. Such deep and self-directed learning approaches help students make decisions, as well as synthesize, relate, and transform information.

Active learning requires learners to be more responsible. Perkins (1991) identified three demands imposed on learners in active learning: cognitive complexity, task management, and acceptance of the approach. In active learning, learners do not simply memorize the content of lectures and repeat it on assignments and tests. They are responsible for reorganizing and constructing new models based on their existing knowledge structures. These types of tasks are cognitively more complex as compared to the traditional lecture-based approach to learning. The active learning approach considers that learners are responsible for managing their own learning process as opposed to instructors taking responsibility for the learning process. Students involved in active learning approaches have to think more about the concept at hand and the process of mastering the concept. Collectively, this approach may lead to better performance and learning experiences for students than the traditional teaching approach. Thus, we hypothesize:

Hypothesis 3: The active learning approach is more effective than the traditional teaching approach for an introductory programming course.

2.4 Antecedents of Student Learning Performance

Antecedents of student learning performance in programming courses have been studied extensively (e.g., Beise et al., 2003; Chou, 2001; Hasan and Ali, 2004; Simon and Werner, 1996; Szajna and Mackay, 1995). For instance, Beise et al. (2003) examined age, race, and gender as well as

SAT scores as predictors of students' learning performance for computer science and information systems majors. In another study, Hasan and Ali (2004) assessed the effects of computer attitudes, computer experience, and computer self-efficacy on students' learning performance. Other factors that have been studied include training approaches (Chou, 2001; Simon and Werner, 1996) and computer anxiety (Chou, 2001).

In this study, we consider several other important antecedents of student learning performance drawn from prior studies, including students' existing programming skills, prior programming experience, grade expectations, and overall GPA. It is easy to argue that students' current programming skills and prior programming experience will contribute to their learning performance in a programming class (e.g., Hasan and Ali, 2004). It is also not difficult to argue that students who want to achieve a better grade will likely perform better than those who do not, because of the beneficial effects of goal orientation. And, like the predictive role SAT scores as indicated by Beise et al. (2003), grade performance in the form of overall GPA also tends to reflect student capabilities for learning. Thus, we hypothesize:

Hypothesis 4: Students with higher levels of current programming skills will perform better in an introductory programming course.

Hypothesis 5: Students with more programming experience will perform better in an introductory programming course.

Hypothesis 6: Students with higher grade expectation will perform better in an introductory programming course.

Hypothesis 7: Students with higher overall GPA will perform better in an introductory programming course.

Note that H4 hypothesizes the relationship between levels of students' current programming skills and their course performance; the levels of students' current programming skills were measured by their pretest scores. H5 hypothesizes the relationship between students' programming experience and their course performance; the students' programming experience was measured by a self-reported item in a short survey, in which students were requested to choose one from the following five levels of their programming experience: none, some, fair amount, a lot, and expert. We would expect programming skills and programming experience to be inter-related, but to the extent that skills (as measured by knowledge of the topic) can potentially be high without the necessity of having applied experience, we consider skills and experience to be distinct from each other.

3. RESEARCH METHOD

3.1 Course Background

To test the hypotheses we have developed, an introductory programming course in an urban public university in the mid-south region of the United States was used to collect data. The course, titled "Application Program Development," had been offered in the college of business for quite a few years and was well established. Most of the students registered in this course were juniors or seniors majoring in Management Information Systems (MIS). The textbook used for the programming course was *C: How to Program (5th edition)* by Deitel and Deitel (2007). This introductory programming course covered the first 7 chapters of the text, including: (1) Introduction to Computers, the Internet and the Web, (2) Introduction to C Programming, (3) Structured Program Development in C, (4) C Program Control, (5) C Functions, (6) C Arrays, and (7) C Pointers.

The prerequisite for this class was a class titled "Computer Hardware and Systems Software." The programming course was normally offered in two sections per semester, which were typically taught by two different instructors, and was a required course for undergraduate MIS majors. During the data collection semester, as was usual practice, the course was offered in two sections, each taught by a different instructor. One section had 17 students, and the other had 19 students. Each student enrolled into one of the two sections by his/her own choice.

The two instructors shared the same syllabus, used the same textbook, covered the same number of chapters in the same order, assigned the same set of seven programming assignments, and gave identical tests. Both sections met in class twice a week (each for 75 minutes) for programming exercises, and both instructors were available for clarifying concepts and helping with programming assignments in class and outside of class (mostly through emails and visits during the instructor's office hours).

Hence, the courses were identical except for two specific circumstances: one instructor lectured for half of the class time and assigned programming exercises during the other half, whereas the other instructor only assigned programming exercises without giving lectures. The instructor providing the assignment-only curriculum also attempted to create a more relaxed atmosphere by allowing students to talk, debate, and move freely in the classroom when the class was in session. Students in this assignment-only section were encouraged to discuss class-related topics with their fellow classmates, and routinely posed questions to and sought answers from each other.

In the assignment-only section, a more liberal schedule for completion of the assignments was used. Due dates for assignments were less rigid: instead of setting up a specific due date for each assignment, as was the case in the lecture-and-assignments section, assignments were assembled into groups and a more liberal due date was assigned to the combined group of projects. As such, students in the experiential section were permitted to manage the pace of their own learning processes by completing their work in accord with their own needs and priorities. With this scheduling flexibility, students had more control over learning and as a result, were required implicitly to be more responsible for their learning effectiveness and efficiency.

Even though one section was designed for lecture and exercises and the other was an active learning section centered on exercises, only, both sections had access to the PowerPoint lecture slides that were the basis of the lecture-centric section. Upon successful completion of the course, according to the course objectives from both instructors' syllabi, the student should be able to:

- Define common programming terms, operators and conventions.
- Demonstrate the ability to create and run programs using appropriate editing, compiling, and linking tools.

- Understand selecting and using proper data types.
- Identify and correct errors in programming code (debugging).
- Explain the characteristics of sequence, selection, iterative, and modular control structures.
- Implement results of problem solving techniques in a program design.
- Illustrate logically correct programming code (e.g., through pseudocode).
- Create working programming code from pseudocode, UML, etc.

3.2 Data Collection

Data were collected from both sections throughout the semester, including three major parts: a pretest and a posttest, three course examinations, and a short survey. The pretest was given to the students at the first class meeting. It contained 30 multiple choice questions, evenly covering the contents of all the 7 intended chapters of the textbook, and this was the operational assessment of programming skills at the pretest phase. The students were encouraged to give their best efforts to get the highest score they could even though they might have felt unprepared for the material. A posttest using the same set of questions as the pretest was given to the students at the end of the semester.

The three course examinations were administered at different points during the semester. The first exam covered chapters 1, 2, and 3, with the second covering chapters 4 and 5 and the third covering chapters 6 and 7. Each test contained 40 multiple choice questions, and were presented in the same format as the pretest and posttest.

A short survey (aside from the programming skills pretest) was given to the students at the beginning of the semester to assess past programming experience, grade expectation for the course, and overall GPA. This information was used to test whether these factors were significant predictors of student learning performance. Students were asked to provide their name. They were asked about their programming experience, by choosing from experience levels that included "none," "some," "a fair amount," "a lot," and "expert." Students indicated their grade expectation for the course using standard letter grades and indicated their overall GPA by indicating one of five levels: 2.2 or less; 2.3-2.5; 2.6-2.9; 3.0-3.3; and 3.4-4.0.

3.3 Data Analysis and Hypothesis Test Results

Two datasets were obtained, one from each of the two class sections. The first dataset, containing 17 responses, represented the traditional teaching approach. The second dataset, containing 19 samples, represented the active learning approach. Data analysis was conducted within each dataset, across the datasets, and over a dataset of the two combined.

Within each dataset, we conducted one-sample t-tests on the pre and post-tests which covered the course content, in order to determine whether the difference between student knowledge at the beginning of the course and at the end were different. This test was performed within both samples, and the results, as shown in Table 1, indicated that both tests were significant, with each of the p values being less than 0.001. This provided evidence that each of the teaching approaches had a beneficial effect in terms of increasing knowledge across the course of a semester. Thus, both H1 and H2 were supported. The question was which teaching approach was *more* effective.

Section	Test Value = 0			
	t	df	Sig. (2-tailed)	Mean Diff
1	7.094	16	.000	15.353
2	7.465	18	.000	24.789

Table 1. One-Sample T Test Results for Datasets One and Two

In testing across the datasets, we assessed whether there was a significant difference in terms of the two teaching approaches between the two pretest results, between the two posttest results, between the difference score between pretest and posttest per dataset, and between the two overall score results for each dataset. The overall course performance was represented by the sum of scores from the three course exams. The group statistics for pretest, posttest, difference between posttest and pretest, and overall score are shown in Table 2. The independent samples t-test results, as shown in Table 3, indicated that there was no significant difference between the two pretest results ($df = 34$, $t = 0.588$, $p = 0.560$), no significant difference between the two posttest results ($df = 34$, $t = -1.600$, $p = 0.119$), and no significant difference between the two overall score results ($df = 34$, $t = -1.092$, $p = 0.283$). This suggested that on average, students from the two different sections performed at the same level of programming skill at the beginning of the semester as well as at the end of the semester. The difference between the two differences (pretest vs. posttest, per dataset) *was* significant ($df = 34$, $t = -2.320$, $p = 0.026$), as also shown in Table 3. This suggested that the programming-exercises-only approach (i.e., the active learning approach) was *more* effective than the other approach, having produced a significantly greater difference score in programming competency for the experiential section. Thus, H3 was supported.

	Sect.	N	Mean	Std. Deviation	Std. Error Mean
Pretest	1	17	49.24	12.969	3.145
	2	19	46.58	14.009	3.214
Posttest	1	17	64.59	12.238	2.968
	2	19	71.37	13.086	3.002
Diff	1	17	15.35	8.923	2.164
	2	19	24.79	14.474	3.321
OS	1	17	66.37	10.198	2.473
	2	19	70.30	11.259	2.583
Notes: OS = Overall Score					

Table 2. Group Statistics

In the analysis results (see Table 3), it was shown that there is no significant difference in the pretest, posttest, and overall score results. However, the difference between the

improvements across sections (pretest results subtracted from posttest results) was shown to be significantly different. The implications of this outcome require additional consideration.

	t-test for Equality of Means				
	t	df	Sig. (2-tailed	Mean Diff	Std. Error Diff
Pretest	.588	34	.560	2.656	4.517
Posttest	-1.600	34	.119	-6.780	4.238
Diff	-2.320	34	.026	-9.437	4.067
OS	-1.092	34	.283	-3.926	3.596
Notes: OS = Overall Score					

Table 3. Independent Samples T Test Results

Simply put, the difference score reflects changes in programming knowledge between the start and the end of the course. While both sections improved significantly from pretest to posttest, the experiential learning section showed a much larger improvement. Essentially, while both approaches to teaching programming have merit, the experiential approach has significantly greater merit, based on student performance.

We combined both datasets into a single group for additional analysis. Our assessments of pretest, programming experience, expected grade, and overall GPA was used to perform regression tests, which we presume to be legitimated by the sample size exceeding the lower bound for central limit theorem effects. Overall course performance, as in prior analysis, was assessed by the sum of the three course exams. This score was the criterion for regression analysis; predictors were pretest, programming experience, expected grade, and overall GPA.

As shown in Table 4, the results of the regression testing for the predictors explained about 66% of the variance in the students' overall course performance. Of the predictors, pretest scores (p = 0.0009), programming experience (p = 0.051), and expected grade (p = 0.0041) were significant predictors of overall course performance. GPA was not a significant predictor of overall course performance (p = 0.2152). Thus, H4, H5, and H6 were supported, but H7 was not.

Source	DF	Type I SS	Mean Square	F Value	Pr > F
Pretest	1	0.18780078	0.18780078	14.47	0.0009
PE	2	0.14322714	0.03580678	2.76	0.0510
EG	3	0.22357159	0.07452386	5.74	0.0041
OG	3	0.06233811	0.02077937	1.60	0.2152
				R-Square: 0.664562	
Notes: PE = Programming Experience; EG = Expected Grade; OG = Overall GPA					

Table 4. The GLM Procedure Results

4. DISCUSSION

4.1 Implications of Findings

An introductory programming course can be taught in a traditional lecture setting or in an active learning environment. Results suggest either approach will have a beneficial effect on programming knowledge, in general. However, some teaching approaches will produce better learning outcomes than others, and the experiential approach appears to have marked advantages in terms of improvement in programming skills over the course of a semester.

This paper describes a comparative analysis study that investigated the effectiveness of the two teaching approaches on students' learning performance in an introductory programming course. The results indicate that both teaching approaches are effective in improving students' programming knowledge and skills, but they also show that the active learning environment is more effective than the traditional lecture setting. We also demonstrate that current programming skills, prior programming experience, and grade expectations are significant predictors of student learning performance in terms of final grades for a programming course.

These findings have important practical implications. Instructors who are teaching programming courses may consider choosing the active learning approach over the traditional lecture-based approach, or at least integrate active learning components into their classes, since the experiential learning components appear to have superior outcomes for students (Bakke and Faley, 2007; Schiller, 2009; Williams and Chinn, 2009). However, because both instructor-centric and experience-centric teaching approaches have been shown to be effective in improving students' learning performance, instructors who are comfortable with the traditional instructional approach might consider a mixed approach. Instructors can also consider a hybrid approach with lectures for concept explanation and programming demonstration, and active problem-based learning for exercises and assignments.

Students who take programming courses can benefit from the positive effects that active learning can produce. Students engaged in problem-based learning will be more motivated and engaged, and take more personal responsibility for their learning process (Poindexter, 2003; Prince, 2004; Vernon and Blake, 1993). This is certainly a point for both students and instructors to consider.

Our findings that predictive factors such as students' current programming skills, prior programming experience, and grade expectation strongly influence learning outcomes have important implications (e.g., Beise et al., 2003; Hasan and Ali, 2004; Szajna and Mackay, 1995). One approach, for example, might be for instructors in the early days of a course's administration to motivate students with little prior programming knowledge and experience to build intentions to attempt to earn a higher grade. Since grade expectations are shown here to lead to better learning outcomes, regular encouragement to seek better grades may help in improving students' overall learning performance.

Of course, it is important to note that active learning approaches do not automatically result in better learning performance; instructor engagement is required, as well. The structure of the learning environment (e.g., the curriculum and assessment) is a critical factor to the success of active learning approaches (Miller et al., 1996; Poindexter, 2003). Factors of student motivation, engagement, and personal responsibility are essential to the success of active learning

curricula, as we have shown, but aside from the specific set of predictors tested here, some of these factors will vary greatly between individuals.

4.2 Limitations and Future Research

This study has several limitations that need to be addressed in future research. The first limitation regards sample size, which is small but not untenable in the analysis of this experimental design. A compilation of 17 responses in one dataset and 19 in another are adequate for the statistical tests of hypotheses presented here, but future research can certainly seek to investigate similar variables and effects in the setting of larger samples and differing contexts.

A specific limitation to the design and analysis related to the specification and testing of our hypotheses pertaining to learning effectiveness is that we did not index outcomes against teaching effectiveness measures for the two instructors. Given that both samples produced increased learning over the course of the semester, and in view of the fact that both courses were designed around a consistent syllabus, text, and lecture materials (even if one section merely had access to the lectures slides instead of experiencing them in class), we think this may not be a serious problem. However, it is a legitimate regarding any potential differences seen between alternative sections employing differing teaching methods and should be considered in further examinations of the effects we demonstrate here.

A final limitation of this study regards the extent to which our results can be generalized. Our data were produced from convenience samples derived from the student choice of which section to enroll in. There was no practical way in this convenience sample to control sampling in support of investigating potential covariates and mediators of learning processes arising from demographic characteristics such as gender and age, or prior education. Moreover, factors such as time spent on the course, the extent of peer assistance with assignments, and individual learning styles are variables with potential impact on the learning process and should be considered in further applications of the findings we demonstrate here. Longitudinal studies of student learning patterns and outcomes could also serve to better bolster generality of the results we demonstrate here.

5. CONCLUSION

Teaching an introductory programming course to IS students is a challenge for instructors; effective learning in such courses is a challenge to students, as well. The personal preferences of instructors will lead to a variety of teaching approaches applied across courses (Saulnier et al., 2008), of which active engagement is but one approach that might be considered.

This study compared the effects of the two teaching approaches on learning performance – instructor-centric lecture and exercise approaches, as compared to student-centric exercise only approaches. Results indicate that, when executed by competent instructors and compared across sections with consistent syllabi, examinations, and study materials, either approach can produce effective learning outcomes. But, the student-centered active learning approach is shown to have clear advantages, in that it is *more* effective than instructor-centric approaches. When optimal learning outcomes are desired by instructors and administrators of IS departments, active learning approaches provide a means to achieving the best learning performance from among the portfolio of teaching techniques available for delivering introductory classes on programming.

Goode et al. (2007) suggest that educators should leverage the potent influence that student control over and active participation in their learning processes has on outcomes. The active learning approach will be particularly applicable to situations where students lack motivation, engagement, and self-directedness, and can increase student perceptions of self-control and engagement for producing more positive learning outcomes (Law, 2007). Through active and problem-based learning, students become more interested in technical course materials and engage more in learning; this, in turn, improves their learning performance. As we consider the combined roles of instructor encouragement and choice of pedagogical approaches along with beneficial results of student engagement and active participation in programming classes, both students and their teachers can realize important and beneficial outcomes in the classroom experience where the active learning approach is used.

6. REFERENCES

Abrahams, A. S. and Singh, T. (2010). An active, reflective learning cycle for e-commerce classes: Learning about e-commerce by doing and teaching. *Journal of Information Systems Education*, 21(4), 383-390.

Bakke, S. and Faley, R. H. (2007). A student-centric approach to large introductory IS survey courses. *Journal of Information Systems Education*, 18(3), 321-328.

Beise, C., Myers, M., VanBrackle, L., and Chevli-Saroq, N. (2003). An examination of age, race, and sex as predictors of success in the first programming course. *Journal of Informatics Education Research*, 5(1), 51-64.

Blumenfeld, P. C., Kempler, T. M., and Krajcik, J. S. (2006). Motivation and cognitive engagement in learning environment. In R.K. Sawyer (Ed.), *The Cambridge Handbook of the Learning Sciences* (pp. 475-488). New York, NY: Cambridge University Press.

Bonwell, C. C. and Eison, J. A. (1991). Active learning: Creating excitement in the classroom. *ERIC Digests* (ED340272, pp. 1-4). George Washington University, Washington D.C.: ERIC Clearinghouse on Higher Education.

Chou, H. W. (2001). Effects of training method and computer anxiety on learning performance and self-efficacy. *Computers in Human Behavior*, 17(1), 51-69.

Cordes, D. and Parrish, A. (1996). Active learning in technical courses. *Proceedings of the 17th Annual National Educational Computing Conference* (pp. 105-110). Minneapolis, MN, June 11-13.

Deitel, P. and Deitel, H. (2007). *C: How to program* (5th ed.). Upper Saddle River, NJ: Pearson/Prentice Hall.

Fredricks, J. A., Blumenfeld, P. C., and Paris, A. H. (2004). School engagement: Potential of the concept, state of the evidence. *Review of Educational Research*, 74(1), 59-109.

Gallagher, S. (1997). Problem-based learning: Where did it come from, what does it do, and where is it going? *Journal for Education of the Gifted*, 20(4), 332-362.

Gill, T. G. and Holton, C. F. (2006). A self-paced introductory programming course. *Journal of Information Technology Education*, 5, 95-105.

Goode, S., Willis, R. A., Wolf, J. R., and Harris, A. L. (2007). Enhancing IS education with flexible teaching and learning. *Journal of Information Systems Education*, 18(3), 297-302.

Harris, R. B. and Vaught, K. L. (2008). The recovery care and treatment center: A database design and development case. *Journal of Information Systems Education*, 19(3), 277-280.

Hasan, B. and Ali, J. M. H. (2004). An empirical examination of a model of computer learning performance. *Journal of Computer Information Systems*, 44(4), 27-33.

Law, W. K. (2007). Frontiers for learner-centered IS education. *Journal of Information Systems Education*, 18(3), 313-320.

Lippert, S. K. and Granger, M. J. (1997). Peer learning in an introductory programming course. *Proceedings of the 12th International Academy for Information Management Annual Conference* (pp. 123-130). Atlanta, GA, December 12-14.

Major, C. H. and Palmer, B. (2001). Assessing the effectiveness of problem-based learning in Higher Education: Lessons from the literature. *Academic Exchange Quarterly*, 5(1), 4-11.

McConnell, J. J. (1996). Active learning and its use in computer science. *ACM SIGCSE Bulletin*, 28(SI), 52-54.

Miller, J., Groccia, J., and Wilkes, J. (1996). Providing structure: The critical element. In T. Sutherland and C. Bonwell (Eds.), *Using Active Learning in College Classes: A Range of Options for Faculty* (pp. 17-30). San Francisco, CA: Jossey-Bass.

Neufeld, D. and Haggerty, N. (2001). Collaborative team learning in information systems: A pedagogy for developing team skills and high performance. *Journal of Computer Information Systems*, 42(1), 37-43.

Norman, G. R. and Schmidt, H. G. (1992). The psychological basis of problem-based learning: A review of evidence. *Academic Medicine*, 67(9), 557-565.

Pendergast, M. O. (2006). Teaching introductory programming to IS students: Java problems and pitfalls. *Journal of Information Technology Education*, 5, 491-515.

Perkins, D. N. (1991). Technology meets constructivism: Do they make a marriage? *Educational Technology*, 31(5), 18-23.

Poindexter, S. (2003). Assessing active alternatives for teaching programming. *Journal of Information Technology Education*, 2, 257-265.

Prince, M. (2004). Does active learning work? A review of the research. *Journal of Engineering Education*, 93(3), 222-231.

Saulnier, B. M., Landry, J. P., Longenecker Jr., H. E., and Wagner, T. A. (2008). From teaching to learning: Learner-centered teaching and assessment in information systems education. *Journal of Information Systems Education*, 19(2), 169-174.

Schiller, S. Z. (2009). Practicing learner-centered teaching: Pedagogical design and assessment of a Second Life project. *Journal of Information Systems Education*, 20(3), 369-381.

Simon, S. J. and Werner, J. M. (1996). Computer training through behavior modeling, self-paced, and instructional approaches: A field experiment. *Journal of Applied Psychology*, 81(6), 648-659.

Szajna, B. and Mackay, J. M. (1995). Predictors of learning performance in a computer-user training environment: A path-analytic study. *International Journal of Human-Computer Interaction*, 7(2), 167-185.

Vernon, D. and Blake, R. (1993). Does problem-based learning work? A meta-analysis of evaluative research. *Academic Medicine*, 68(7), 550-563.

Williams, J. and Chinn, S. J. (2009). Using Web 2.0 to support the active learning experience. *Journal of Information Systems Education*, 20(2), 165-174.

Williamson, S. and Chang, V. (2009). Enhancing the success of SOTL research: A case study using modified problem-based learning in social work education. *Journal of the Scholarship of Teaching and Learning*, 9(2), 1-9.

Wilson, B. (1995). Metaphors for instruction: Why we talk about learning environments. *Educational Technology*, 35(5), 25-30.

Woszczynski, A. B., Guthrie, T. C., and Shade, S. (2005a). Personality and programming. *Journal of Information Systems Educations*, 16(3), 293-299.

Woszczynski, A. B., Haddad, H. M., and Zgambo, A. F. (2005b). An IS students worst nightmare: Programming courses. *Proceedings of the Southern Association of Information Systems Conferences* (pp. 130-133). Savannah, GA, February 25-26.

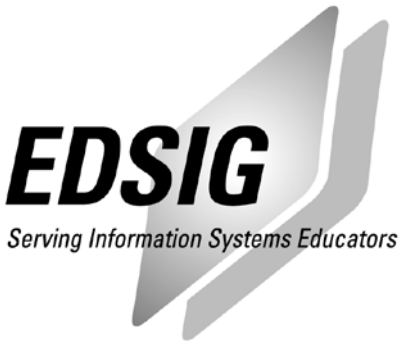
AUTHOR BIOGRAPHIES

Xihui Zhang is an Assistant Professor of Computer Information Systems in the College of Business at the University of North Alabama. He earned a Ph.D. in Business Administration with a concentration in Management Information Systems from the University of Memphis. His teaching and research interests include the human, social, and organizational aspects of Information Systems. His research has appeared in the *Journal of Strategic Information Systems*, *Journal of Database Management*, *Journal of Computer Information Systems*, *Journal of Organizational and End User Computing*, *e-Service Journal*, *Journal of Information Systems Education*, *Journal of Information Technology Management*, *Journal of Information Technology Education*, and other leading journals.

Chi Zhang is an Assistant Professor of Information Technology in the School of Computing and Software Engineering at Southern Polytechnic State University. She received her Ph.D. in Information Technology from the University of Nebraska at Omaha. Her current research involves Health Information Technology and Electronic Health Record Systems adoption & use, best practices of technology-mediated learning, and IT education. She is a member of Association for Information Systems (AIS), Special Interest Group for IS education (AIS-SIGED), Special Interest Group for Information Technology in Healthcare (AIS-SIGhealth), and ACM group for Information Technology Education (ACM-SIGITE).

Thomas F. Stafford is Professor of Management Information Systems for the Fogelman College of Business and Economics at the University of Memphis, and past Editor of *ACM Data Base for Advances in Information Systems*. He holds doctorates in MIS from University of Texas – Arlington, and in Marketing from University of Georgia. His research spans issues of human computer interaction and technology adoption, and has appeared in journals such as *Decision Sciences*, *Communications of the ACM*, and *IEEE Transactions on Engineering Management*.

Ping Zhang is an Associate Professor in the Department of Mathematical Sciences at Middle Tennessee State University. She received a Ph.D. in Mathematics from the University of Memphis, as well as an M.S. in Mathematics from Yangzhou University, China. Her current research interests include developing stochastic and state space models for AIDS and Cancer, Diabete study, Proteomics study, and Informatics Education. Her work has been published in *Education and Information Technologies*, *Journal of Nephrology,* and *Deterministic and Stochastic Models for AIDS Epidemics and HIV Infection with Interventions*. She has presented papers at the ENAR Spring Meeting, AMS, and Joint Statistical Meeting.



STATEMENT OF PEER REVIEW INTEGRITY

All papers published in the Journal of Information Systems Education have undergone rigorous peer review. This includes an initial editor screening and double-blind refereeing by three or more expert referees.

ISSN 1055-3096