Information Communication Technologies for Enhanced Education and Learning:
Advanced Applications and Developments

Lawrence Tomei
Robert Morris University, USA

Information Science REFERENCE

INFORMATION SCIENCE REFERENCE

Hershey · New York

Director of Editorial Content: Kristin Klinger
Director of Production: Jennifer Neidig
Managing Editor: Jamie Snavely
Assistant Managing Editor: Carole Coulson
Typesetter: Lindsay Bergman
Cover Design: Lisa Tosheff
Printed at: Yurchak Printing Inc.

Published in the United States of America by
Information Science Reference (an imprint of IGI Global)
701 E. Chocolate Avenue, Suite 200
Hershey PA 17033
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@igi-global.com
Web site: http://www.igi-global.com

and in the United Kingdom by
Information Science Reference (an imprint of IGI Global)
3 Henrietta Street
Covent Garden
London WC2E 8LU
Tel: 44 20 7240 0856
Fax: 44 20 7379 0609
Web site: http://www.eurospanbookstore.com

British Cataloguing in Publication Data
A Cataloguing in Publication record for this book is available from the British Library.

Editorial Advisory Board

This chapter was originally published as a journal article as follows: Emurian, H.H. Teaching Java: Managing instructional tactics to optimize student learning. International Journal of Information & Communication Technology Education, 2007, 3(4), 34-49.

Chapter XIV
Teaching Java™: Managing Instructional Tactics to Optimize Student Learning

Henry H. Emurian
University of Maryland—Baltimore, USA

ABSTRACT

Information systems students in a graduate section and an undergraduate section of an introductory Java graphical user interface course completed the following initial assignments to learn a simple program: (1) automated programmed instruction tutoring, (2) hands-on learning with a lecture, and (3) collaborative peer tutoring. Tests of knowledge transfer and software self-efficacy were administered before students began the first assignment and following completion of each one. The results showed progressive improvement in rule test performance and software self-efficacy across the several instructional events. Taken together, the results of these classroom observations extend the generality of previous work to an updated set of instructional materials and assignments, and that outcome shows the reliability of the learning processes with new groups of students. Students who are new to Java had the privilege of exposure to an initial repertoire of teaching tactics that are synergistic and cumulative.

INTRODUCTION

The research reported here is part of an ongoing stream of formative evaluations of instructional tactics that are intended to help novice, college-level students acquire skill and confidence in computer programming by means of an integrative approach to curriculum development (Emurian, in press:a). Direct mastery of the core knowledge in a discipline is recognized as a fundamental requirement to apply and extend that knowledge to solve novel problems, and that implies consid-

eration of an instructional design to overcome the empirically verified shortcomings of teaching tactics that provide minimal guidance during a student's learning experiences (Kirschner, Sweller, & Clark, 2006). The integrative tactics adopted in our classrooms are in furtherance of helping all of our students to succeed.

Our previous work consistently confirmed the value of programmed instruction in teaching introductory information systems students a simple Java applet as a first technical training exercise in preparation for advanced learning (Emurian, 2004, 2005, 2006a,b). A Web-based, programmed instruction tutoring system to accomplish that objective was presented in Emurian, Hu, Wang et al., (2000), and behavior principles supporting the design and implementation of the system were described by Emurian, Wang, and Durham (2003) and Emurian and Durham (2003). Similar value of programmed instruction is evident in its applications within other symbol intensive disciplines, such as chemistry (Kurbanoglu, Taskesenligil, & Sozbilir, 2006), and its training effectiveness in fostering parent-teacher communications has been demonstrated (Ingvarsson & Hanley, 2006). The objectives of our work are to apply programmed instruction and to assess its effectiveness as a tactic to promote a common level of mastery by all students for a designated learning objective in Java programming. An optimal outcome of such a direct mastery approach is taken to reflect a *true gain* in learning (Anderson, Corbett, Koedinger et al., 1995).

Among several recommendations for effective learning principles to foster retention and transfer of knowledge is repeated practice with different instructional modalities (Halpern & Hakel, 2003) and with socially supported interactions (Fox & Hackerman, 2003). The modalities that have been adopted in our most recent classroom applications include: (1) programmed instruction, (2) lectures with hands-on learning, and (3) collaborative peer tutoring (Emurian, 2006b; in press:b). These tactics are demonstrably effective in promoting programming skill, software self-efficacy, and generalizable knowledge, the latter reflecting *far transfer* of learning (Barnett & Ceci, 2002). The benefits on student learning of a somewhat different, "blended" instructional approach to teaching introductory Java have been reported by Boyle, Bradley, Chalk et al. (2003), where repetition of similar topics occurred throughout the course syllabus. Our assessments of student learning, however, sometimes showed room for improvement in the goal of achieving maximal performance by all students on a far transfer test that was administered immediately following collaborative peer tutoring (Emurian, 2006b; in press:b).

To potentiate the effectiveness of the collaborative peer tutoring, the present classroom studies undertook a modification to the instructions and materials that made available to students to prepare them for collaborative peer tutoring and to use during the collaboration session. The modified procedure allowed the collaborating students to view and discuss together the questions that constituted the test of far transfer. Collaborating students also had direct hypertext access to instructional frames that were otherwise presented sequentially and contingently within the Java programmed instruction tutoring system. Finally, the Java program to be learned by students, as the first technical exercise in a course, contained more items of code to be mastered in comparison to the previous work in this area of classroom applications and research.

METHOD

Subjects

Subjects were as follows: (1) 13 graduate students, four females and nine males, taking IS 613 (*GUI Systems Using Java*) during a four-week summer session (summer 2006), and (2) 14 upper-level undergraduate students, six females and eight

males, taking the equivalent undergraduate course (IS 413) during a 14-week fall session (fall 2006). There were more students enrolled in each class than are represented in the data analysis, which was based only on data collected on all assessment occasions by the students. If a student missed any data collection class or assignment, that student was not included in the analysis. The summer 2006 class met three times each week, and each class lasted three hours. The fall 2006 class met once each week for 2.5 hours. The course was designed for information systems students, and the prerequisite was one prior programming course for both classes. The technical content was identical for both classes, but there were more presentation and writing assignments, based upon reviews of journal articles, for the graduate students in comparison to the undergraduate students.

Prior to using the tutor, demographic information was collected, including age, number of prior programming courses taken, rated Java experience, and rated programming experience. The rating scales were 10-point ordinal scales where ***1 = No experience. I am a novice*** to ***10 = Extensive experience. I am an expert.*** Appendix A presents the scales that were administered during the pre-tutor and post-tutor assessments.

For the summer 2006 class, the background characteristics of the students were as follows: age (median = 28 yrs, range = 23 to 33), number of prior programming courses taken (median = 3, range = 1 to 15), rated prior Java experience (median = 2, range = 1 to 5), and rated prior programming experience (median = 5, range = 2 to 8).

For the fall 2006 class, the background characteristics of the students were as follows: age (median = 22 yrs, range = 21 to 32), number of prior programming courses taken (median = 5.5, range = 3 to 8), rated prior Java experience (median = 2, range = 1 to 7), and rated prior programming experience (median = 5, range = 2 to 8). A Welch robust test (Maxwell & Delaney, 2004, p. 134) showed a significant difference only for the age variable (W = 11.231, p = .003).

The research protocol was exempt from informed consent by the IRB, and the course syllabus clearly indicated that questions both embedded in the Java tutor and administered during several assessment occasions in class were eligible to appear on a graded quiz. The course description and syllabus provided information about the Java tutor and the collaborative peer tutoring, and they presented the rationale for the repetition of initial learning using the several different instructional modalities under consideration.

Materials[2]

Java Program and Tutor

The instructional tactics in this study are based upon teaching students a JApplet program that would display a JLabel object within a browser window on the World Wide Web. The program was arbitrarily organized into 11 lines of code (*e.g.*, ***JLabel myLabel;***) and 37 separate items of code (*e.g.*, ***getContentPane()***). The 37 items (1 item per cell), and the 11 lines of code are presented in Table 1.

The rationale supporting the tutor's design is based upon the learn unit formulation of Greer and McDonough (1999). In the tutoring system, each successive component, or learn unit, within eight tutor stages, required accurate responding for the learner to transition from one component to the next. The occasion and events supporting such a transition constitute a ***natural fracture of instruction***, which is "a unit of a compound that separates naturally from other components as a result of lawful conditions" (Greer, 2002, p. 18). Each cell and each line in Table 1 constituted a learn unit, and there were other learn units in the tutor.

The Web-based Java tutor consists of the following eight stages: (1) introduction and example of the program running in a browser (learn units = 1), (2) learning to copy an item of code (learn units = 37), (3) learning to recognize an item of

Table 1. The Java program

import	**javax.swing.JApplet**	**;**			
import	**javax.swing.JLabel**	**;**			
import	**java.awt.Color**	**;**			
public	**class**	**MyProgram**	**extends**	**JApplet**	**{**
JLabel	**myLabel**	**;**			
public	**void**	**init()**	**{**		
myLabel	**=**	**new**	**JLabel ("Java")**	**;**	
getContentPane()	**.**	**setBackground (Color.yellow)**	**;**		
getContentPane()	**.**	**add(myLabel)**	**;**		
}					
}					

code in a list (learn units = 37), (4) learning the semantics of an item of code (learn units = 37) and learning the syntax by typing the item by recall (learn units = 37), (5) learning to type a line of code (learn units = 11), (6) learning to recognize a line of code in a list (learn units = 11), (7) learning the semantics of a line of code (learn units = 11) and learning the syntax by typing the line by recall (learn units = 11), and (8) writing the entire program by recall (learn units = 1). Thus, the minimum number of learn units to complete the tutor was 194. If a learner answered incorrectly at any point, the components of the learn unit were repeated iteratively until the correct answer was produced. Some learn units, such as Stage 1, only required a button click to initiate a transition. Those learn units did not iterate because the correct response was simply to follow the instruction to click the button.

Multiple-choice tests for items and lines of code were embedded in the tutor, and each question had five answer choices. For an incorrect items answer, there was a 5-sec delay or "time-out" in the tutor's interaction with the learner. For a correct items answer, a confirmation window appeared stating a general rule associated with the correct answer or an elaboration of the explanation of the meaning of the item. The lines Stage 7 had no delay interval or confirmation window. Experience suggested that most students in our courses could complete the tutor within two to three hours. The tutor transitioned automatically between stages, and students were able to take breaks between and within stages. The instructions, however, encouraged students to complete each stage before taking a break.

Questionnaires

Java software self-efficacy was assessed by requesting a rating of confidence, for each of the 23 unique items of code (*e.g.*, ***import***) in the program, in being able to use the Java item to write a program that displays a text string, as a JLabel object, in a browser window. The scale anchors were ***1 = No confidence*** to ***10 = Total confidence.*** Twelve multiple-choice questions were also administered that required applying a general concept (*i.e.*, rule) of Java object-oriented programming to solve. Appendix B presents the

12 rule questions. These 12 rule-based questions did not appear within the Java tutor, and they intended to assess far transfer or meaningful learning (Mayer, 2002). Each question had five choices, and for each question, a rating of confidence was made that the selected choice was the correct choice. The scale anchors were ***1 = Not at all confident*** to ***10 = Totally confident.*** Ratings of classification and functionality learning for eight pairs of Java symbols were obtained, as given in the online material, but they are beyond the scope of this paper. The questionnaire version that was first presented (pre-tutor questionnaire) also solicited demographic information.

The post-tutor questionnaire omitted the demographic information, and it additionally assessed evaluations of the tutor for: (1) overall effectiveness, (2) effectiveness in learning Java, and (3) usability. The anchors were ***1 =Totally negative*** to ***10 = Totally positive***. Questionnaires presented after the lecture and after the interteaching omitted evaluations of the tutor.

Procedure

Java Tutor

At the first class meeting, students completed the pre-tutor questionnaire. Students next completed the Web-based Java tutor. The tutor taught a JApplet program that displays a text string, as a JLabel object, in a browser window on the Web. The Java code and a brief description of the eight stages of the tutor are presented as part of the open source material. When a student finished the tutor, he or she completed a post-tutor questionnaire, which duplicated the software self-efficacy ratings and multiple-choice rule questions and confidence ratings. The student next accessed a set of questions and guidelines (Appendix C), posted on Blackboard, that were to be used to structure the collaborative peer tutoring session during a subsequent class. This material also presented a link to access the textual explanations of the items and lines of code presented in the Java tutor. The instructions with this material indicated that the questions presented were eligible to appear on a quiz.

Lecture

At the second class meeting, the instructor (HHE) gave a lecture on the program taught in the Java tutor. The students wrote the code in a Unix™ text editor during the lecture, which repeated the information presented in the tutor. The students were also taught the HTML file, used to access the Java bytecode file, as a URL on the Web. Support was provided so all students successfully ran the JApplet program at the conclusion of this lecture-based exercise.

This lecture required approximately one hour to complete, and the remaining class time was spent on the next unit of material, which related to the life cycle of an Applet. Students were encouraged to help each other during the subsequent classes in the semester, which combined lectures and hands-on demonstrations, with the understanding that files were not to be copied without prior permission of the instructor.

Interteaching

At the third class meeting, a collaborative peer tutoring session occurred based upon the dyadic "interteaching" model (Boyce & Hineline, 2002). Students formed dyads on their own for the session, which lasted one hour. If there were an odd number of students, one three-person group was formed. The assignment was for the students to discuss the set of questions and guidelines made available at the conclusion of the Java tutor work undertaken at the first class meeting. Also presented was the questionnaire, to include the rule questions, and students were encouraged to discuss the questions together prior to answering individually. The interteaching questionnaire instructions stated that the 12 rule questions were eligible to appear

on a quiz, but the remaining items were there only to assess instructional effectiveness of the interteaching session. The interteaching questionnaire also requested ratings of the effectiveness of the session for: (1) learning the material and (2) readiness to be tested on the material, where ***1 = Not effective*** to ***10 = Totally effective***.

During the interteaching session, students also had access to a hypertext version of the Java program that returned the textual frames of information that were embedded within the tutor[3]. These, then, were the major innovations in the current study: (1) providing the opportunity for students to discuss the rule questions together, (2) and providing direct access to information embedded within the Java tutor. During the interteaching session, students posted questions on a Blackboard discussion forum, and the instructor provided feedback.

Later on that day as the interteaching session, the instructor posted an announcement on Blackboard giving the single question that was answered incorrectly by two of the students in the summer 2006 class. The announcement was as follows: "Some students answered 'c' below for this question (also presented in the announcement). The 'c' choice is not correct because JScrollPane is a class, not an object. An object name begins with a lowercase letter. If you have a question about this, please send me email." All student inquiries were answered privately in a way to promote understanding of the principle involved. The correct answer was not given.

For the fall 2006 class, nine of the 14 students made at least one incorrect choice on the rule questions, and 11 of the 12 questions were answered incorrectly across these students. Accordingly, later on the same day as the interteaching session for this class, these 11 questions were posted on Blackboard along with the correct answer. Students' inquiries about these questions and answers could be posted on an anonymous Discussion forum on Blackboard.

The two approaches to providing feedback were based upon our intention to facilitate optimal learning in relationship to the students' performances observed within and between classes. The tactic was adjusted in accordance with our perceived needs of the students as they pursued mastery of this challenging material. This tactic is consistent with design-based research (Wang & Hannafin, 2005) as a method to improve instructional effectiveness and student performance over successive offerings of a course. In all cases, the instructor bears responsibility for providing what are considered optimal tools of learning for the students.

Graded Quiz

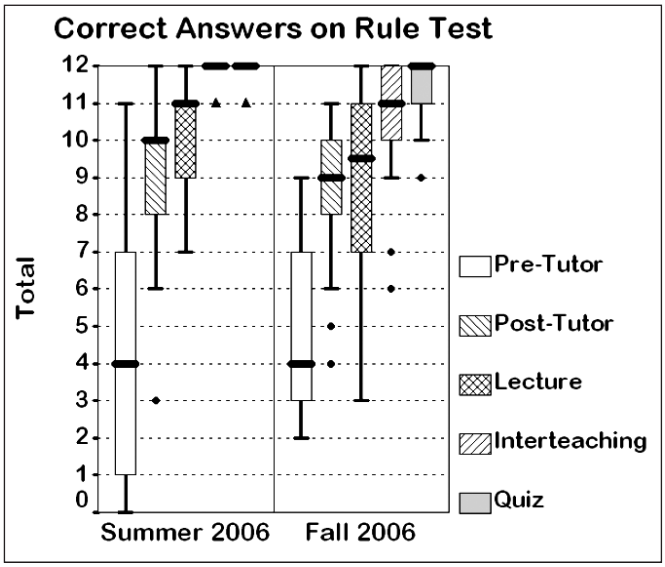
At the fourth class meeting, a quiz was administered that included questions embedded within the Java tutor and the 12 rule questions as indicated above. The graded quiz did not include any rating assessments.

RESULTS

Figure 1 presents boxplots of correct answers on the rule test over the five assessment occasions for students in the summer 2006, and fall 2006 classes. For each of the 12 questions answered during the pre-tutor assessment, one student in the summer 2006 class did not select any answer, but instead indicated being unprepared to answer. The figure shows graphically that the median total correct answers increased over the first four occasions and reached the ceiling of 12 on the interteaching occasion for the summer 2006 students and on the quiz occasion for the fall 2006 students.

For the summer 2006 students, a Friedman test (Conover, 1971, p. 264) was significant (Chi-Square = 42.259, p = 0.000). The figure shows that the greatest change for these students occurred between the pre-tutor and post-tutor occasions,

Figure 1. Boxplots of total correct answers on the rule test for students in the summer, 2006, and fall, 2006 classes across the five assessment occasions. Circles are outliers and triangles are extreme values.



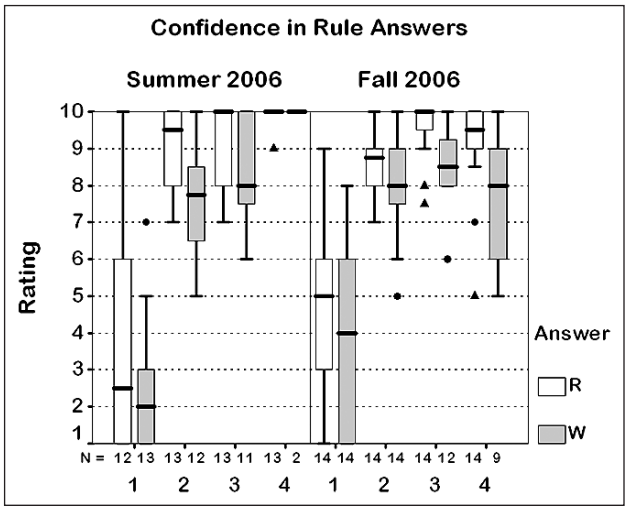
and both medians were 12 for the interteaching and quiz occasions. A Welch test, based on the differences, Di, in correct answers between successive pairs[4] of occasions over the five occasions, was significant (W = 10.889, p = 0.000). Planned pairwise comparisons were significant[5] for D1 and D2 (W = 10.145, p = 0.005), not significant for D2 and D3 (W = 1.513, p = 0.231), and significant for D3 and D4 (W = 12.295, p = 0.003).

For the fall 2006 students, a Friedman test was significant (Chi-Square = 44.000, p = 0.000). A Welch test based on the differences, Di, in correct answers between successive pairs of occasions over the five occasions, was significant (W = 8.950, p = 0.000). Planned pairwise comparisons were significant for D1 and D2 (W = 24.870, p = 0.000), not significant for D2 and D3 (W = 1.125, p = 0.301), and not significant for D3 and D4 (W = 0.207, p = 0.654).

The improvement process was somewhat different between the two classes, but the outcome for both classes reached the intended ceiling for the quiz, at least with respect to the median. With respect to individual student performance on the quiz, in the summer 2006 class two students made one error on the rule test. In the fall 2006 class, two students made one error, one student made two errors, and one student made three errors.

Figure 2 presents boxplots, over four successive occasions, of the ratings made by the students regarding confidence that the selected answer on the rule test was correct for answers that were right (R) and for answers that were wrong (W). Ratings were not obtained during the graded quiz. The number below each boxplot reflects the number of students who answered right and/or wrong over the four assessment occasions, and that is the reason that the frequency for a boxplot is sometimes less than 13 or 14 (*e.g.*, number of students giving incorrect answers for the interteaching occasion). The Welch robust test was used for both classes because of unequal sample

Figure 2. Boxplots of confidence ratings in the correctness of the rule test answers for students in the summer, 2006, and fall, 2006 classes across the four assessment occasions: 1 = Pre-Tutor, 2 = Post-Tutor, 3 = Lecture, and 4 = Interteaching. The scale anchors were ***1 = No confidence*** *to* ***10 = Total confidence****. The figure shows ratings for answers that were right (R) and for answers that were wrong (W). The N reflects the total number of students who answered correctly and/or incorrectly across the assessment occasions. Circles are outliers and triangles are extreme values.*



sizes, although the summer 2006 class did show all 14 students consistently making correct answers across the four assessment occasions.

For the summer 2006 students, the Welch test was significant for right answers (W = 16.632, p = 0.000) and for wrong answers (W = 40.864, p = 0.000). The latter test was based on the first three occasions because the variance for the interteaching occasion was zero. For right answers, planned pairwise comparisons were significant for pre-tutor and post-tutor (W = 27.398, p = 0.000), not significant for post-tutor and lecture (W = 0.108, p = 0.745), and not significant for lecture and interteaching (W = 4.959, p = 0.044) occasions. For wrong answers, planned pairwise comparisons were significant for pre-tutor and post-tutor (W = 55.646, p = 0.000) and not significant for post-tutor and lecture (W = 1.220, p = 0.282) occasions. An overall comparison of confidence ratings between right and wrong answers was significant (W = 9.481, p = 0.003).

For the fall 2006 students, the Welch test was significant for right answers (W = 16.231, p = 0.000) and for wrong answers (W = 13.477, p = 0.000). For right answers, planned pairwise, comparisons were significant for pre-tutor and post-tutor (W = 27.955, p = 0.000), significant for post-tutor and lecture (W = 9.512, p = 0.005), and not significant for lecture and interteaching (W = 1.265, p = 0.274) occasions. For wrong answers, planned pairwise, comparisons were significant for pre-tutor and post-tutor (W = 29.141, p = 0.000) not significant for post-tutor and lecture (W = 2.009, p = 0.169), and not significant for lecture and interteaching (W = 1.943, p = 0.190) occasions. An overall comparison of confidence ratings between right and wrong answers was significant (W = 4.690, p = 0.033).

Figure 3. Boxplots of ratings of the interteaching session for students in the summer, 2006, and fall, 2006 classes. Ratings were obtained for effectiveness of the session in understanding the material and for confidence in being tested on the material. The scale anchors were 1 = Lowest effectiveness or confidence to 10 = Highest effectiveness or confidence. The circle is an outlier and the triangle is an extreme value.

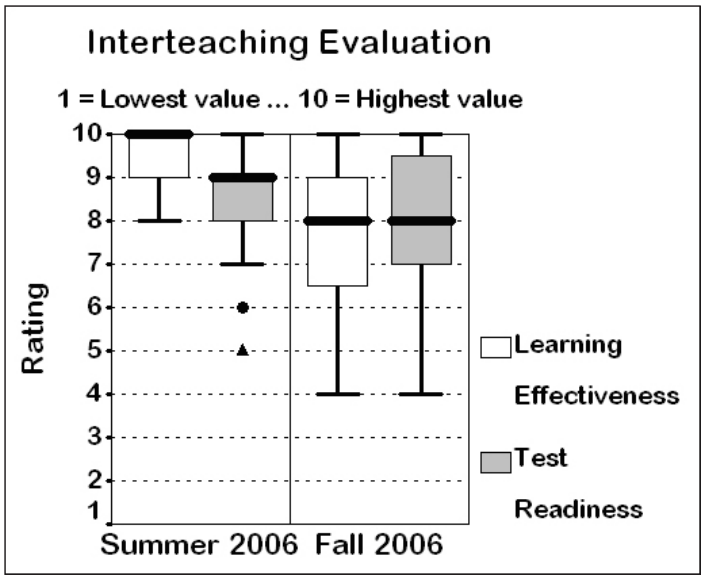
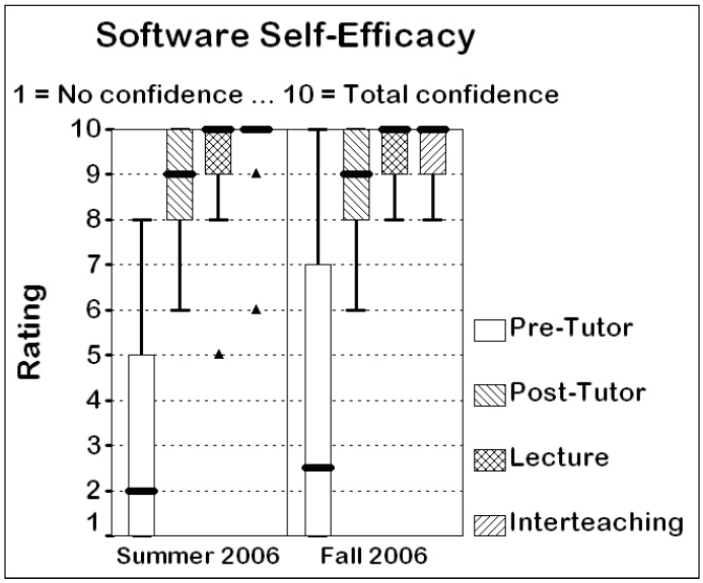


Figure 4. Boxplots of ratings of software self-efficacy for students in the Summer 2006 and Fall 2006 classes across the four assessment occasions. The ratings are based on the 23 unique items of code in the program. The scale anchors were 1 = No confidence to 10 = Total confidence. The triangles are extreme values.

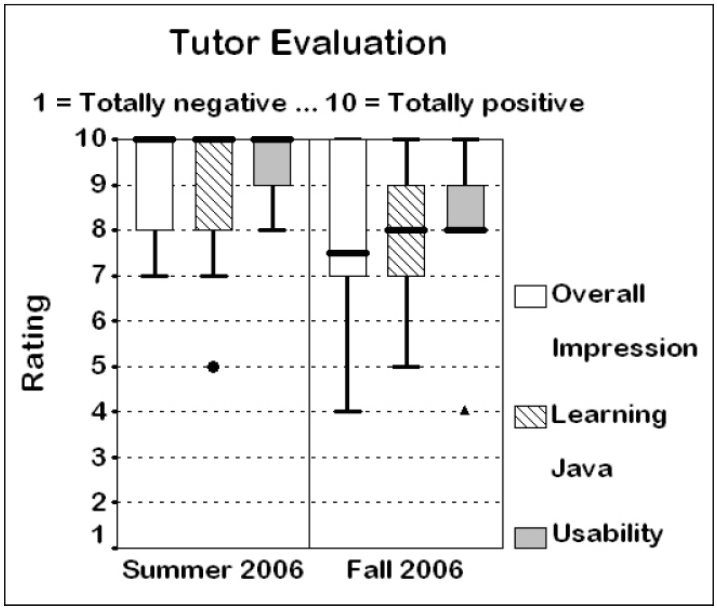


For both classes, confidence generally increased over the assessment occasions, reaching the ceiling for correct answers after the lecture. However, confidence increased for both correct and incorrect answers, although an overall comparison favored the correct answer choices.

Figure 3 presents boxplots of ratings on the interteaching evaluation, which was administered at the conclusion of the interteaching session, for students in both classes. Only 11 of the 14 students in the fall 2006 class provided an evaluation, although all 14 students participated in the interteaching session. The figure shows graphically the students' reported value in the interteaching session even when it occurred after using the Java tutor and after running the program on the Web. For the summer 2006 students, the median rating of learning impact reached the scale's ceiling of 10, with eight being the lowest rating observed. The rating of test readiness was only slightly less, with a median of nine. A Friedman's test was significant (Chi-Square = 5.444, p = 0.020). For the fall 2006 students, both scales showed median ratings of eight, and a Friedman's test was not significant for this class (Chi-Square = 0.667, p = .414). Although the median ratings for the fall 2006 students were comparatively lower than the summer 2006 students, taken together, these data show that almost all students reported value in the collaborative peer tutoring even when the session followed several other instructional experiences. No rating below a value of four was observed by any student.

Figure 4 presents boxplots of software self-efficacy ratings across the first four assessment occasions for students in the summer 2006 and fall 2006 classes. These ratings were not obtained during the graded quiz. Each boxplot is based upon the median rating over the 23 unique items of code in the program for the 13 students in the

*Figure 5. Boxplots of ratings of the tutor for students in the summer, 2006, and fall, 2006 classes for three scales. The scale anchors were **1 = Totally negative** to **10 = Totally positive**. The circle is an outlier, and the triangle is an extreme value.*



summer 2006 class and for the 14 students in the fall 2006 class. For ratings across all occasions for both classes, Cronbach's alpha reliability of the ratings within each assessment exceeded 0.90, and all values were significant ($p < .05$).

For the summer 2006 class, a Friedman test was significant (Chi-Square = 32.614, p = 000). A Welch test, based on the differences in ratings between successive pairs of occasions, was significant (W = 30.222, p = 0.000). Planned pairwise, comparisons of the differences, Di, were significant for D1 and D2 (W = 60.215, p = 0.000) and not significant for D2 and D3 (W = 1.330, p = 0.260). Software self-efficacy increased over the assessment occasions, and it reached the ceiling following the lecture.

For the fall 2006 class, a Friedman test was significant (Chi-Square = 32.741, p = 000). A Welch test, based on the differences in ratings between successive pairs of occasions, was significant (W = 18.450, p = 0.000). Planned pairwise comparisons of the differences, Di, were significant for D1 and D2 (W = 29.911, p = 0.000) and not significant for D2 and D3 (W = 3.452, p = 0.075). Similar to the summer 2006 students, software self-efficacy increased over the assessment occasions, and it reached the ceiling following the Lecture.

Figure 5 presents boxplots of ratings of evaluation of the tutor taken during the post-tutor assessment for students in both classes. Ratings on the following three scales were requested: (1) overall impression of the tutor, (2) effectiveness of the tutor in learning Java, and (3) usability of the tutor interfaces. The scale anchors on each 10-point scale were ***1 = Totally negative*** to ***10 = Totally positive***. For students in the summer 2006 class, median ratings for all three scales reached the scale ceiling of ten, with only a single outlier observed for Java Learning. For students in the fall 2006 class, the medians were comparatively lower, but all medians were higher than seven. Since ordinal data are problematic for between-group comparisons, these differences will not be interpreted statistically. However, the evaluation ratings from both classes together suggest that students reported value in their use of the tutor, despite an occasional extreme value toward the lower end of a scale.

DISCUSSION

The results of this study show the value of applying several different instructional modalities in furtherance of having information systems students achieve skill and understanding with respect to a simple Java applet, presented as a first technical exercise in a semester-long course. The data support the utility of this approach as reflected in students' rule test performance and software self-efficacy, which progressively improved over the successive assessment occasions. Rehearsal is an intuitively obvious and well-researched factor in knowledge and skill acquisition (*e.g.*, Salas & Cannon-Bowers, 2001), and the present study shows how structured rehearsal may be managed using the several modalities under consideration. Principles underlying such managed skill acquisition with different instructional modalities are presented elsewhere (Fox & Hackerman, 2003; Halpern & Hakel, 2003). Finally, although the predictive influence of self-efficacy on future performance has been questioned (Heggestad & Kanfer, 2005), self-efficacy assessments continue to be viewed as an important indicator of the effectiveness of training programs that are intended to produce both skill and motivation to learn (*e.g.*, Johnson, 2005).

Despite the apparent benefits of applying different instructional modalities to support student learning, however, the research base in instructional design typically compares one modality or instructional method with others with respect to student performance assessed at only one point in time. Even the U.S. Department of Education's What Works Clearinghouse[6] favors such an approach. Related to the present study, for example, Harrington (1999) reported that graduate social-

work students with relatively high grade-point averages did not differ in final grades when a statistics course was taught either by a traditional lecture format or by "programmed instruction" in a distance learning setting. Saville, Zinn, Neef et al. (2006) reported that quiz scores for graduate and undergraduate students were higher after an interteaching session in comparison to scores observed after a lecture.

With respect to teaching computer programming to college-level students, Williams, Wiebe, Yang et al. (2002) reported that the percent of undergraduate students passing an introductory Java course was higher for a pair-programming laboratory section in comparison to students whose laboratory section involved solo programming. The benefits of collaborative learning, in comparison to solitary learning, when applied to computer programming were also shown in college students' program generation abilities using LISP-LOGO (Jehng, 1997). Although experimental designs that compare average student performances between and among conditions may have value in identifying an optimal technique to use when there is only a single and time-limited occasion to teach or to learn, such studies have little to offer in the engineering of instructional tactics when the objective is to have each individual student reach a criterion of mastery (cf Perone, 1999). Meyer (2004) questioned the value of old-fashioned experimental "horse-race" designs in another context, but the argument seems relevant within the current context as well.

This study constitutes a systematic replication (Sidman, 1960). A set of teaching tactics was revised with the expectation that student learning would be improved. The methodology reflects design-based research, which is a type of formative evaluation (Collins, Joseph, & Bielaczyc, 2004) that is emerging as an alternative methodology in support of developing and assessing improvements in instructional design within the context of the classroom (Bell, Hoadley, & Linn, 2004; Design-Based Research Collective, 2003). In that regard, the order of presenting the several instructional tactics was determined by anecdotal observations of student performance over the several classroom evaluations that were previously undertaken in this stream of work. It was decided that a hands-on lecture would benefit from students' prior rehearsal with the Java code and that collaborative peer tutoring would benefit from the cumulative learning obtained from the programmed instruction and the lecture.

Since the components in the current ordering are well received by students and since a desired learning outcome was achieved, we have the view that it is worthwhile now to direct our attention to developing advanced instructional material, rather than to "prove" the optimal ordering under conditions of a traditional "effect-size" experiment. Support for that view is implicit within designed-based research and has been discussed by educational scholars and training designers (*e.g.*, Mayer, 2004; Sackett & Mullen, 1993). Importantly, students reported value in the Java tutor and in the collaborative peer tutoring, and taken together with the lecture, these approaches to managing rehearsal in the classroom environment converge on what are increasingly recognized as vital ingredients to facilitate science education, in general (DeHaan, 2005).

The content and functionality of the Web-based programmed instruction tutoring system have been upgraded and continuously revised since the initial report (Emurian et al., 2000), and the system has been demonstrably effective and well received by our students. However, it is to be understood that other approaches to automated tutoring systems offer advantages in meeting the needs of the individual learner. For example, Butz, Hua, and Mcguire (2006) reported the application of Baysian networks to determine instructional events at the level of the individual learner in a Web-based intelligent tutoring system for computer programming. That and similar artificial intelligence (*e.g.*, Zhang, 2004) and multi-media applications (*e.g.*, Zhang, Zhou, Briggs et al., 2006)

have obvious promise in improving the capabilities of the current programmed instruction orientation to automated instructional design.

Having students discuss rule questions together may have enhanced understanding and retention in the present context as indicated in subsequent rule test performance. However, an obvious challenge for collaborative peer tutoring, in general, and for interteaching, in particular, is to insure that participating students are, indeed, teaching one another and to make certain that they are sufficiently informed to know when their solutions to questions are correct. Boyce and Hineline (2002) and Saville et al. (2006) suggest several approaches to oversee and to evaluate interteaching to assure a beneficial session, such as the awarding of "quality points" by a monitor of the session. Similar to our previous observations, however, students showed "overconfidence" in incorrect rule answers, and that issue requires exploration in the design of future work. Tactics to be explored to improve the effectiveness of interteaching include the adoption of vignettes and rubrics to facilitate higher-order thinking and academic achievement (Kish, 2006).

The list of approaches to teaching and learning computer programming continues to grow. In this article, reported techniques include (1) a "blended" instructional approach (Boyle et al., 2003); (2) an emphasis on mathematics and algorithms (Hu, 2006); (3) supportive programming environments such as BlueJ (Kolling, Quig, & Rosenberg, 2003), DrJava (Hsia, Simpson, Smith et al., 2005), and PigWorld (Lister, 2004); (4) Problem-Based Learning (Tsang & Chan, 2004); (5) the Environment for Learning to Program (Truong, Bancroft, & Roe, 2005); (6) collaborative peer tutoring (Williams et al., 2002) and collaborative learning (Jehng, 1997); (7) a Traffic Light System Simulator (Yuen, 2006); a Computer Clubhouse learning environment (McDougall & Boyle, 2004), and (9) a Web-based personalized system of instruction (Koen, 2005). With the possible exception of Boyle et al. (2003), research studies in this domain typically emphasize a student's singular exposure to a task within the context of a single instructional modality.

As an alternative to the aforementioned approaches, the instructional tactics adopted in the classroom at the start of a semester's work are based initially upon *programmed instruction*, which is a form of structured and optionally automated instruction, as discussed by Emurian and Durham (2003) and Emurian et al., (2003) with respect to teaching computer programming. They also include a *lecture* with hands-on learning. They also include *interteaching*, which is a form of collaborative peer tutoring (Boyce & Hineline, 2002). As implemented in the present context to foster repeated practice with different instructional modalities and with socially supported interactions, these tactics originated from behavior analysis. The Cambridge Center for Behavioral Studies[7] provides fundamental definitions and a wealth of information regarding the philosophical underpinnings and applications of this approach to science, in general, and to education, in particular. Finally, these tactics are to be understood as providing only an initial series of learning experiences to students in preparation for subsequent learning with other instructional and program development tools and techniques, to include the use of an integrated development environment (IDE) such as Eclipse.

Although educators might have the success of their students as a primary goal of teaching, it is less certain that what happens in the classroom is based on empirical evidence of effectiveness: a rational pedagogy (Emurian, 2001). In addition, it is sometimes the case that expecting students prematurely to solve general computer programming problems and to understand complex control structures and algorithms neglects the skills that students must possess to undertake such higher-order learning. Too often, perhaps, educators may view an introductory course in science, engineering, and mathematics (STEM) as an occasion to eliminate marginally prepared students rather

than as an opportunity to teach them the skills necessary to succeed. Although we also have the goal of helping students to learn the syntax and semantics of advanced programming such as recursion, we argue that our approach is deliberately and constructively designed to meet the needs of novice students, those *ineffective novices* who lack experience and self-efficacy in this domain (Robins, Rountree, & Rountree, 2003).

In furtherance of providing those skills to our students, techniques derived from behavior analysis have been demonstrably effective in promoting skill, confidence, and meaningful learning by novitiate students regarding an object-oriented programming language. Behavior analysis is one promising approach in identifying the ontogenetic instructional learn units (Greer & McDonough, 1999) whose mastery provides the textual tools essential for advanced understanding, thinking, and problem solving in the domain of computer programming. Teachers facing the difficult challenge of providing effective instruction to the diversity of students who enroll in introductory computer programming courses need to be mindful of all approaches to helping their students succeed. The present study represents a reconfirmation of one set of instructional tactics that are effective for information systems students and well received by them. All students deserve to have access to such evidenced-based tactics.

REFERENCES

Anderson, J.R., Corbett, A.T., Koedinger, K.R., & Pelletier, R. (1995). Cognitive tutors: Lessons learned. *Journal of the Learning Sciences*, *4*(2), 167-207.

Barnett, S.M., & Ceci, S.J. (2002). When and where do we apply what we learn? A taxonomy for far transfer. *Psychological Bulletin*, *128*, 612-637.

Bell, P., Hoadley, C.M., & Linn, M.C. (2004). Design-based research in education. In M.C. Linn, E.A. Davis, & P. Bell (Eds.). *Internet environments for science education* (pp. 73-88). Laurence Erlbaum Associates.

Boyce, T.E., & Hineline, P.N. (2002). Interteaching: A strategy for enhancing the user-friendliness of behavioral arrangements in the college classroom. *The Behavior Analyst*, *25*, 215-226.

Boyle, T., Bradley, C., Chalk, P., Jones, R., & Pickard, P. (2003). Using blended learning to improve student success rates in learning to program. *Journal of Educational Media*, *28*(2-3), 165-178.

Butz, C.J., Hua, S., & Mcguire, R.B. (2006). A web-based bayesian intelligent tutoring system for computer programming. *Web Intelligence & Agent Systems*, *4*(1), 61-67.

Collins, A., Joseph, D., & Bielaczyc, K. (2004). Design research: Theoretical and methodological issues. *Journal of the Learning Sciences*, *13*(1), 15-42.

Conover, W.J. (1971). *Practical nonparametric statistics.* New York, NY: John Wiley & Sons, Inc.

DeHaan, R.L. (2005). The impending revolution in undergraduate science education. *Journal of Science Education and Technology*, *14*(2), 253-269.

Design-Based Research Collective (2003). *Educational Researcher*, *32*(1), 5-8.

Emurian, H.H. (2001). The consequences of e-learning (Editorial), *Information Resources Management Journal*, April-June, 3-5.

Emurian, H.H. (2004). A programmed instruction tutoring system for Java: Consideration of learning performance and software self-efficacy. *Computers in Human Behavior*, *20*(3), 423-459.

Emurian, H.H. (2005). Web-based programmed instruction: Evidence of rule-governed learning. *Computers in Human Behavior*, *21*(6), 893-915.

Emurian, H.H. (2006a). A Web-based tutor for Java™: Evidence of meaningful learning. *Journal of Distance Education Technologies*, *4*(2), 10-30.

Emurian, H.H. (2006b). Assessing the effectiveness of programmed instruction and collaborative peer tutoring in teaching Java™. *International Journal of Information and Communication Technology Education*, *2*(2), 1-16.

Emurian, H.H. (in press:a). Applications of behavior analysis to ICT education: Teaching java with programmed instruction and interteaching. *Encyclopedia of Information Technology Curriculum Integration*. Hershey, PA: IRM Press.

Emurian, H.H. (in press:b). Managing programmed instruction and collaborative peer tutoring in the classroom: Applications in teaching Java™. *Computers in Human Behavior.*

Emurian, H.H., & Durham, A.G. (2003). Computer-based tutoring systems: A behavioral approach. In J.A. Jacko & A. Sears (Eds.), *Handbook of human-computer interaction* (pp. 677-697). Mahwah, NJ: Lawrence Erlbaum & Associates.

Emurian, H.H., Wang, J., & Durham, A.G. (2003). Analysis of learner performance on a tutoring system for Java. In T. McGill (Ed.), *Current Issues in IT Education* (pp. 46-76). Hershey, PA: IRM Press.

Emurian, H.H., Hu, X., Wang, J., & Durham, A.G. (2000). Learning Java: A programmed instruction approach using applets. *Computers in Human Behavior*, *16*, 395-422.

Fox, M.A., & Hackerman, N. (2003). *Evaluating and improving undergraduate teaching in science, technology, engineering, and mathematics.* Washington, DC: The National Academies of Science Press.

Greer, R.D. (2002). *Designing teaching strategies: An applied behavior analysis systems approach*. New York: Academic Press.

Greer, R.D., & McDonough, S.H. (1999). Is the learn unit a fundamental measure of pedagogy? *The Behavior Analyst*, *22*, 5-16.

Halpern, D.F., & Hakel, M.F. (2003). Applying the science of learning to the university and beyond: Teaching for long-term retention and transfer. *Change*, *35*(4), 37-41.

Harrington, D. (1999). Teaching statistics: A comparison of traditional classroom and programmed instruction/distance learning approaches. *Journal of Social Work Education*, *35*(3), 343-352.

Heggestad, E.D., & Kanfer, R. (2005). The predictive validity of self-efficacy in training performance: Little more than past performance. *Journal of Experimental Psychology: Applied*, *11*(2), 84-97.

Hsia, J.I., Simpson, E., Smith, D., & Cartwright, R. (2005, February 23-27). Taming Java for the classroom. *SIGCSE'05,* St. Louis, MI, 327-331.

Hu, C. (2006). It's mathematical after all: The nature of learning computer programming. *Education and Information Technologies*, *11*(1), 83-92.

Ingvarsson, E.T., & Hanley, G.P. (2006). An evaluation of computer-based programmed instruction for promoting teachers' greeting of parents by name. *Journal of Applied Behavior Analysis*, *39*(2), 203-214.

Jehng, J-C. J. (1997). The psycho-social processes and cognitive effects of peer-based collaborative interactions with computers. *Journal of Educational Computing Research*, *17*(1), 19-46.

Johnson, R.D. (2005). An empirical investigation of sources of application-specific computer-self-efficacy and mediators of the efficacy-performance relationship. *International Journal of Human-Computer Studies*, *62*(6), 737-758.

Kirschner, P.A., Sweller, J., & Clark, R.E. (2006). Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching. *Educational Psychologist*, *41*(2), 75-86.

Kish, M.H.Z. (2006). Overview of using vignettes to develop higher order thinking and academic achievement in adult learners in an online learning environment. *International Journal of Information and Communication Technology Education*, *2*(3), 60-74.

Koen, B.V. (2005). Creating a sense of "presence" in a web-based PSI course: The search for Mark Hopkins' log in a digital log. *IEEE Transactions on Education*, *48*(4), 599-604.

Kolling, M., Quig, B., Patterson, A., & Rosenberg, J. (2003). The BlueJ system and its pedagogy. *Journal of Computer Science Education*, *14*(Dec), 1-12.

Kurbanoglu, N.I., Taskesenligil, Y., & Sozbilir, M. (2006). Programmed instruction revisited: A study on teaching stereochemistry. *Chemistry Education Research and Practice*, 7(1), 13-21.

Lister, R. (2004). Teaching java first: Experiments with a pigs-early pedagogy. *Proceedings of the 6th Conference on Australian Computing Education* Vol. 30 (pp. 177-183), Dunedin: Australian Computer Society, Inc.

Mayer, R.E. (2002). *The promise of educational psychology. Volume II. Teaching for meaningful learning.* Upper Saddle River, NJ: Pearson Education, Inc.

Mayer, R.E. (2004). Should there be a three-strikes rule against pure discovery learning? *American Psychologist*, *59*(1), 14-19.

Maxwell, S.E., & Delaney, H.D. (2004). *Designing experiments and analyzing data: A model comparison perspective (2nd Ed)*. Mahwah, NJ: Lawrence Erlbaum Associates.

McDougall, A., & Boyle, M. (2004). Student strategies for learning computer programming: Implications for pedagogy in informatics. *Education and Information Technologies*, *9*(2), 109-116.

Perone, M. (1999). Statistical inference in behavior analysis: Experimental control is better. *The Behavior Analyst*, *22*(2), 109-116.

Robins, A., Rountree, J., Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education*, *13*(2), 137-172.

Sackett, R.R., & Mullen, E.J. (1993). Beyond formal experimental design: Towards an expanded view of the training evaluation process. *Personnel Psychology*, *46*, 613-627.

Salas, E., & Cannon-Bowers, J.A. (2001). The science of training: A decade of progress. *Annual Review of Psychology*, *52*, 471-499.

Saville, B.K., Zinn, T.E., Neef, N.A., Norman, R.V., & Ferreri, S.J. (2006). A comparison of interteaching and lecture in the college classroom. *Journal of Applied Behavior Analysis*, 39, 49-61.

Sidman, M. (1960). *Tactics of scientific research*. New York: Basic Books.

Truong, N., Bancroft, P., & Roe, P. (2005, June 27–29). Learning to program through the web. *Proceedings of the 10th annual SIGSCE conference on innovation and technology in computer science education*, ITiCSE'05. Monte de Caparica, Portugal: ACM Press.

Tsang, A.C.W., & Chan, N. (2004). An online problem-based model for the learning of java. *Journal of Electronic Commerce in Organizations*, *2*(2), 55-64.

Wang, F., & Hannafin, M.J. (2005). Design-based research and technology-enhanced learning environments. *Educational Technology Research and Development*, *55*(4), 5-24.

Williams, L.A., Wiebe, E., Yang, K., Ferzli, M., & Miller, C. (2002). In support of pair programming in the introductory computer science course. *Computer Science Education*, *12*(3), 197-212.

Yuen, A.H.K. (2006). Learning to program through interactive simulation. *Educational Media International*, *43*(3), 251-268.

Zhang, D. (2004). Virtual mentor and the LBA system—towards building an interactive, personalized, and intelligent e-learning environment. *Journal of Computer Information Systems, XLIV*, (3), 35-43.

Zhang, D, Zhou, L., Briggs, B., & Nunamaker, J. F. (2006). Instructional video in e-learning: Assessing the impact of interactive video on learning effectiveness. *Information & Management*. *43*(1), pp. 15-27.

ENDNOTES

1. A portion of the summer, 2006, data was accepted for presentation at the 2007 convention of the Information Resources Management Association.
2. All materials used in this study are freely available. They include the online Java tutor, the open source code for the tutor, the course materials, and all assessment instruments: http://nasa1.ifsm.umbc.edu/learnJava/tutorLinks/TutorLinks.html
3. http://userpages.umbc.edu/~emurian/learnJava/swing/tutor/v2/explanations/Explanations.html
4. In the present study, the difference values for respective assessment variables were computed as follows: D1 = (Post-Tutor – Pre-Tutor); D2 = (Lecture – Post-Tutor); D3 = (Interteaching – Lecture); and D4 = (Quiz – Interteaching). The Welch test applied to these differences is similar to the multivariate approach for within-subjects designs recommended by Maxwell and Delaney (2004, p. 624). Planned pairwise comparisons were to detect possible differences in effect magnitude over the successive conditions.
5. To control for the experimentwise error rate, the significant *p* value for each planned comparison must be less than 0.05/number-of-planned-comparisons.
6. http://www.whatworks.ed.gov/
7. http://www.behavior.org/index.cfm