8-6-2011

The More You Teach, the Less They Learn: Effects of Teaching Approaches on Learning Performance

Xihui Zhang
University of North Alabama, xzhang6@una.edu

Chi Zhang
Southern Polytechnic State University, chizhang@spsu.edu

The More You Teach, the Less They Learn: Effects of Teaching Approaches on Learning Performance

Xihui Zhang
University of North Alabama
xzhang6@una.edu

Chi Zhang
Southern Polytechnic State University
chizhang@spsu.edu

ABSTRACT

Typically, an introductory programming course is a required course for business college undergraduate students majoring in information systems. Different instructors may have different approaches to teaching this course: Some give lectures and assign programming exercises, while others only assign programming exercises without giving lectures. This research deliberately compared the effects of the two teaching approaches on learning performance. Results indicated that: (1) both approaches are effective, and (2) the programming-exercises-only approach is more effective than the other approach. Further analysis indicated that students' current programming skills, prior programming experience, and grade expectation, are significant antecedents of students' performance in terms of their final grades for the course. These results suggested that when teaching introductory programming courses, instructors may consider choosing the student-centered active learning over the traditional lecture format in order to improve students' learning performance.

Keywords

Teaching approach, learning performance, active learning, introductory programming course.

INTRODUCTION

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

Unfortunately, IS students continue to struggle with this introductory programming course. Anecdotal evidence suggests that students taking this course are stressful and panic-stricken (Woszczynski et al., 2005a; Woszczynski et al., 2005b). Results from empirical studies indicate that more than one-third of the students taking this course end up with DWF (D-grades, withdrawals, and failures), failing to complete the course successfully with an A, B, or C as required (Beise et al., 2003; Gill and Holton, 2006).

On the other hand, at the teaching side, different instructors often use different approaches to teaching this course (Chou, 2001; Poindexter, 2003): Some give lectures and assign programming exercises, while others only assign programming exercises without giving lectures. Instructors using the former approach believe that lectures will help students better understand programming concepts and ultimately help improve their programming skills; whereas instructors using the latter approach believe that the understanding of programming concepts will come to the students once they are actively engaged in coding to solve concrete business and computing problems (Chou, 2001; Gill and Holton, 2006).

A question emerges from the teaching stand point of view: which approach is more effective? Note that in this study, teaching effectiveness is reflected by students' learning performance; students' learning performance is measured by five objective tests. To shed light on the question, this research deliberately compares the effects of the two teaching approaches on learning performance. Specifically, we address three research questions: (1) Are both of the approaches effective? (2) Which approach is more effective? (3) What factors can predict the student's performance?

This paper proceeds as follows. In the next section, we review the literature and develop hypotheses. Then, we present the research method, including course background, data collection, as well as data analysis and results. The implications of the findings, the limitations of the present study, and potential future research are discussed next. Finally, we conclude the paper with our recommendations.

LITERATURE REVIEW AND HYPOTHESES DEVELOPMENT

Traditional Teaching Approach

Traditional teaching approach, which relies mainly on school-based instruction, is primarily an instructor-led and instructor-centered method of teaching (Wilson, 1995). This approach suggests that instruction is the major course delivery method in classrooms. Usually, programming development courses are taught as lecture only or lecture coupled with lab/discussion. Nowadays, the major instruction method is typically the PowerPoint-based lecturing, supplemented by audios, videos, and other multimedia teaching materials.

When the traditional teaching approach is used in an introductory programming course, it often goes as follows. First, the instructor goes through the content of a chapter, explaining the key terms and concepts. Second, the instructor assigns programming exercises. Third, the students complete the programming assignments. Fourth, the instructor gives students feedback on their submissions. The traditional teaching approach, which has been used for decades in programming courses, usually produces satisfactory results in terms of students' learning performance on the tests of the concepts explained in the lectures and feedback to their assignments. In this traditional setting, students use low-level learning strategies to memorize information and do well on tests even if their motivation is simply to pay attention. Thus, we hypothesize:

Hypothesis 1: The traditional teaching approach is effective for an introductory programming course.

Active Learning Approach

With the traditional instructional approach, student passively received information from the instructor (Prince, 2004); as such, they often perceived that programming was "dry, boring, and tedious" (Lippert and Granger, 1997). Given this perception, educators began to make changes in the established instructional process. They redesigned software development course from a traditional lecture setting to a more active learning environment with learning activities such as mini-lectures, active exercises, and peer learning. Active learning is one of the alternatives to the traditional lecture-based mode of course delivery to teaching programming development courses (Poindexter, 2003). Active learning refers to "instructional activities involving students in doing things and thinking about what they are doing" (Bonwell and Eison, 1991, p. 1). As a result, active learning literature emerged as a subject in the early 1990s.

The core elements of active learning are student activity and engagement in the learning process. Indeed, student engagement is critical to their learning. One of the active learning approaches is students working individually or in groups. In the active learning environment, students do not simply participate but are willing to (or have to) invest in learning and understanding. Student learning is enhanced when students pay attention to their own learning because of student engagement and positive attitude. Furthermore, students' motivation is enhanced, and interpersonal skills are gained through active learning (Albanese and Mitchell, 1999; Poindexter, 2003; Prince, 2004; Vernon and Blake, 1993). Both the active exercises and the active learning environment can produce benefits to students. For instance, studies showed that active exercises shorten the learning cycle and improve problem-solving abilities. Studies also showed that the active learning environment reduces boredom and improves performance (Cords and Parrish, 1996; Lippert and Granger, 1997; McConnell, 1996; Neufeld and Haggerty, 2001). When active learning is 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.

Characteristics of Active Learning Approach

Active learning uses problem-based learning. Usually, active learning approach uses problem-based learning. Problem-based learning is an instructional method that presents some information initially and then invites the students to determine what else they need to know and how they can go about learning it (Prince, 2004; Williamson and Chang, 2009). Problem-based learning typically involves significant amounts of self-directed learning on the part of the students. This approach is likely to influence student attitudes and study habits positively (Prince 2004). Vernon and Blake (1993) conducted a meta-analysis of problem-based learning on 35 studies. They concluded that student attitudes, class attendance, and student mood or distress were consistently more positive for problem-based learning than for traditional courses. Studies also suggest that students will improve the long-term retention of knowledge compared to traditional instruction and perhaps develop enhanced critical thinking and problem-solving skills (Albanese and Mitchell, 1993; Gallagher, 1997; Major and Palmer, 2001; Martenses et al., 1985; Norman and Schmidt, 1992).

Active learning promotes student engagement. The core elements of active learning are introducing activities into the traditional lecture and promoting student engagement. There is more self-directed learning involved in this approach. Students need to be more motivated and engaged in the learning process. Motivation research indicates that understanding of content is enhanced when students are committed to building knowledge and employing deeper learning strategies such as

active learning (Blumenfeld et al., 2006). Motivation sets the stage for cognitive engagement. Cognitive engagement can be superficial or deep (Fredrick et al., 2004). Superficial cognitive engagement involves memorizing and elaboration. Deep level cognitive engagement involves the strategies as students try to relate new materials to prior knowledge.

It is worth noting that motivation alone is not sufficient for ensuring better achievement. Cognitive engagement mediates the ways to learning and achievement. Students who value the subject matter and perceive that their needs have been met are more likely to employ deep-level learning strategies (Blumenfeld et al., 2006). Students' motivation is enhanced when they have opportunities to decide what and how to collect, analyze, and interpret information. As students make decisions, synthesize, relate, and transform information, deep-level learning strategies and self-directness are required.

Active learning requires learners to be more responsible. Learners are asked to take more responsibilities in such learning environments. 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 and structures. These types of tasks place a high cognitive load on the learners. Second, learners are responsible to manage their own learning as opposed to the conventional instruction in which teachers take on more responsibilities for task management. Students choosing active learning approaches will have to think more about the concept and the process of learning the concept. Collectively, this approach may lead to better performance and learning experience 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.

Antecedents of Students' Learning performance

Antecedents of students' learning performance in the context of 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 correlations among age, race, and sex as predictors of students' learning performance for computer science and information systems majors. They also included SAT scores in their study. 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 methods or approaches (Chou, 2001; Simon and Werner, 1996) and computer anxiety (Chou, 2001).

In this study, we consider several other important antecedents of students' learning performance inspired by prior studies, including students' current programming skills, prior programming experience, grade expectation, and overall GPA. It is easy to argue that students' current programming skills and prior programming experience will help improve their learning performance (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 the former has a goal which will motivate them to work harder to achieve that goal. Typically, overall GPA, like SAT included in Beise et al. (2003), reflects a student's IQ and effort for his/her learning performance. Thus, we hypothesize:

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

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

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

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

RESEARCH METHOD

Course Background

To address the research questions and test the hypotheses outlined and proposed in previous sections, we chose an introductory programming course in a public, urban university in the mid-south region of the United States to collect data for this empirical study. The course was titled "Application Program Development," and was offered by the Department of Management Information Systems in the school's business college. This course was well established. In a single semester, the course was normally offered in two sections, which were typically taught by two different instructors, respectively. It was a required course for undergraduate students who were majoring in MIS. Most of the students registered in this course were MIS juniors or seniors. The prerequisite of this course was another MIS course entitled "Computer Hardware and Systems Software." The textbook used for this course was "C: How to Program (6th edition)" by Deitel and Deitel (2009). As an

introductory programming course, it only covered the first 7 chapters of the textbook, 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.

During the data collection semester, this course was offered in two sections. One section was taught by a junior instructor and the other section was taught by a senior instructor. The two instructors shared the same syllabus, used the same textbook, covered the same number of chapters, assigned the same set of programming exercises, and gave the identical tests. The junior instructor gave lectures and assigned programming exercises, whereas the senior instructor only assigned programming exercises without giving lectures. Note that the PowerPoint slides were made available to students in both sections. The junior instructor's section had 17 students, and the senior instructor's section had 19 students. Each student enrolled into one the two sections by his/her own choice.

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.

Data Collection

Data were collected from both sections throughout the semester, including three major parts: a pretest and a posttest; three other tests; and a short survey. The pretest was given to the students at the first day of the class. It contained 30 multiple choice questions, evenly covering the contents of all the 7 intended chapters of the textbook. The students were encouraged to give their best efforts to get the highest score they could even though they might have no idea of some of the questions. A posttest, using the same set of questions as the pretest, was given to the students at the end of the semester. Scores from both pretest and posttest were recorded.

Three other tests were given throughout the semester, covering different chapters. Test 1 covered chapters 1, 2, and 3; Test 2 covered chapters 4 and 5; and Test 3 covered chapters 6 and 7. Each test contained 40 multiple choice questions. Scores from all three tests were recorded. Note that we used multiple choice questions in these three tests as well as in the pretest and posttest to make sure that all the scores were graded as objectively as possible.

A short survey was also given to the students at the beginning of the semester to collect students' past programming experience, grade expectation for the course, and overall GPA. This information was used to test whether these factors are significant predictors of the student's performance. Specifically, four questions were included in this short survey. Question 1 was about the student's name. Question 2 was about the student's programming experience; they were asked to choose one from the following five levels: none, some, fair amount, a lot, and expert. Question 3 was about a student's grade expectation for the course, including A, B, C, D, and F in letter grade format. Question 4 was about the students' self-reported overall GPA; they were asked to choose one from the following five levels: 2.2 or less; 2.3-2.5; 2.6-2.9; 3.0-3.3; and 3.4-4.0. All the students' responses to the survey were recorded.

Data Analysis and Hypothesis Test Results

Two data sets were obtained, each from one of the two class sections. The first data set, containing 17 samples, was collected from the junior instructor's section (the traditional teaching approach section). The second data set, containing 19 samples, was collected from the senior instructor's section (the active learning approach section). Data analysis included tests within each data set, tests across the two data sets, and tests of a simple model over the combined data set. Within each data set, we ran a one-sample t test with SPSS 17.0 to test whether the difference between the pretest results and posttest results was significant. The one-sample t test results, as shown in Tables 1 and 2, indicated that both were significant, with each of the p values being less than 0.001. This suggested that both teaching approaches were effective. Thus, both H1 and H2 were supported.

	Test Value = 0					
					95% Confidence Interval of the Difference	
	t	df	Sig. (2-tailed)	Mean Difference	Lower	Upper
Diff	7.094	16	.000	15.353	10.77	19.94

Table 1. One-Sample Test Results for Data Set One

	Test Value = 0					
					95% Confidence Interval of the Difference	
	t	df	Sig. (2-tailed)	Mean Difference	Lower	Upper
Diff	7.465	18	.000	24.789	17.81	31.77

Table 2. One-Sample T Test Results for Data Set Two

Across the two data sets, we tested whether there was a significant difference (1) between the two pretest results, (2) between the two posttest results, (3) between the two differences (i.e., the difference between the two results of [posttest - pretest]), and (4) between the two overall score results. Note that the overall course performance was represented by the overall score which was the sum of scores from the three tests other than the pretest and the posttest. The group statistics for pretest, posttest, difference between posttest and pretest, and overall score are shown in Table 3. The independent samples t test results, as shown in Table 4, 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 students from the two sections were at the same level with their C programming skills at the beginning of the semester as well as at the end of the semester. The difference between the two differences, however, was significant (df = 34, t = -2.320, p = 0.026), as also shown in Table 4. This suggested that the programming-exercises-only approach (i.e., the active learning approach) was more effective than the other approach. Thus, H3 was supported.

	Section	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
OverallScore	1	17	66.37	10.198	2.473
	2	19	70.30	11.259	2.583

Table 3. Group Statistics

		Levene's Test for Equality of Variances		t-test for Equality of Means						
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	95% Confidence Interval of the Difference	
									Lower	Upper
Pretest	Equal variances assumed	.015	.903	.588	34	.560	2.656	4.517	-6.523	11.836
	Equal variances not assumed			.591	33.953	.559	2.656	4.497	-6.483	11.796
Posttest	Equal variances assumed	.417	.523	-1.600	34	.119	-6.780	4.238	-15.392	1.832
	Equal variances not assumed			-1.606	33.924	.118	-6.780	4.222	-15.360	1.800
Diff	Equal variances assumed	3.605	.066	-2.320	34	.026	-9.437	4.067	-17.701	-1.172
	Equal variances not assumed			-2.381	30.373	.024	-9.437	3.964	-17.527	-1.346
OverallScore	Equal variances assumed	.010	.921	-1.092	34	.283	-3.926	3.596	-11.235	3.383
	Equal variances not assumed			-1.098	33.992	.280	-3.926	3.576	-11.194	3.342

Table 4. Independent Samples T Test Results

We built a general linear model (GLM) of pretest, programming experience, expected grade, and overall GPA on the overall course performance, combining both data sets into one (n = 36). Note again that the overall course performance was represented by the overall score which was the sum of scores from the three tests other than the pretest and the posttest. The GLM procedure, which uses the method of least squares to fit general linear models, is used to estimate this model. We assumed the model was: OverallScore = a_0 + a_1 * Pretest + a_2 * ProgrammingExperience + a_3 * ExpectedGrade + a_4 * OverallGPA, where a_0, a_1, a_2, a_3, and a_4 were parameters needed to be estimated.

The results of the GLM procedure, as shown in Table 5, indicated that pretest (p = 0.0009), programming experience (p = 0.0510; almost significant), and expected grade (p = 0.0041) were significant indicators to the students' overall course performance. The overall GPA, however, was not a significant indicator to students' overall course performance. Together, the four independent variables (i.e., pretest, programming experience, expected grade, and overall GPA) could explain about 67% of the variance of the dependent variable (i.e., students' overall course performance), as indicated by R^2 = 66.4562% in Table 5. Thus, H4, H5, and H6 were supported, but H7 was not supported.

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

Table 5. The GLM Procedure Results

DISCUSSION

Implications of Findings

An introductory programming course can be taught in a traditional lecture setting or in an active learning environment. This paper describes a comparison study that investigated the effectiveness of the two teaching approaches on students' learning performance in an introductory programming course. The results indicate that (1) both teaching approaches are effective in improving students' programming knowledge and skills; (2) the active learning environment is more effective than the traditional lecture setting; and (3) students' current programming skills, prior programming experience, and grade expectation are significant predictors of their learning performance in terms of final grades for the course.

These findings have important practical implications to instructors as well as to students. To instructors who are teaching programming development courses, they may consider to choose the active learning approach over the traditional lecture-based approach because the former approach is more effective than the latter approach. However, because both teaching approaches have been shown to be effective in improving students' learning performance, instructors who are comfortable with the traditional instructional approach do not have to be under great pressure to switch to the active learning approach in a hurry. The transition process can take its time, with detailed plan to ensure its success.

To students who are taking programming development courses, they should be aware of the positive effects that active learning can produce, regardless of what teaching approach their instructors are using. This means that students should use problem-based learning, be more motivated and engaged, and take more responsibilities in their learning. This certainly needs more of the instructor's encouragement and guidance.

The results that factors such as students' current programming skills, prior programming experience, and grade expectation are significant predictors of their learning performance have important implications too. The instructors can motivate their students, especially those who do not have much prior programming knowledge and experience, to try to get a higher grade for the course from the first meeting of the course and keep encouraging them throughout the semester. This, as suggested by the findings of this study, will improve students' overall learning performance.

It is worth noting that using an active learning approach does not automatically result in better students' learning performance. Studies have shown that structures of the learning environment (e.g., the curriculum and assessment) are a critical factor to the success of active learning (Miller et al., 1996; Poindexter, 2003). Of course, students' motivation, engagement, and being more responsible are also essential to the success of active learning. Another key to the success of active and problem-based learning is to find a balance between the instructor guidance and the student self-directedness.

Limitations and Future Research

This study has several limitations that need to be addressed in future research. The first limitation is about the sample size. Even though the current 17 samples in one data set and 19 samples in the other are adequate to produce valid statistical results, future research can use a larger sample size to build a stronger case. The second limitation concerns the lack of teaching effectiveness comparison between the two instructors, which may have complicated the difference of students' learning performance between the two sections. However, the courses were well established; and the course design and course delivery were identical except the different learning settings and different instructor in each setting. Furthermore, all the tests given in the course were objective tests. The fact that students in the active learning section outperformed implied that the different learning settings, rather than the difference between the instructors, could be a more significant factor on learning effectiveness. Future study can eliminate this limitation with two settings delivered by the same instructor or by two different instructors at the same level of seniority. The third limitation has to do with the extent to which the findings can be generalized beyond the present study. This study used convenient samples (students enrolled voluntarily in each of the two sections). Students' demographic information, time spent on the course, the extent of peer helping, and learning styles can be

collected, synthesized, and compared in future studies. Future studies can also be designed as a longitudinal study to better generalize the findings of a single study. Future studies can also collect students' reflection in each section about their learning experience. Qualitative data can provide more insights to the phenomenon and supplement the quantitative data analysis.

CONCLUSION

This paper has presented results demonstrating the effectiveness of active learning in an introductory programming course. The active learning approach will be particularly applicable to situations where students lack motivation, engagement, and self-directness. Through active and problem-based learning, students engage more in learning, which in turn improves their learning performance. For the self-directed approach described in this paper to be widely applied, two areas appear to need attention: different learning styles of students and the constant instructor encouragement and guidance. It is our hope that this study will help both the instructors and the students of introductory programming courses improve their teaching and learning effectiveness and efficiency.

REFERENCES

1. Albanese, M., & Mitchell, S. (1993) Problem-based learning: A review of literature on its outcomes and implementation Issues, *Academic Medicine*, 68, 1, 52-81.
2. Beise, C., Myers, M., VanBrackle, L., & 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.
3. Blumenfeld, P.D., Kempler, T.M., & 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), Cambridge University Press, New York, NY.
4. Bonwell, C., & Eison, J. (1991) Active learning: Creating excitement in the classroom, *ERIC Digests*, ED340272 (pp. 1-4), ERIC Clearinghouse on Higher Education, Washington DC: The George Washington University, School of Education and Human Development.
5. Chou, H.W. (2001) Effects of training method and computer anxiety on learning performance and self-efficacy, *Computer in Human Behavior*, 17, 1, 51-69.
6. Cordes, D., & Parrish, A. (1996). Active learning in technical courses, *Proceedings of the 17th Annual National Educational Computing Conference*, Minneapolis, MI.
7. Deitel, P., & Deitel, H. (2009) C: How to program, Pearson/Prentice Hall, Upper Saddle River, NJ.
8. Fredricks, J.A., Blumenfeld, P.C., & Paris, A.H. (2004) School engagement: Potential of the concept, state of the evidence, *Review of Educational Research*, 74, 1, 59-109.
9. 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.
10. Gill, T.G., & Holton, C.F. (2006) A self-paced introductory programming course, *Journal of Information Technology Education*, 5, 95-105.
11. Lippert, S., & Granger, M. (1997) Peer learning in an introductory programming course, *Proceedings of the 12th International Academy for Information Management Annual Conference*, Atlanta, GA.
12. Major, C.H, & Palmer, B. (2001) Assessing the effectiveness of problem-based learning in higher education: Lessons from the literature, *Academic Exchange Quarterly*, 5, 1, 4-11.
13. Martensen, D., Eriksson, H., & Ingleman-Sundberg, M. (1985) Medical chemistry: Evaluation of active and problem-oriented teaching methods, *Medical Education*, 19, 1, 34-42.
14. McConnell, J.J. (1996) Active learning and its use in computer science, *ACM SIGCSE Bulletin*, 28, SI, 52-54.
15. Miller, J., Groccia, J., & 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), Jossey-Bass, San Francisco, CA.
16. Neufeld, D., & 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.
17. Norman, G.R., & Schmidt, H.G. (1992) The psychological basis of problem-based learning: A review of evidence, *Academic Medicine*, 67, 9, 557–565.

18. Perkins, D.N. (1991) Technology meets constructivism: Do they make a marriage? *Educational Technology*, 31, 5, 18-23.
19. Poindexter, S. (2003) Assessing active alternatives for teaching programming, *Journal of Information Technology Education*, 2, 257-265.
20. Prince, M. (2004) Does active learning work? A review of the research, *Journal of Engineering Education*, 93, 3, 222-231.
21. Simon, S.J., & 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.
22. Szajna, B., & 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.
23. Vernon, D., & Blake, R. (1993) Does problem-based learning work? A meta-analysis of evaluative research, *Academic Medicine*, 68, 7, 550-563.
24. Williamson, S., & 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.
25. Wilson, B. (1995) Metaphors for instruction: Why we talk about learning environments, *Educational Technology*, 35, 5, 25-30.
26. Woszczynski, A.B., Guthrie, T.C., & Shade, S. (2005a) Personality and programming, *Journal of Information Systems Educations*, 16, 3, 293-299.
27. Woszczynski, A.B., Haddad, H.M., & Zgambo, A.F. (2005b) An IS students worst nightmare: Programming courses, *Proceedings of the Southern Association of Information Systems Conferences (SAIS 2005)*, February 25-26, Savannah, GA, USA, 130-133.