Journal of
Information
Systems
Education

Volume 33
Issue 2
Spring 2022

Apply Small Teaching Tactics in an Introductory Programming Course: Impact on Learning Performance

Yabing Jiang

Recommended Citation: Jiang, Y. (2022). Apply Small Teaching Tactics in an Introductory Programming Course: Impact on Learning Performance. *Journal of Information Systems Education*, 33(2), 149-158.

Article Link: https://jise.org/Volume33/n2/JISE2022v33n2pp149-158.html

Initial Submission:	22 November 2020
Accepted:	12 May 2021
Published:	15 June 2022

ISSN: 2574-3872 (Online) 1055-3096 (Print)

Apply Small Teaching Tactics in an Introductory Programming Course: Impact on Learning Performance

Yabing Jiang
Department of Information Systems & Operations Management
Lutgert College of Business
Florida Gulf Coast University
Fort Myers, FL 33965, USA
yjiang@fgcu.edu

ABSTRACT

Small teaching approaches are well-structured, incremental teaching improvement techniques supported by research in cognitive science, memory, and learning. I systematically implement a series of small teaching activities in an introductory programming course to tackle the teaching and learning challenges faced by instructors and students. The small teaching activities are designed to promote effective learning strategies such as knowledge retrieval, spacing-out practice, and interleaving learning. I examine the impact of such approaches on students' performance through comparative analyses. The test results indicate that small teaching approaches are effective in improving students' lower- and higher-level thinking skills and help boost students' long-term knowledge retention. Because the small teaching approaches are flexible and easy to implement, instructors teaching technical information systems topics can quickly integrate at least some small teaching activities into their classes.

Keywords: Small teaching, Teaching effectiveness, Student performance, Active learning, Introductory programming

1. INTRODUCTION

Programming fundamentals and programming languages are subtopics within the information systems (IS) body of knowledge specified in the undergraduate IS model curriculum (Gorgone et al., 2003; Topi et al., 2010). At many business schools, an introductory programming course is often designated as a core or an elective course for undergraduate students majoring or minoring in IS, and it typically teaches an object-oriented programming language, such as C++, Java, or Python. Regardless the choice of language, these courses are often considered to be challenging to teach by instructors and difficult to pass by students (Beise et al., 2003; Gill & Holton, 2006; Mok, 2014; Sengupta 2009; Woszczynski et al., 2005).

Introductory programming textbooks often teach programming in a problem-driven way; they focus on problem solving rather than syntax (Liang, 2015). Instructors can rarely teach programming concepts and techniques by just discussing the concepts and rules. Instead, they often use examples that represent different application areas such as business, gaming, and science. This means students must not only learn how to use a new language—mastering rules, concepts, and syntaxes—but also develop applications to solve problems. Many students often have a hard time understanding the abstract aspects of programming and are unable to develop coding solutions because they do not have adequate analytic-thinking and problem-solving skills. Additionally, a solid grasp of materials discussed early is a must for successful learning of new subjects for any programming languages. Students who do not fully understand programming conventions, such as basic rules and syntaxes, or leave their issues on logic, selection, and loops unresolved will quickly find it challenging to learn new subjects. Poor performance on earlier fundamental subjects will snowball into a formidable mountain of cumulated coding errors later, leading to escalated frustrations and quick disengagement from the course (Cavaiani, 2006; Mok, 2014; Sengupta, 2009).

Similar instructional and learning challenges are also found in other IS curricula, such as introductory IS (Riordan et al., 2017), database analysis and design (Connolly & Begg, 2006), business analytics (Saundage et al., 2016), business process integration and enterprise systems (Seethamraju, 2011), and systems analysis and design (Parker et al., 2005). Teaching business students technical IS subjects with traditional lecturing and learning methods is challenging and less effective. As a result, IS faculty have been actively exploring and testing various pedagogical approaches to enhance student engagement in coursework and improve student competency and performance. For example, Zhang et al. (2013) examine the effectiveness of two teaching approaches used in an introductory programming course, one with both lectures and assignments and the other with only assignments. They find that the assignment-only teaching approach produces a significantly higher score improvement than the more traditional approach. Mok (2014) experiments with the flipped-classroom approach in a programming course and finds that it encourages student engagement and is positively received by students. Frost et al. (2015) study the effectiveness of gamification of a learning management system (LMS) in a core IS course in increasing student interest, learning, motivation, satisfaction, and perception of pedagogical effect. Seethamraju (2011) examines the impact of an ERP simulation game on teaching and learning and finds that the simulation game improves students' abilities and contributes to deep learning. Saundage et al. (2016) demonstrate that a learning environment that combines interactivity, visualization, and narratives can help improve student learning outcomes and engagement in a business analytic course. Riordan et al. (2017) find some positive outcomes in a redesigned introduction to IS core course, which is contextualized in a simulated environment and filled with role-playing activities that focus on experiential and active learning.

What these suggested teaching approaches and methods have in common is that they often require a fundamental redesign of content delivery methods, learning activities, assessments, or even the learning management systems. Instructors must not only spend extensive time and efforts to create the redesigned courses, but also put in extra efforts in implementing and delivering the redesigned courses. Furthermore, some redesigns, such as simulation games or gamification of LMS, may require additional financial and/or technical support from the departments. Drastic course redesigns also demand extra effort from students, who are often unfamiliar with these new approaches. As a result, instructors often hesitate to initiate major transformations to a course.

Applying small teaching techniques to teach IS courses, in comparison, is an effective alternative that does not require fundamental course redesign. Small teaching is "an approach that seeks to spark positive change in higher education through small but powerful modifications to our course design and teaching practices" (Lang, 2016, p. 5). The principle behind small teaching techniques is that significant instructional and student performance improvement can be accomplished by incorporating incremental changes in courses instead of conducting dramatic redesigns. While educators have experimented with different small teaching approaches in various disciplines from K-12 to higher education, small teaching is still a rarely studied subject in IS education literature. The purpose of this research is to study the efficacy of applying small teaching techniques to teach technical IS subjects. There are three main objectives of this paper. First, it reviews learning strategies and small teaching techniques that are tested and proven to be effective in improving student learning. Second, it demonstrates how to implement several small teaching techniques in an introductory programming course. Third, it examines the effectiveness of the adopted small teaching approaches in improving student performance and makes recommendations on applying small teaching techniques in IS curriculum.

The remainder of the paper is organized as follows. In the next section, I present a review of principles of learning and discuss the associated small teaching techniques, based on which I develop my hypotheses. In section 3, I discuss the research methodology, including course background, course redesign, data collection, and data analysis. Results of hypothesis testing are also presented. Following that, I discuss the findings, their implications, and directions for future research in section 4. I conclude the paper with recommendations in section 5.

2. PRINCIPLES OF LEARNING, SMALL TEACHING TECHNIQUES, AND HYPOTHESES DEVELOPMENT

To succeed in any IS program, students need to adopt the deep learning approach instead of relying on surface learning, an easier approach that may have worked for them in the past. With surface learning, students concentrate on learning the text itself and memorizing the facts; with deep learning, however, students focus on comprehending the meanings conveyed in the text (Marton & Saljo, 1976). Learners who take the deep-learning approach reflect on what they have read and relate new information to what they already know to develop understanding, rather than focusing on memorization. To develop effective teaching approaches and promote more student learning, instructors must understand the evidence-based principles of learning, encourage deep learning, and educate students on how to study efficiently.

2.1 Principles of Learning

Learning means "acquiring knowledge and skills and having them readily available from memory so you can make sense of future problems and opportunities" (Brown et al., 2014, p. 2). We all know the saying that "practice makes perfect," but simple repetition does not necessarily enhance learning. Massed practices, referring to a learning approach of single-minded, rapid-fire repetition of subjects, such as practicing one type of problem or reading/reviewing text and notes repeatedly, are widely adopted by students and educators, but proven to be less productive (Callender & McDaniel, 2009; Karpicke et al., 2009; McCabe, 2011). The familiarity with, or even fluency in, the underlining subjects gained through such massed practices can create an illusion of learning, but does not translate into actual understanding and mastery of the ideas behind the subjects or how they relate to pre-existing knowledge. Anecdotally, I frequently observe that students in my introductory programming course often have no questions and feel confident that they have mastered the subjects after a few in-class coding practices. However, most of them are puzzled by assignments that are similar to what they have practiced in class but require them to apply what they have learned in multiple classes.

Instead of performing massed practices, students can benefit greatly by adopting more effective learning strategies such as retrieval, spacing-out practices, and interleaving learning. Retrieval practices, such as flashcards, require students to recall what they have read or learned from memory. Quizzes, whether they are self-quizzes or graded quizzes, are a powerful form of retrieval practice because they force students to recall what they have learned (Roediger & Karpicke, 2006; Wheeler & Roediger, 1992). Elaboration practices, asking learners to explain the new material using their own words, also require learners to practice knowledge retrieval and can help them develop understanding, integrate concepts, and connect new learning to previous material. Retrieval activities that require students to put more thought into their answers, such as completing short-answer questions, can produce better learning outcomes than activities that are less effortful, such as answering multiple-choice questions (Butler & Roediger, 2007; Kang et al., 2007; McDaniel et al., 2007).

Spaced practices are retrieval practices that are purposely spaced-out. They have been proven to be even more effective, as they allow for potential forgetting in between sessions and require more cognitive effort from learners (Carey, 2015;

McDaniel et al., 2011; Roediger et al., 2011). In a study conducted on 38 surgical residents, randomly assigned into two groups, Moulton et al. (2006) compared the effectiveness of two approaches teaching microvascular anastomosis: an intensive one-day session (massed practice) covering four lessons versus a spaced-practice approach covering one lesson a week for four weeks. They found that the spaced-instruction produced better outcomes in all aspects measured, because the increased effort required for students to retrieve the learning helped them further embed the learning in their long-term memory, leading to durable learning.

Another effective alternative to massed practice is interleaving. "Interleaving refers to the practice of spending some time learning one thing and then pausing to concentrate on learning a second thing before having quite mastered that first thing, and then returning to the first thing, and then moving onto a third thing, and then returning to the second thing, and so forth" (Lang, 2016, p. 68). Interleaved approach is like a form of spiraling because learners add new layers of learning each time they iterate through the material. Interleaving promotes long-term retention because it involves spacing-out learning sessions over time and mixing-up varied learning activities. This approach forces students to practice how to select and apply the correct solutions for different types of problems and helps them develop a deeper understanding of the associated underlying principles and rules such that they are more capable of choosing the right solutions in unfamiliar situations (Birnbaum et al., 2013). In an experiment conducted on 18 college students, Rohrer and Taylor (2007) showed that the interleaving approach, introducing four models of mathematics problems together and letting students practice how to solve problems involving one of the four types in a random order, produced a significantly better test outcome a week later than the massed practice approach, which teaches one model and lets students practice it repeatedly before moving on to the next model. Studies in different contexts have further shown that interleaving outperforms massed practice in long-term retention and conceptual learning because it helps learners develop understanding of interrelationships of elements (Goode et al., 2008; Jacoby et al., 2010; Kang & Pashler, 2012; Kornell & Bjork, 2008).

Each retrieval practice deepens the neural pathway to the subject in memory, making it readily available for future use (Zull, 2002). Even though massed practice may lead to higher scores on an immediate test, a reason why it is widely accepted and practiced, the resulting learning is often shallow (i.e., illusion of mastery) and not lasting since it leans on short-term memory and promotes short-term learning. Retrieval practices, especially when spaced out and interleaved, call for more cognitive effort and are more challenging to perform. They may seem to be less productive during the practice, compared to massed practices, but they result in stronger learning, long-term retention, and versatile application of knowledge solving known and new problems (Brown et al., 2014).

Rather than letting students pursue the familiar massed practices, IS instructors should encourage students to purposely practice retrieving knowledge from memory when studying. For example, rephrasing the main ideas in their own words after reading the text, or completing self-check questions in the book. Furthermore, instructors can incorporate retrieval practices in course design and content delivery to improve learning, and these can be done incrementally through small teaching techniques, all within the control of instructors.

2.2 Small Teaching Techniques

Lang (2016) categorizes small teaching approaches into three forms. Approach #1: incorporate a brief classroom learning activity, lasting 5 to 10 minutes, at the beginning or end of a class. Such activities, which occupy a small portion of a class, are designed to capture students' attention, promote student engagement, and enhance student learning. Approach #2: conduct a one-time course intervention activity, occupying an entire class period. Such activities could be a session for mindful practice or discussion/debate; a session to create a brief thesis, helping students see the big picture and connections of various subjects discussed; or a session to develop a concept map, helping students visualize the organization of key concepts. These may be a new format, requiring additional preparation compared to regular sessions, but are only used a single time in the semester, accounting for only a small portion of the course. Approach #3: introduce small modifications in course design or communication with students. Such minor changes could be in course and assignment description, course schedule modification, or responses and feedback to students—all of which focus on promoting mindful learning and do not require radical redesign of a course.

Through such small, incremental changes, instructors can provide ample opportunities for students to practice retrieval of older knowledge, compare and connect new and old material, and apply knowledge to new contexts, all with the goal of improving learning and boosting long-term retention. In contrast to drastic approaches that demand significant instructor time and effort to prepare course redesign before the beginning of a semester, each of these small teaching approaches can be designed and implemented right away with limited preparation, and none of them require extra financial or technical support. Additionally, they are accessible to instructors of all ranks and disciplines and are flexible for implementation in a specific class session, in the middle of a semester, or throughout a semester. Adopting small teaching techniques in course design and delivery is by no means an inferior choice compared to teaching techniques that require big changes such as flipped classroom or simulation games. These small teaching techniques are well-structured teaching strategies that are built upon solid research on learning and education and are proven to produce better student learning outcomes than overprepared lectures (Ambrose et al., 2010; Brown et al., 2014; Lang, 2016; Miller, 2014). These techniques have been implemented, studied, and proven to be effective in various disciplines such as art (Kang & Pashler, 2012; Kornell & Bjork, 2008), chemistry (Rogerson, 2003), math (Rohrer & Taylor, 2007), medicine (Moulton et al., 2006), psychology (Leeming, 2002), and social studies (McDaniel et al., 2011).

2.3 Hypotheses

Bloom's Taxonomy represents individuals' cognitive processes on a continuum of increasing cognitive complexity, from lower-level thinking, such as remembering and understanding, to intermediate and higher-level thinking, like applying, analyzing, evaluating and creating (Bloom et al., 1956; Krathwohl, 2002). While introductory programming courses mostly cover the fundamental programing concepts and techniques, the teaching objectives have never been limited to simply memorizing these basics but have focused more on developing understanding of the course materials and applying learning to create coding solutions to business problems. Hence, to do well, students must go beyond employing lower-order thinking skills.

In this study, I examine whether small teaching approaches that emphasize highly effective learning strategies—retrieval, spacing-out practice, and interleaving learning can help students improve lower- and higher-order thinking skills and retain knowledge longer in an introductory programming course. Psychologists have discovered that the more learners practice retrievals and the more effort they must exert to such retrieval practices, the better and deeper they learn and retain knowledge for long-term (Brown et al., 2014). To be effective, retrievals need to be practiced repeatedly. Hence, instead of experimenting with just one or two small teaching activities for one class session, I developed and implemented a series of small teaching activities that emphasize these effective learning strategies in multiple sessions.

Rather than separating the effectiveness of each small teaching activity or learning strategy, I am interested in examining the collective effect of implementing various small teaching activities on student performance. Prior research has found that retrieval practice strengthens the memory and boosts knowledge retention; spacing-out practice requires more effort from learners, leading to stronger learning; and interleaving learning creates opportunities for learners to connect and apply old knowledge to new material and contexts, helping them develop understanding and engage in higher-order thinking tasks (Brown et al., 2014; Lang, 2016; Zull, 2002). Hence, I hypothesize the following:

- **Hypothesis 1 (H1)**: Small teaching approaches that mix up retrieval activities with spacing-out and interleaving practices will help improve students' lower- and intermediate-level thinking skills in an introductory programming course.
- **Hypothesis 2 (H2)**: Small teaching approaches that mix up retrieval activities with spacing-out and interleaving practices will help improve students' higher-order thinking skills in an introductory programming course.
- **Hypothesis 3 (H3)**: Small teaching approaches that mix up retrieval activities with spacing-out and interleaving practices will help improve students' long-term performance in an introductory programming course.

3. RESEARCH METHOD

3.1 Course Background

To test the hypotheses, an introductory programming course in a mid-sized public university in the southeast region of the United States was used to collect data. The course, titled "Introductory Business Programming," introduces students to basic Java programming and has been offered in the college of business as a required course for IS majors and as an elective for IS minors for many years. Most of the students registered in this course are juniors or seniors. This course uses Daniel Liang's "REVEL for Introduction to Java Programming," an animated, interactive digital version of Liang (2015). It covers the following chapters from the textbook: (1) Introduction to Computers, Programs, and Java, (2) Elementary Programming, (3) Selections, (4) Mathematical Functions Characters, and Strings, (5) Loops, (6) Methods, (7) Single-Dimensional Arrays, (9) Objects and Classes, (10) Object-Oriented Thinking, and (11) Inheritance and Polymorphism. Upon successful completion of the course, students should be able to comprehend and apply the basic object-oriented programming concepts and techniques to create applications that solve simple business problems.

One section of the course is offered in both the spring and fall semesters. All sections meet in class twice a week, each for 75 minutes, for 15 weeks. This course uses both individual homework assignments and quizzes to assess learning. Students are informed through the syllabus that the course promotes a student-centered, active-learning approach, under which the lecture session only focuses on explaining key concepts and addressing students' questions, and most of the class time is allocated to hands-on activities to enhance students' understanding of concepts.

3.2 Course Redesign

For this study, the course contents and main course delivery approach were identical in all sections, except that several small teaching techniques were implemented in an experimental section. I used a short presentation in the experimental section to introduce students to retrieval practices, such as interleaving and elaboration, and to inform them that the class will adopt retrieval and interleaving practices.

I often used the first few minutes to review materials covered in the previous session and used the last few minutes to summarize materials discussed for that session. Instead of doing these learning tasks for students, I switched to letting students conduct the review and summary through a series of specially designed retrieval activities in the experimental section to engage more students to actively practice retrieval. Table 1 summaries a few of such small teaching activities. For each of these retrieval exercises, I reviewed answers and provided feedback on students' submissions either in the same or the next session.

Additionally, spacing-out practices and interleaving learning were also purposely incorporated in several sessions. I, having taught the course for many years, observed that students often struggle with selection statements and loops, which are the foundational programming subjects taught at the beginning stage of the course. Some also struggle with the more abstract subjects, such as methods and classes. Hence, I designed small teaching activities to target these problem areas. For example, four class sessions were dedicated for loop structures (Ch5). During the first session, the three loop structures were introduced one by one, and the loop-design strategy was also discussed. Students then practiced on solving a problem three times, using a different loop each time. In the following three sessions, students practiced solving the same set of problems used in the control sections, but each problem was coded three times using a different loop during the same class session, i.e., these activities purposely used interleave learning and space-out retrieval practices. In comparison, for the control sections, the first class focused only on the while-loop structure and how to design a while loop. Students then practiced designing a while-loop in three examples. The second class introduced the do-while loop, and students practiced designing both the while and do-while loops with examples. The third class focused only on the for-loop, and the fourth class summarized all three loops and the remaining materials of the chapter. Interleaving and spacing-out practices were purposely

Subjects and Assessments	Sample Small Teaching Activities on Retrieval Practice
Ch3 Selection Statement (assessed in quiz 2 and assignments #2 and #3)	Write down the syntax of the if-else statement discussed in the previous session.
	Summarize the common errors associated with designing if-else statements.
	Write down the syntax for the switch statement and compare the switch statement with the multi-level if-else statements.
Ch5 Loops (assessed in quiz 2 and assignment #3)	Summarize the loop-design strategy that applies to all three types of loops.
	Write down the pseudocode for each loop structure.
	Compare the three loop structures, while, for, and do-while, and discuss when to use which type.
Ch6 Methods (assessed in quiz 3 and assignment #3)	List and describe the methods that you have been using prior to Ch 6.
	Summarize the method structure introduced in the previous session.
	Write down the syntax for method header and explain each element.
Ch9 and Ch10 Objects and Classes (assessed in quiz 4 and assignment #4)	List and describe the classes that you have been using prior to Ch 9.
	Summarize the class structure introduced in the previous session.
	Define the UML (unified modeling language) diagram for a Student class.
	Summarize the properties of constructors and write down every property that you can recall regarding classes.

Table 1. Sample Small Teaching Activities on Retrieval Practice

not implemented. As a result, for the experimental section, the class discussed and compared the three loop structures in multiple sessions whereas for the control sections the class only did it in one session.

Similarly, when covering methods and classes, I purposely introduced most key concepts during the first session for the experimental section. In comparison, for the control sections, I spread out discussions of key concepts into multiple sessions. Thus, for the experimental section, the main concepts were demonstrated, retrieved, and further explained in multiple sessions.

3.3 Data Collection

I implemented small teaching techniques in a spring semester. To test the hypotheses, student data from three consecutive spring semesters were collected, with two sections of 15 and 20 students in the earlier semesters as the control group and the small teaching experiment (STE) semester of 20 students as the test group. This course switched to the Python language after the experimental semester, and as a result, only one section was used as the test group. To be more comparable with the test group, previous spring sections were used as the control group because fall sections usually have a bigger class size (35 to 40 students) and a different student dynamic (such as the numbers of transfer students and older students pursuing a second career).

All sections of the course were taught by me. The only instructional difference between the control and the STE sections was that several small teaching activities (as described in section 3.2) were implemented in the latter. Hence, student performance data on a series of common individual assessments for the two groups were collected, and comparisons of the means of students' grades were used to test the hypotheses. Specifically, student data on three quizzes, three homework assignments, and one cumulative exam were collected. The quizzes and assignments were assigned at different points of the course after the assessed chapters had been discussed (see the subjects and assessments mapping in Table 1). Each of these collected quizzes was worth 2 points; each assignment was worth 3 points; the cumulative exam was worth 15 points; and the total points for the course were 100. For both the control and STE groups, the quizzes and the exam were auto-graded by Canvas, a learning management system, and the assignments were graded by me, following the same rubrics.

All quizzes were composed of multiple-choice and multiple-answer types of questions, and they were administered through Canvas. Students did not need to write code when taking quizzes. However, there were some problem-solving types of questions that required them to analyze code, interpret code, identify coding errors, and compare coding approaches to select the correct answers. That is, quizzes were not limited to lower-order thinking such as recalling and understanding, but also required some degree of intermediate-level thinking such as applying and analyzing. Therefore, student data on the three quizzes were used to test H1.

Each assignment required students to develop programming code to solve a given problem. Students were expected to analyze the problem, apply programming concepts and procedures that they have learned, evaluate possible alternative approaches, and perform further analysis and evaluation to debug code. Thus, assignments required students to apply higher-order thinking skills as defined in the updated Bloom's taxonomy (Krathwohl, 2002). Student data on the three assignments were used to test H2.

For the cumulative exam, 49 out of the 60 test questions were on subjects that were taught at least four weeks before the exam. Hence, student data on the exam were used to test H3.

3.4 Data Analysis and Hypothesis Test Results

Two datasets were obtained for this study. Table 2 presents the descriptive statics of the control and STE groups on quizzes, assignments, and the exam. Here, N refers to the group size for the assessed item. The analysis was based on available data since not all students submitted all the assessment items and missing submissions provide no information regarding students' performance. A simple comparison of the means of the students' grades for the two groups shows that students in the STE section performed better than students in the control sections on all the quizzes and the cumulative exam. Also, students in the STE group performed better than students in the control group on assignments #3 and #4, but students in the control group performed slightly better than students in the STE group on assignment #2.

Next, I tested whether the two samples were of equal variance with the Levene's test to determine which type of t-test to perform for each assessment. The null hypothesis of a Levene's test states that the variances for the two groups are equal, and the null hypothesis is rejected if the *p*-value is less than 0.05. Since the null hypothesis was rejected for quiz 2, quiz 3, the exam, and assignment #3, t-tests for two-sample of unequal variance were performed on these assessments, and t-tests for two-sample of equal variance were performed on others. Table 3 summarizes the results of the Levene's tests and the corresponding t-tests.

Because quizzes assessed students' ability of recognizing the correct answer through recalling definitions, contrasting options, analyzing and interpreting code, and identifying errors, the three quizzes were used to test H1 that small teaching approaches help improve students' lower- and intermediate-level thinking skills. The two-sample t-tests indicated that the means of the STE group were statistically significantly higher (at $p < 0.1$ level) than that of the control group for quiz 3 and quiz 4, whereas the STE group's improvement on quiz 2 was not significant. Thus, H1 was weakly supported by two of the three quizzes.

The two-sample t-tests on assignments indicated that students in the STE section performed statistically significantly better than that of the control sections on assignment #3 (at $p < 0.05$ level) and assignment #4 (at $p < 0.1$ level). For assignment #2, the control group performed better, but the difference was not statistically significant. Each of these assignments required students to apply higher-level thinking skills to create coding solutions for a problem, and hence they were used to test H2. Past student data and student feedback showed that they considered assignment #3 as the most complicated assignment as it required students to develop multi-level if-else statements, both a for-loop and a while- (or do-while) loop, and four methods. In the STE section, spacing-out practice and interleaving learning were purposely implemented when covering subjects assessed in assignments #3 and #4. The two-sample t-test on assignment #3 provided a statistically strong support for H2 whereas the test on assignment #4 only provided a statistically weak support for H2, though results of assignment #2 did not support H2.

The two-sample t-test indicated that the mean of the STE group on the cumulative exam was significantly higher (at $p < 0.05$ level) than that of the control group. Because the only instructional difference between the STE and the control sections was that the former implemented small teaching techniques, this test result provided a statistically strong support for H3 that the small teaching approaches help improve students' long-term performance.

	quiz2		quiz3		quiz4		#2		#3		#4		exam	
	STE	Control	STE	Control	STE	Control	STE	Control	STE	Control	STE	Control	STE	Control
N	20	33	20	32	19	31	20	32	20	33	20	28	20	32
Mean	1.52	1.40	1.45	1.28	1.59	1.46	2.60	2.62	2.76	2.46	2.91	2.77	12.07	11.02
Median	1.60	1.50	1.52	1.23	1.60	1.46	2.60	2.70	3.00	2.80	3.00	3.00	12.11	11.34
Std Dev	0.26	0.54	0.32	0.46	0.26	0.32	0.36	0.34	0.34	0.66	0.26	0.42	1.48	1.97
Variance	0.07	0.29	0.10	0.22	0.07	0.10	0.08	0.06	0.08	0.11	0.06	0.08	2.20	3.89
Std Err	0.06	0.09	0.07	0.08	0.06	0.06	0.13	0.12	0.12	0.44	0.07	0.18	0.33	0.35
Min	0.88	0.33	0.66	0.45	0.88	0.84	1.50	1.70	2.20	0.70	1.85	1.40	9.25	7.64
Max	1.90	2.00	1.88	1.98	1.90	2.00	3.00	3.00	3.00	3.00	3.00	3.00	14.13	14.38
1st quartile	1.32	1.00	1.37	0.86	1.47	1.21	2.50	2.50	2.48	2.00	2.98	2.80	11.53	9.12
3rd quartile	1.63	1.90	1.63	1.66	1.77	1.68	2.85	3.00	3.00	3.00	3.00	3.00	13.14	12.71

Table 2. Descriptive Statistics for Quizzes, Assignments, and the Exam

		Levene's Test for Equality of Variances		t-test for Equality of Means			
Hypothesis tested	Assessments	F	P-value	t Stat	df	Sig (one-tail)	Mean Diff
H1	quiz 2	10.2296	0.0024***	1.1039	49	0.1375	0.1214
H1	quiz3	8.4061	0.0055***	1.6401	49	0.0537*	0.1787
H1	quiz4	1.8820	0.1765	1.5200	48	0.0675*	0.1309
H2	#2	0.0199	0.8883	-0.1894	50	0.5747	-0.0187
H2	#3	4.9939	0.0298**	2.1964	50	0.0164**	0.3024
H2	#4	1.8588	0.1794	1.3634	46	0.0897*	0.1446
H3	exam	4.6161	0.0365**	2.1836	48	0.0170**	1.0513
**p<0.1, **p < .05, ***p < .01*							

Table 3. Results of Levene's Tests for Homogeneity of Variance Based on the Median and Comparisons of Means of Assessments between the STE and Control Group

4. DISCUSSION

4.1 Implications of Findings

The results of independent samples' t-tests showed that students in the STE section performed statistically significantly better than students in the control sections on five out of the seven tested assessments. These findings have important practical implications. First, small teaching approaches can help improve not only students' lower-level thinking skills (H1, supported by tests on quizzes 3 and 4) but also their higher-level thinking skills (H2, supported by tests on assignments #3 and #4). As part of the implemented small teaching techniques, repeated retrieval activities helped students retrieve knowledge and develop understanding better because they forced students to practice recalling and summarizing key programming concepts, rules, and basic syntax. Naturally, good lower-order thinking skills provide the imperative foundation for developing higher-order thinking skills. Spaced-out practices and interleaved learning activities further provided opportunities for students to develop higher-order thinking skills, because they not only let students practice retrieving older knowledge, but also relate new subjects to what they already knew, differentiate and select appropriate tools, learn to apply knowledge to new contexts, organize and design coding elements, and create coding solutions to presented problems.

Second, small teaching approaches can help boost students' long-term knowledge retention (H3, supported by the test on the cumulative exam). In this study, I designed small teaching activities to engage all students to respond to short-answer questions or solve problems; solutions were not directly presented to students. These small teaching approaches focused on active learning, which is more effectual in motivating and engaging students (Prince, 2004). They were more effective in producing stronger and long-lasting learning benefits, compared to simply presenting students with answers (traditional lecturing) or practicing with multiple-choice questions. They required students to not only recall knowledge but also generate the answers, and hence were more challenging to practice, demanding more learning efforts from students, but can also lead to complex mastery and deeper and long-lasting understanding. Repeated practices of such small teaching activities can strengthen students' memory of the course subject for the long term and boost their abilities to recall and apply it for future use.

Third, small teaching approaches need to be practiced repeatedly to gain students' acceptance and to obtain better learning outcomes. In this study, assignment #2 and quiz 2 were the first two assessments administered after I began practicing small teaching approaches, and they were the only assessments that did not provide a significant support for the tested hypotheses. Students might be new to the effective learning strategies promoted by small teaching, and it may take repeated practices for them to adapt to and engage in retrieval practices to achieve performance improvement, evidenced by their significant performance improvements on the next five assessments. Introducing students to the evidence-based effective learning strategies and informing them of the small teaching approaches adopted for the course may help raise their awareness, and making retrieval exercises as low-stake assessments may help promote students' buying-in. Regularly reminding students to draw the connections between the coding exercises and the concepts discussed and asking them to elaborate on what they are doing and why may also help in establishing students' acceptance of the small teaching approaches and improving their overall learning performance.

Fourth, a comparison of the students' performance data indicates that the small teaching approaches are effective in improving student performance, especially for students in the bottom-half of the class. For quizzes and the exam, the STE group not only had a higher mean score, but also a higher median than the control group. In fact, the score ranges for the bottom-half of the students of the STE group were much higher than that of the control group on all quizzes and the exam (see Table 2). Similarly, students of the STE group performed better than that of the control group on assignments #3 and #4, mostly because the performance of the bottom-half of the students had improved significantly. This could imply that the small teaching approaches have helped the less able and moderately able students the most. One reason could be that not all students are accustomed to the effective learning strategies. The implemented small teaching activities may have helped students realize the actual state of their learning, i.e., what they did and did not know, and informed them where to focus on for further study. It is important to provide timely review and feedback so that students know and understand the answers and the correct approaches.

Overall, findings of this study indicate that small teaching approaches are effective in improving students' lower- and higher-level thinking skills in the introductory programming course. The applied changes in course design and content delivery are all small and manageable, but they do require some preparation by the instructors, such as designing the questions used for retrieval practice, rearranging the sequence of subjects to space out retrievals and interleave learning, and modifying class plans to accommodate these changes. Instructors also need to grade submissions of retrieval exercises to identify gaps in student learning and to provide feedback and targeted reviews. Because the small teaching approaches are flexible and easy to implement, instructors who are teaching programming or other IS courses can quickly integrate at least some small teaching activities into their classes.

4.2 Limitations and Future Research

The findings of this study demonstrate that small teaching approaches improve student learning in an introductory programming course. This research, by design, has controlled potential influencing factors such as the differences in instructors and course content. However, it still has a few limitations. First, the sample was limited to the students taking a specific introductory programming course across multiple

		Programming Skill Levels							
	Total	High	Intermediate	Some or limited	Weak	Entry or novice	Basic	Min/very min	Little/no experience
STE	15	0.00%	0.00%	0.00%	13.33%	13.33%	0.00%	40.00%	33.33%
Control 1	15	6.67%	0.00%	6.67%	0.00%	40.00%	6.67%	0.00%	40.00%
Control 2	19	0.00%	5.26%	5.26%	0.00%	31.58%	15.79%	21.05%	21.05%

Table 4. Summary of Student Self-Reported Programming Skill Levels

semesters. The control group included two sections for a good sample size. In comparison, the sample size for the STE group was small, though not untenable for the statistical tests of hypotheses depicted here (Moulton et al. 2006; Zhang et al. 2013). Expanding the study to other IS courses with larger samples will help further validate the findings and provide support for the general effectiveness of small teaching techniques in improving student performance.

Second, it was assumed that students in the two groups were comparable in their overall abilities, but students' GPAs were not collected to validate this assumption. Nevertheless, during the first week, students self-reported their programming skill levels, which are summarized in Table 4.

For the STE section, 15 students reported. For the control group of two sections, 15 and 19 students reported. As indicated in Table 4, students of the STE section had no better prior programming skill levels than that of the control group. From my subjective observations, the students in each group were comparable in their ability of learning. In this study, students in the STE section were taught in the same way as the students in the control group for chapters 1 and 2 since the implementation of small teaching activities began with chapter 3. I analyzed student performance data on two more assessments, quiz 1 and assignment #1, which covered subjects discussed in chapters 1 and 2. A comparison of the means of students' grades for the two groups showed that the STE section had a higher mean on quiz 1 (1.62 vs. 1.55) but a lower mean on assignment #1 (2.78 vs. 2.80) than the control sections. The two-sample t-tests, however, showed that none of these differences were statistically significant (quiz 1: df = 49, t = 1.02263, p = 0.15575; #1: df = 51, t = -0.22552, p = 0.58876). Since students performed similarly when there were no instructional differences, this provides some support that students in the two groups were comparable in their overall ability.

Student feedback was not incorporated into this research to evaluate their opinions on the small teaching approaches. It is possible that some students were more engaged in small teaching activities than others. This research focuses on studying the impact of small teaching on the overall performance of the whole student group. Taking a different research direction, future studies can include a student survey to evaluate students' opinions on small teaching activities, whether these teaching approaches have changed their learning behaviors, and how students' opinions and changes in learning behaviors have affected their individual learning performance. For this study, I adopted the active-learning approach for both the control and STE sections. It is possible that students who are exposed to the active-learning approach may be more receptive to the small teaching approaches. Future research could examine and compare the effectiveness of small teaching on courses with and without active-learning. Additionally, this research focuses on implementing small teaching approaches #1 and #3, not because they are easier to carry out but because they are more suitable for the introductory programming course and can generally be applied to other IS courses. One can think of approach #2 as a miniature of a teaching approach that requires dramatic course redesign, such as gamification, since instructors only need to use whatever the chosen course intervention activity once rather than throughout the semester. Future research could study the implementation and effectiveness of small teaching approach #2 or a combination of all three approaches on assorted IS courses.

5. CONCLUSIONS

This paper demonstrates how instructors can tackle the teaching and learning challenges in introductory programming courses by systematically implementing small teaching activities that promote effective learning strategies such as knowledge retrieval, spacing-out practice, and interleaving learning. The results of comparative analyses demonstrate that small teaching approaches are effective in improving students' lower- and higher-level thinking skills and help boost students' long-term knowledge retention.

The small teaching techniques adopted in this study are guided by research developments in cognitive science, memory, and learning. Instead of undertaking a drastic course redesign that may be time-consuming to develop and implement, small teaching takes a deliberate and well-structured incremental approach in course design and content delivery. It is flexible and easy to implement, making it accessible to instructors of all ranks and suitable for a face-to-face or virtual setting. While instructors can control which small teaching approaches to take and how much small teaching to implement, it is important to note that retrievals need to be practiced repeatedly in spaced-out sessions and with some degree of difficulty to achieve stronger learning and longer retention. Instructors do need to exert additional efforts to design learning activities, modify class plans and course schedule, and revise assignments and assessments, in addition to handling extra grading and providing feedback. Effective small teaching also requires students' cooperation and frequent interaction between the instructor and the students.

This study demonstrates various examples of small teaching activities for an introductory programing course. For instructors interested in adopting small teaching approaches, they can easily adapt such activities to the contexts of other IS courses. For example, they can create their own retrieval practice questions by replacing the programming subjects in Table 1 with subjects related to their courses. They can also create their own interleaving learning activities and purposely space-out retrieval practices by redesigning exercises and making minor changes in their teaching plans. They can design assignments requiring students to apply concepts and skills that they have learned in multiple units. Additionally, instead of using a midterm and a final exam as the main assessments, instructors may consider spacing out several major quizzes and making them all cumulative. Instead of using all multiple-choice questions, instructors can incorporate more short-answer or problem-solving types of questions to purposely let students practice effortful retrievals. They can start with one approach or one-type of activities and gradually create more activities or introduce additional approaches. One benefit of the small teaching approaches is that the created activities and redesigned assignments can be reused in subsequent terms. Furthermore, the instructors can transfer small teaching experience gained from one course to other IS courses with ease.

I encourage IS educators to explore the small teaching framework, experiment, adapt and develop various formats and activities that work for their courses, and examine the effectiveness of small teaching in improving student learning on assorted IS subjects. Through implementing and promoting effective learning strategies and active learning, the small teaching approaches are promising in improving students' learning of technical IS topics that require deep learning to succeed and producing beneficial outcomes for both students and instructors.

REFERENCES

Ambrose, S. A., Bridges, M. W., DiPietro, M., Lovett, M. C., & Norman, M. K. (2010). *How Learning Works: Seven Research-Based Principles for Smart Teaching*. San Francisco, CA: Jossey-Bass.

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.

Birnbaum, M. S., Kornell, N., Bjork, E. L., & Bjork, R. A. (2013). Why Interleaving Enhances Inductive Learning: The Roles of Discrimination and Retrieval. *Memory & Cognition*, 41(3), 392-402.

Bloom, B., Engelhart, M., Furst, E. J., Hill, W. H., & Krathwohl, D. R. (1956). *Taxonomy of Educational Objectives: The Classification of Educational Goals. Handbook I: Cognitive Domain*. London, UK: Longmans.

Brown, P. C., Roediger III, H. L., & McDaniel, M. A. (2014) *Make It Stick: The Science of Successful Learning*. Cambridge, MA: Harvard University Press.

Butler, A. C., & Roediger III, H. L. (2007). Testing Improves Long-Term Retention in a Simulated Classroom Setting. *European Journal of Cognitive Psychology*, 19(4-5), 514-527.

Callender, A. A., & McDaniel, M. A. (2009). The Limited Benefits of Rereading Educational Texts. *Contemporary Educational Psychology*, 34(1), 30-41.

Carey, B. (2015). *How We Learn: The Surprising Truth About When, Where, and Why It Happens*. New York, NY: Random House.

Cavaiani, T. P. (2006). Object-Oriented Programming Principles and the Java Class Library. *Journal of Information Systems Education*, 17(4), 365-368.

Connolly, T. M., & Begg, C. E. (2006). A Constructivist-Based Approach to Teaching Database Analysis and Design. *Journal of Information Systems Education*, 17(1), 43-53.

Frost, R. D., Matta, V., & MacIvor, E. (2015). Assessing the Efficacy of Incorporating Game Dynamics in a Learning Management System. *Journal of Information Systems Education*, 26(1), 59-70.

Gill, T. G., & Holton, C. F. (2006). A Self-Paced Introductory Programming Course. *Journal of Information Technology Education*, 5(1), 95-105.

Goode, M. K., Geraci, L., & Roediger, H. L. (2008). Superiority of Variable to Repeated Practice in Transfer on Anagram Solution. *Psychonomic Bulletin & Review*, 15(3), 662-666.

Gorgone, J., Davis, G. B., Valacich, J. S., Topi, H., Feinstein, D. L., & Longenecker, H. E. (2003). IS 2002 Model Curriculum and Guidelines for Undergraduate Degree Programs in Information Systems. *Communications of the Association for Information Systems*, 11(1), Article 1.

Jacoby, L. L., Wahlheim, C. N., & Coane, J. H. (2010). Test-Enhanced Learning of Natural Concepts: Effects on Recognition Memory, Classification, and Metacognition. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 36(6), 1441-1451.

Kang, S. H., McDermott, K. B., & Roediger III, H. L. (2007). Test Format and Corrective Feedback Modify the Effect of Testing on Long-Term Retention. *European Journal of Cognitive Psychology*, 19(4-5), 528-558.

Kang, S. H., & Pashler, H. (2012). Learning Painting Styles: Spacing Is Advantageous When It Promotes Discriminative Contrast. *Applied Cognitive Psychology*, 26(1), 97-103.

Karpicke, J. D., Butler, A. C., & Roediger III, H. L. (2009). Metacognitive Strategies in Student Learning: Do Students Practice Retrieval When They Study on Their Own? *Memory*, 17(4), 471-479.

Kornell, N., & Bjork, R. A. (2008). Learning Concepts and Categories: Is Spacing the "Enemy of Induction"? *Psychological Science*, 19(6), 585-592.

Krathwohl, D. R. (2002). A Revision of Bloom's Taxonomy: An Overview. *Theory Into Practice*, 41(4), 212-218.

Lang, J. M. (2016). *Small Teaching: Everyday Lessons from the Science of Learning*. San Francisco, CA: Jossey-Bass.

Leeming, F. C. (2002). The Exam-A-Day Procedure Improves Performance in Psychology Classes. *Teaching of Psychology*, 29(3), 210-212.

Liang, Y. D. (2015). *Introduction to Java programming (10th ed. brief version)*. Hoboken, NJ: Pearson.

Marton, F., & Saljo, R. (1976) On Qualitative Differences in Learning I: Outcome and Process. *British Journal of Educational Psychology*, 46(1), 4-11.

McCabe, J. (2011). Metacognitive Awareness of Learning Strategies in Undergraduates. *Memory & Cognition*, 39(3), 462-476.

McDaniel, M. A., Agarwal, P. K., Huelser, B. J., McDermott, K. B., & Roediger III, H. L. (2011). Test-Enhanced Learning in a Middle School Science Classroom: The Effects of Quiz Frequency and Placement. *Journal of Educational Psychology*, 103(2), 399-414.

McDaniel, M. A., Anderson, J. L., Derbish, M. H., & Morrisette, N. (2007). Testing the Testing Effect in the Classroom. *European Journal of Cognitive Psychology*, 19(4-5), 494-513.

Miller, M. D. (2014). *Minds Online*. Cambridge, MA: Harvard University Press.

Mok, H. N. (2014). Teaching Tip: The flipped Classroom. *Journal of Information Systems Education*, 25(1), 7-11.

Moulton, C. A. E., Dubrowski, A., MacRae, H., Graham, B., Grober, E., & Reznick, R. (2006). Teaching Surgical Skills: What Kind of Practice Makes Perfect? *Annals of Surgery*, 244(3), 400-409.

Parker, K. R., LeRouge, C., & Trimmer, K. (2005). Alternative Instructional Strategies in An IS Curriculum. *Journal of Information Technology Education: Research*, 4(1), 43-60.

Prince, M. (2004). Does Active Learning Work? A Review of The Research. *Journal of Engineering Education*, 93(3), 223-231.

Riordan, R. J., Hine, M. J., & Smith, T. C. (2017). An Integrated Learning Approach to Teaching an Undergraduate Information Systems Course. *Journal of Information Systems Education*, 28(1), 59-69.

Roediger III, H. L., Agarwal, P. K., McDaniel, M. A., & McDermott, K. B. (2011). Test-enhanced Learning in the Classroom: Long-Term Improvements from Quizzing. *Journal of Experimental Psychology: Applied*, 17(4), 382-395.

Roediger III, H. L., & Karpicke, J. D. (2006). Test-Enhanced Learning: Taking Memory Tests Improves Long-Term Retention. *Psychological Science*, 17(3), 249-255.

Rogerson, B. (2003). Effectiveness of a Daily Class Progress Assessment Technique in Introductory Chemistry. *Journal of Chemical Education*, 80 (2), 160-164.

Rohrer, D., & Taylor, K. (2007). The Shuffling of Mathematics Problems Improves Learning. *Instructional Science*, 35(6), 481-498.

Saundage, D., Cybulski, J. L., Keller, S., & Dharmasena, L. (2016). Teaching Data Analysis with Interactive Visual Narratives. *Journal of Information Systems Education*, 27(4), 233-247.

Seethamraju, R. (2011). Enhancing Student Learning of Enterprise Integration and Business Process Orientation Through an ERP Business Simulation Game. *Journal of Information Systems Education*, 22(1), 19-29.

Sengupta, A. (2009). CFC (Comment-First-Coding) - A Simple Yet Effective Method for Teaching Programming to Information Systems Students. *Journal of Information Systems Education*, 20(4), 393-399.

Topi, H., Valacich, J., Wright, R. T., Kaiser, K. M., Nunamaker, J. F., Sipior, J. C., & Vreede, G. J. (2010). IS 2010: Curriculum Guidelines for Undergraduate Degree Programs in Information Systems. *Communications of the Association for Information Systems*, 26(1), 359-428.

Wheeler, M. A., & Roediger III, H. L. (1992). Disparate Effects of Repeated Testing: Reconciling Ballard's (1913) and Bartlett's (1932) Results. *Psychological Science*, 3(4), 240-246.

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

Zhang, X., Zhang, C., Stafford, T. F., & Zhang, P. (2013). Teaching Introductory Programming to IS Students: The Impact of Teaching Approaches on Learning Performance. *Journal of Information Systems Education*, 24(2), 147-155.

Zull, J. E. (2002). *The Art of Changing the Brain: Enriching Teaching by Exploring the Biology of Learning*. Sterling, VA: Stylus Publishing, LLC.

AUTHOR BIOGRAPHY

Yabing Jiang is an associate professor of information systems in the Lutgert College of Business at the Florida Gulf Coast University. She holds a Ph.D. in Computer Information Systems from the William E. Simon Graduate School of Business Administration, University of Rochester. Her research interests focus on applying economic theories and methodologies to study IT related topics such as new business models and pricing strategies in electronic commerce, online word-of-mouth, incentive contracting in service facilities, outsourcing contract design, integration and sabotage, the role of IT in corporate governance, and teaching innovation. Her research appears in *Decision Support Systems*, *Electronic Commerce Research and Applications*, *Information Systems Research*, *Journal of Management Information Systems*, *Journal of Revenue & Pricing Management*, *Production and Operations Management*, among others. She has received grants and awards recognizing her research impacts.

Information Systems & Computing Academic Professionals

Education Special Interest Group

ISSN 2574-3872