Int'l J. of Information and Communication Technology Education, 2(2), 1-16, April-June 2006 1

Assessing the Effectiveness of Programmed Instruction and Collaborative Peer Tutoring in Teaching Java™

Henry H. Emurian, University of Maryland, Baltimore County, USA

ABSTRACT

Students in two Java programming classes completed an individualized tutoring system that taught a simple applet program. Before and after using the tutor, students completed questionnaires that assessed software self-efficacy and understanding of general programming principles. The questionnaires also were administered following a lecture session on the program that included having the students successfully run the applet in a browser on the Web. For the second class, a collaborative peer tutoring session based on the applet program occurred between completion of the tutor and the lecture session. Students in both classes increased in software self-efficacy and program understanding across the assessment occasions. For students in the second class, correct answers on the final test of understanding were higher than observed in the first class. Collaborative peer tutoring used in combination with a programmed instruction tutoring system may potentiate learning for novitiate students.

Keywords: collaborative peer tutoring; interteaching; Java training; programmed instruction

INTRODUCTION

Teaching computer programming is not easy (Traynor & Gibson, 2004). Reviews of the instructional literature indicate that many students struggle with their programming courses (Robins, Rountree, & Rountree, 2003), and the complexity and instability of Java pose unique challenges to both educators and students (Roberts, 2004). The research reported here, then, reflects an attempt to improve the effectiveness of Java programming instruction, as evidenced by enhanced student performance, which was accomplished by com-

bining an individualized tutoring system with collaborative peer tutoring. This tactic of teaching students with novel instructional approaches is fundamental to success-oriented classroom strategies, which are mindful of the alarming dropout rates reported for students in many science, technology, engineering, and mathematics (STEM) programs of study (Wormley, 2003).

Tutoring System

The research group previously reported a series of evaluations in the development of a Web-based tutoring system[1] and its classroom application as the first technical training exercise for students in a Java computer programming course (Emurian, 2004, 2005, in press; Emurian & Durham, 2001, 2002, 2003; Emurian, Hu, Wang, & Durham, 2000; Emurian, Wang, & Durham, 2003). The tutor teaches a simple 32-item, 10-line Java applet that displays a Label object in a browser window on the Web.[2] The learning theory supporting the development of the tutoring system is a behavior analysis model based upon the learn unit formulation (Greer & McDonough, 1999; Singer-Dudek & Greer, 2005) as applied to programmed instruction for technology education (Greer, 2002).

The objective of the tutor is to provide each and every student with elementary knowledge and skill in preparation for continuing study of the Java programming language. The tutoring system is effective in promoting skill and cultivating self-confidence in beginning students by giving them a successful learning experience that motivates their further study of Java using textbooks, lectures, laboratory demonstrations, independent problem solving, and the like. The tutoring system is intended to meet the needs of Information Systems majors, whose professional interests typically are outside of the scope of computer programming.

Dyadic Collaboration

Although group discussions involving three or more participants may have value in facilitating learning to write computer programs (Davy & Jenkins, 1999), having students study together in dyads recently has been investigated as an even more powerful tactic to improve learning at the level of individuals. There are several variations to the structure of a dyadic interaction, which include reciprocal peer tutoring (Griffin & Griffin, 1998), pair programming (Williams et al., 2000), and interteaching (Boyce & Hineline, 2002). The variant of collaborative peer tutoring that was adopted in the present study is the interteaching dialog, which is a mutually probing, mutually informing conversation between two people (Boyce & Hineline, 2002). Interteaching has the objective of insuring by the participants as a team that each member of a dyad can answer a previously disclosed set of questions. This approach is similar to the peer collaboration paradigm to teach recursion that was studied by Jehng (1997). It is suggested here that an interteaching session following individualized tutoring can potentiate the prior learning and result in enhanced competency and understanding.

Background and Rationale

Our previous research (Emurian, 2005, in press) showed that students who completed the Java tutoring system learned general rules of Java programming that could be applied to answer questions on problems not explicitly presented in the tutor itself. These findings supported the value of the tutor to produce meaningful learning

(Mayer, 2002) or far transfer of learning (Barnett & Ceci, 2002), indicating that informed students could apply general rules to solve novel problems. The research methodology is similar to design-based research (Brown, 1992; Design-Based Research Collective, 2003; Hoadley, 2004) in that instructional design effectiveness was assessed within the context of the classroom over several successive semesters. In assessing meaningful learning over the semesters, the number of rule-based questions was increased, and the opportunity to evaluate tutor effectiveness with several different groups of students showed the reliability and generality of the tutor's application. These outcomes were encouraging, but the magnitude of the learning effects assessed immediately after students completed the tutor and after a final lecture on the material left room for improvement, especially when the number of rule-based questions was increased from four (Emurian, 2005) to 10 (Emurian, in press). The present study intends to potentiate the learning effects by combining programmed instruction and interteaching in the classroom.

The research reported here is based upon two successive offerings of an elective course entitled Graphical User Interface Systems Using Java.[3] The first class, offered during the summer of 2004, consisted of master's degree students, and the second class, offered during the fall of 2004, consisted of advanced undergraduate students. The content and objectives of the course were equivalent for both classes. Students in the first class completed the Java tutor only, and students in the second class completed the Java tutor and an interteaching dialog. In comparison to the number of rule questions administered to students in the Emurian (in press) study, the number of rule questions was increased in the present study to 12 for students in both classes. The rationale for adopting this design-based approach to compare learning outcomes between two successive classes will be discussed.

METHOD

Materials

All questionnaires and study materials are available as documents on the Web.[4] Appendix A presents the interteaching report. An example of one of the 12 questions on the rules test is as follows:

Which of the following sequences is correct?

a. Declare a TextField object, construct a TextField object, add a TextField object to a container.
b. Construct a TextField object, declare a TextField object, add a TextField object to a container.
c. Declare a TextField object, add a TextField object to a container, construct a TextField object.
d. Add a TextField object to a container, declare a TextField object, construct a TextField object.

Java Tutor

Tutor figures are presented in Emurian (in press). The learning stages in the tutor are as follows:

1. **Introduction.** This stage gives an orientation to the tutor and how it works. It also gives an example of the applet running in a browser window.
2. **Item Familiarity.** This stage teaches the symbols to be used. The student cop-

ies each identifier, keyword, or separator, displayed one at a time, into a text field. This stage is invaluable to ineffective novices (Robins et al., 2003) who deserve the opportunity to be prepared adequately for the learning that follows. The fact that students do make errors on this tutor stage is demonstrated in Emurian (2004).

3. **Item Identification.** This stage teaches the student to recognize differences among the 21 unique symbols. This is accomplished by having the student highlight a displayed symbol from a list of all symbols.
4. **Item Learning.** This is the primary stage for learning the semantics of all items in the code and learning to enter the item at the correct sequential location in the program. First, the item is displayed in a highlighted fashion in a text area at the location in which it will appear in the code. Second, a textual display presents the meaning of the item at that point in the code. Third, a multiple-choice test is presented on the meaning of the specific item. If the answer is incorrect, the explanation is presented again, and the explanation-test cycle is repeated until the test is answered correctly. Fourth, the student types the item into a text field that is displayed in the proper location in the code. If the item is entered incorrectly, the explanation-test cycle is repeated until the item is entered correctly. When the item is entered correctly, the tutor progresses to the next item. As individual items are entered into the text field correctly, the developing program displays cumulatively.
5. **Line Familiarity.** This stage is functionally similar to Stage 2, but the unit of learning is a line rather than an item.
6. **Line Identification.** This stage is functionally similar to Stage 3, but the unit of learning is a line rather than an item.
7. **Line Learning.** This stage is functionally similar to Stage 4. The size of the learn unit, however, is at the level of a line rather than an item.
8. **Program Learning.** This is the last stage in the tutor. This stage requires entering the entire program into a text window. The input is repeated until the code is entered correctly. The learner is able to view the correct program, but selection of that option clears the input window so that the code has to be entered again from memory. The purpose of this stage is to solidify the organizational structure of learning into a manageable sequential stream or a singular unit in which the components have been previously networked into smaller units that can be combined within a new context under conditions showing meaningful learning.

Interteaching

For the interteaching sessions, subjects were paired unsystematically, and a listing of team partners was posted prior to the class meeting. The interteaching reports were available to be downloaded from the course Blackboard site. Subjects were informed to follow the interteaching instructions and to discuss the material together for 30 minutes. At the conclusion of the interteaching session, each team member completed the interteaching evaluation section. Appendix A presents the interteaching report relevant to the present article. The remaining three reports were similar in that they all had questions that could appear on a quiz for the students to discuss.

Subjects

There were 14 students in each of two classes titled Graphical User Interface Systems Using Java. This is an elective course for advanced undergraduate students and for master's degree students. Although the objectives, performance requirements, and prerequisites (one prior programming course) are identical, the classes were offered separately. In this study, the graduate class, identified as the Tutor (T) class, met in the summer of 2004. The undergraduate class, identified as the Tutor-plus-Interteaching (T+I) class, met in the fall of 2004. Background data were collected during the first questionnaire administration (Pre-Tutor Questionnaire). The T class had six female and eight male students, and the T+I class had one female and 13 male students (chi-square = 4.76, df = 1, $p < .05$). The protocols, to be presented next, were exempt from informed consent requirements, because they reflected instructional practices in the classroom.

Experience ratings were based on a 10-point scale where 1 = Novice to 10 = Expert. Comparisons between the two classes were based on the Kruskal-Wallis test. Median ratings were as follows: Java experience (T = 3, T+I = 1.5; chi-square = 3.51, $p > .05$); general programming experience (T = 4, T+I = 5.5; chi-square = 3.34, $p > .05$); and total number of prior programming courses taken (T = 3, T+I = 4; chi-square = 2.31, $p > .10$). The median age of the students was 26 years for the T class and 25 years for the T+I class (chi-square = 1.85, $p > .10$). Although the students were undergraduates in one class and graduates in the other, the evidence did not support differences between the classes on these measures. However, the gender composition between the classes differed, with proportionally more female students in the summer graduate class in comparison to the fall undergraduate class.

Procedure

Table 1 presents the sequence of events for each of the two classes. The summer 2004 class met twice each week for six weeks. The fall 2004 class met once each week for 14 weeks. All classes met for 2.5 hours. The students were informed fully about the requirements of each class, and the sequence of events was included on the syllabus. At the first meeting, students in the fall 2004 class were informed that the rules questions would appear later as part of a graded quiz. All students completed the tutor during Session 1, the first class period. After Session 1 for the fall 2004 students, a study manual was released that duplicated the instructional text within the tutor but omitted the multiple choice tests that were embedded within the tutor. The study manual did not present the 12 rules multiple-choice questions. Students were informed that the manual could be used to prepare for the interteaching on Session 2.

The sequence of events was a compromise that allowed student behavior to be evaluated within the context of a classroom. The justification for such a design-based research approach, together with its strengths and limitations, will be discussed.

RESULTS

Figure 1 presents boxplots of total correct answers on the rules test across the three occasions for both classes. For the T class, a Friedman's test was significant (chi-square = 18.84, df = 2, $p < .001$). Pairwise contrasts, Bonferroni corrected, were not significant only between the Post-

Table 1. Sequence of events

	Summer 2004 Master's Degree Students n = 14	**Fall 2004** Advanced Undergraduates n = 14
2.5 Hours	**Tutor** Questionnaires: SSE & Rules	**Tutor + Interteaching** Questionnaires: SSE & Rules
Session 1	1. Pre-Tutor Questionnaires 2. Tutor 3. Post-Tutor Questionnaires	1. Pre-Tutor Questionnaires 2. Tutor
		Access to Study Manual
Session 2	1. Lecture 2. Run the Program 3. Final Questionnaires	1. Post-Tutor Questionnaires 2. Interteaching 3. Lecture 4. Run the Program
Session 3		1. Final Questionnaires • Test Credit for Rules

Tutor and Final occasions ($p > .05$). For the T+I class, a Friedman's test was significant (chi-square = 23.57, df = 2, $p < .001$). Pairwise contrasts, Bonferroni corrected, were significant for all pairs ($p < .05$). Kruskal-Wallis tests showed no significant difference between the classes for the Pre-Tutor test (chi-square = 0.24, df = 1, $p > .50$) and the Post-Tutor test (chi-square = 0.00, df = 1, $p > .50$). The difference for the Final test was significant (chi-square = 8.40, df = 1, $p < .005$).

For each of the 12 questions on the rules test, the subject was asked to rate the confidence that the correct answer had been selected among the four alternatives. Figure 2 presents confidence ratings for correct and incorrect answers for the T and T+I classes across the three assessment occasions. The boxplots were calculated from the set of median confidence ratings for correct and incorrect answers for all subjects. Where there are fewer than 14 subjects for a given boxplot, this indicates that not all subjects made at least one incorrect answer on that occasion.[5]

For the T class, a Friedman's test for confidence ratings of correct answers across the three occasions was significant (chi-square = 19.24, df = 2, $p < .001$). Pairwise contrasts, Bonferroni corrected, were significant for all pairs ($p < .05$) except between the Post-Tutor and Final occasions ($p > .05$). A Friedman's test for confidence ratings of incorrect answers across the three occasions was also significant (chi-square = 21.23, df = 2, $p < .001$). Pairwise contrasts, Bonferroni corrected, were significant for all pairs ($p < .05$) except between the Post-Tutor and Final occasions ($p > .05$). A Friedman's test between confidence ratings for correct and

Figure 1. Boxplots of total correct answers on the rules test (circles are outliers and triangles are extreme values)

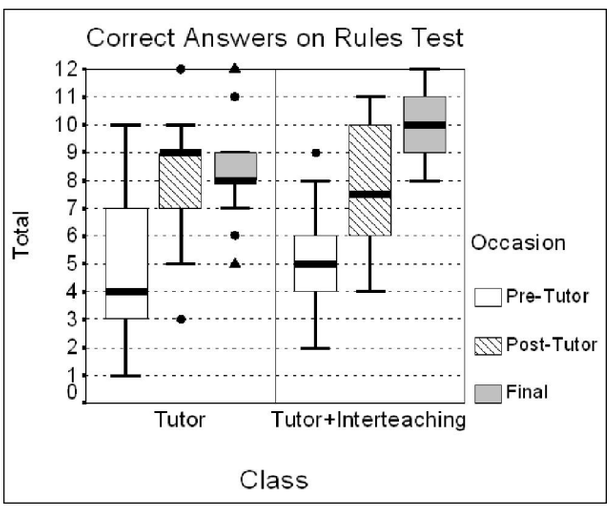


Figure 2. Boxplots of confidence ratings for correct and incorrect answers on the rules test (circles are outliers and triangles are extreme values)

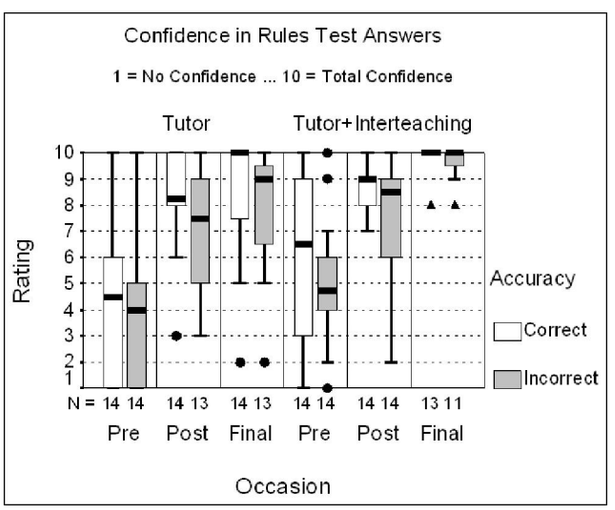
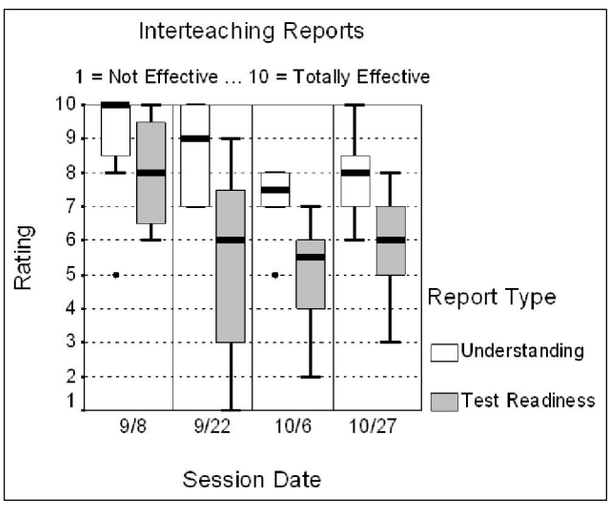


Figure 3. Boxplots of interteaching effectiveness ratings (circles are outliers)

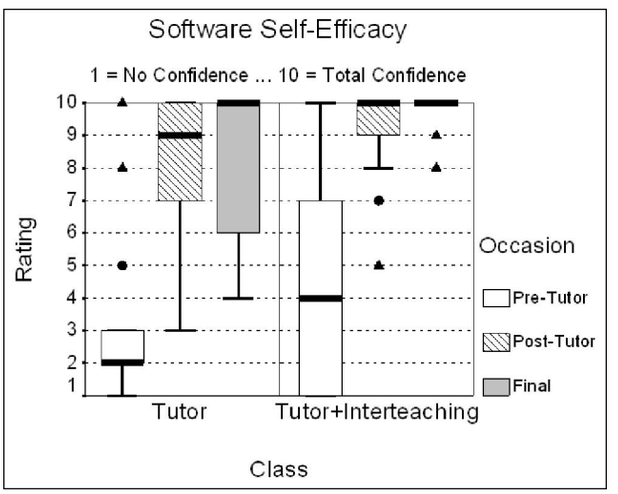


incorrect answers was marginally significant for the Pre-Tutor occasion (chi-square = 3.60, $p < .06$), significant for the Post-Tutor occasion (chi-square = 7.00, $p < .01$), and significant for the Final occasion (chi-square = 4.50, $p < .05$).

For the T+I class, a Friedman's test for confidence ratings of correct answers across the three occasions was significant (chi-square = 20.54, df = 2, $p < .001$). Pairwise contrasts, Bonferroni corrected, were significant for all pairs ($p < .05$). A Friedman's test for confidence ratings of incorrect answers across the three occasions was also significant (chi-square = 13.29, df = 2, $p < .01$). Pairwise contrasts, Bonferroni corrected, were significant for all pairs ($p < .05$) except between the Pre-Tutor and Post-Tutor occasions ($p > .05$). A Friedman's test between confidence ratings for correct and incorrect answers was significant for the Pre-Tutor occasion (chi-square = 5.44, $p < .05$), marginally significant for the Post-Tutor occasion (chi-square = 3.57, $p < .06$), and not significant for the Final occasion (chi-square = 2.00, $p > .10$).

Figure 3 presents boxplots of ratings of the effectiveness of the interteaching session in the T+I class for the two types of ratings: (1) effectiveness of the dialog in understanding the material and (2) effectiveness of the dialog in preparing for a test. The figure presents median ratings across the four sessions for the eight students who were present on all four occasions of interteaching. For understanding, the figure shows graphically that the highest median rating was observed on the first session, in which the value was the maximum of 10. Medians declined thereafter over the next two sessions, and the median increased somewhat during the fourth session. A Friedman's test, however, was not significant (chi-square = 5.10, df = 3, $p > .15$), indicating insufficient evidence to conclude that the changes in medians observed

Figure 4. Boxplots of software self-efficacy (circles are outliers and triangles are extreme values)



graphically were significantly different from each other. For test preparation, the figure shows graphically that the highest median rating was observed on the first session, and ratings were comparatively lower on the other three sessions. A Friedman's test was significant (chi-square = 11.11, df = 3, $p < .05$). Figure 3 also shows graphically that the median understanding rating was higher than the corresponding test readiness median across all four sessions. A Kruskal-Wallis comparison of the differences between the understanding and test readiness ratings for all subjects across the four sessions with a population of zeros was significant (chi-square = 33.75, df = 1, $p < .001$). The correlation between the two sets of ratings was not significant ($r = 0.28$, $p > .10$).

Figure 4 presents boxplots of software self-efficacy ratings across the three occasions for both classes. The ratings are based on the median confidence rating for all 21 unique items of code in the program. The figure shows graphically that students in both classes reported robust increases in confidence between the Pre-Tutor and Post-Tutor occasions, and the median rating reached the ceiling of 10 on the Final occasion. For the T class, Cronbach's alphas for Pre-Tutor, Post-Tutor, and Final occasions were 0.98, 0.99, and 0.99, respectively. The Final alpha was not significant ($p > .05$). A Friedman's test was significant (chi-square = 23.24, df = 2, $p < .001$). For the T+I class, Cronbach's alphas for Pre-Tutor, Post-Tutor, and Final occasions were 0.98, 0.98, and 0.97, respectively. All were significant ($p < .05$). A Friedman's test was significant (chi-square = 20.49, df = 2, $p < .001$). Kruskal-Wallis tests of median ratings between the two classes were not significant for Pre-Tutor, Post-Tutor, and Final occasions (all $p > .05$).

During the Final occasion, students in both classes evaluated the tutor along

three dimensions, where 1 = Negative Opinion to 10 = Positive Opinion. The median ratings were as follows: (1) overall impression (T = 8, T+I = 9), (2) effectiveness of the tutor in learning Java (T = 8, T+I = 8.5), and (3) usability of the tutor (T = 8.5, T+I = 9). Kruskal-Wallis tests between the medians were not significant for any of the three scales (all $p > .05$). The generally positive evaluation of the tutor is indicated by the fact that all medians are eight or higher.

DISCUSSION

Students in two classes showed gains in program understanding and software self-efficacy as a function of participation in several consecutive instructional experiences that were designed to facilitate learning a Java computer program. A programmed instruction tutoring system was effective in promoting initial student confidence and learning, and an interteaching dialog also contributed to performance when these tactics were used within the context of a classroom. Collaborative peer tutoring may have potentiated a student's understanding of general principles of Java that were intended to be taught by the individualized tutoring system. Regarding confidence in understanding, however, the results showed that as student expertise improved, overconfidence was stated in performance, as evidenced by progressively higher confidence ratings for incorrect answers on the rules test. This finding warrants further analysis, especially in light of recent concerns about the validity of the purported relationship between self-efficacy and future performance (Heggestad & Kanfer, 2005).

Taken together, the outcomes of this investigation show how several instructional tactics, including a traditional lecture, may be managed in the classroom to the benefit of introductory programming students. The several tactics applied in sequence provided the occasion for rehearsal, corrective interactions, and overlearning, and these factors have long been related to knowledge and skill development and retention (Salas & Cannon-Bowers, 2001; Swezey & Llaneras, 1997).

The present study falls within the scope of design-based research. This is an attempt to engineer a learning environment by applying principle-based interventions to the classroom and by collecting data on learning effectiveness. It is acknowledged that an actual classroom intervention introduces multiple sources of confounding variables that make causal attribution problematic (Brown, 1992; Collins, 1992; Edelson, 2002). Against a background of increasing criticism that hypothesis-driven laboratory experiments or randomized controlled trials, such as advocated by the U.S. Department of Education (2003), may not yield results having relevance to educational practice, design-based research is emerging as an alternative methodological paradigm. Design-based interdisciplinary activities are now evident by collections of papers in recent special issues of *Educational Researcher* (Kelly, 2002), *Educational Psychologist* (Sandoval & Bell, 2004), and *The Journal of the Learning Sciences* (Barab & Squire, 2004).

As a type of formative evaluation (Collins et al., 2004), the essence of design-based research is systematic replication (Sidman, 1960) in the classroom. Improvements to a previously established and meritorious instructional approach are introduced and evaluated iteratively across successive offerings of a course. Theory

informs the design, and the evaluations stimulate theoretical revisions and subsequent design alterations.

As stated by the Design-Based Research Collective (2003), "Claiming success for an educational intervention is a tricky business" (p. 5). That axiom applies to the present study. To argue that the interteaching improved student performance, it is a given that we accept the potential for confounding influences in the assessments to include the particular students within a class, the gender differences between classes, the time of course delivery, the sequence of events within a class, the sociability of the students, the amount of time spent studying, and perhaps other factors as well. The fundamental question was posed originally by Brown (1992): "What are the absolutely essential features that must be in place to cause change under conditions that one can reasonably hope to exist in normal school settings?" (p. 33). Collins et al. (2004) provide a framework for implementing design-based research to include summative evaluation that is in furtherance of answering that question.

The tactics reported in the present study evolved over several successive semesters. When the author first taught an introductory course in Java, lectures and supervised laboratories were the primary media to deliver information to the students. The reason that the author did that was because he was taught that way, and the traditional approach was the lore of the university culture at the time. The student learning was somewhat active, however, in the sense that students wrote code while the author presented and discussed it. Nevertheless, it was obvious that many bright and highly motivated students, especially international students using English as a second language, were struggling with basic issues such as learning how to type the Java symbols correctly.

In response, an individualized tutoring system was developed that first provided the opportunity for students to learn to type the symbols composing a Java program. The tutor also taught the meaning of the items of code in the program. From a design-based perspective, the tutoring system followed principles of an applied behavior analysis systems approach (Greer, 2002), and the underlying theoretical rationale for the particular design that was implemented is presented in Emurian, Wang, and Durham (2003) and Emurian and Durham (2003). The first tutoring system, which was based on Java AWT, was reported in Emurian, Hu, Wang, and Durham (2000) and Emurian and Durham (2001). Over the years, the tutoring system content and performance were upgraded (Emurian, 2004), and the textual presentation of information was revised to facilitate meaningful learning (Emurian, 2005).

In that latter regard, the current design of the tutor's textual frames follows many of the guidelines offered by Mayer (2002) to promote meaningful learning, which relates to the goal of having students complete the tutoring experience with new knowledge and skill that can be applied to novel situations. Embedded within the tutor frames are advance organizers; signaling; adjunct questions; immediate feedback for performance accuracy and tested understanding of facts, concepts, and rules; and sequential structure building as a superordinate objective that organizes the learning process within a single conceptual framework — the production and understanding of a Java applet.

The features just mentioned were evident in the tutor presented to the sum-

mer 2004 class. Although there were alternatives to the interteaching tactic that was adopted for the fall 2004 class, such as additional lectures and assigned readings, the introduction of a structured social dimension to the classroom learning process was considered the best choice for the next design-based iteration. The theoretical and empirical evidence supporting the value of collaborative peer tutoring was compelling (Rittschof & Griffin, 2001; Slavin, 1996), and teamwork, even in a dyadic situation, provided the occasion for students to accumulate collaborative experiences that are increasingly important in the workplace. That the interteaching also was valuable to the students was evidenced by their positive evaluations of the sessions and by the enhanced performance on the rules-based questions. The reliability and generality of the current outcomes must be determined by subsequent applications of these instructional tactics in the classroom with new groups of students.

According to the Bureau of Labor Statistics (2005), the demand for computer programmers is expected to grow at an average rate with all other occupations through 2012, and employment prospects will be best for programmers with expertise in such object-oriented languages as Java. However, many students in introductory computer programming courses show poor performance that leads to dropping out of courses and programs of study. The tactics presented here revealed a teaching technology that is intended to improve the performance of novice students when they first learn the syntax and semantics of a computer program.

Attempts to understand and overcome the persistently high dropout rates in STEM courses and programs (Thomas et al., 2000; Wormley, 2003) suggest the potential contributions of this study to that objective. The systematic replications over many semesters, which provide the foundation for the present and future work, repeatedly and consistently show the value of providing students with several instructional opportunities that may interact synergistically to promote skill and confidence that are the intended outcomes of our teaching tactics for each and every student in our classes.

REFERENCES

Barab, S., & Squire, K. (2004). Design-based research: Putting a stake in the ground. *The Journal of the Learning Sciences, 13*(1), 1-14.

Barnett, S. M., & Ceci, S. J. (2002). When and where do we apply what we learn? A taxonomy for far transfer. *Psychological Bulletin, 128*(4), 612-637.

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*(2), 215-226.

Brown, A. L. (1992). Design experiments: Theoretical and methodological challenges in creating complex interventions in classroom settings. *The Journal of the Learning Sciences, 2*(2), 141-178.

Bureau of Labor Statistics, U.S. Department of Labor. (2005). *Occupational outlook handbook, 2004-05 edition*. Retrieved September 29, 2005, from http://www.bls.gov/oco/ocos110.htm

Collins, A. (1992). Toward a design science of education. In E. Scanlon, & T. O'Shea (Eds.), *New directions in educational technology* (pp. 15-22). New York: Springer-Verlag.

Collins, A., Joseph, D., & Bielaczyc, K.

(2004). Design research: Theoretical and methodological issues. *The Journal of the Learning Sciences, 13*(1), 15-42.

Davy, J., & Jenkins, T. (1999). Research-led innovation in teaching and learning computer programming. In *Proceedings of the 4th Annual SIGCSE/SIGCUE ITiCSE Conference on Innovation* (pp. 5-8). Retrieved September 29, 2005, from http://portal.acm.org/citation.cfm?id=305786.305826

Design-Based Research Collective. (2003). Design-based research: An emerging paradigm for educational inquiry. *Educational Researcher, 32*(1), 5-8.

Edelson, D. C. (2002). Design research: What we learn when we engage in design. *The Journal of the Learning Sciences, 11*(1), 105-121.

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. (in press). A Web-based tutor for Java™ Evidence of meaningful learning. *International Journal of Distance Education Technologie*s.

Emurian, H. H., & Durham, A. G. (2001, May 20-23). A personalized system of instruction for teaching Java. In M. Khosrow-Pour (Ed.), *Managing Information Technology in a Global Economy, 2001 Information Resources Management Association International Conference* (pp. 155-160), Toronto, Ontario, Canada. Hershey, PA: Idea Group Publishing.

Emurian, H. H., & Durham, A. G. (2002, May 19-22). Enhanced learning on a programmed instruction tutoring system for JAVA. In M. Khosrow-Pour (Ed.), *Issues and Trends of Information Technology Management in Contemporary Organizations, 2002 Information Resources Management Association International Conference* (pp. 205-208), Seattle, Washington, USA. Hershey, PA: Idea Group Publishing.

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., Hu, X., Wang, J., & Durham, A. G. (2000). Learning Java: A programmed instruction approach using applets. *Computers in Human Behavior, 16*(4), 395-422.

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.

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*(1), 5-16.

Griffin, M. M., & Griffin, B. W. (1998). An investigation of the effects of reciprocal peer tutoring on achievement, self-efficacy, and test anxiety. *Contemporary Educational Psychology, 23*(3), 298-311.

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.

Hoadley, C. M. (2004). Methodological alignment in design-based research. *Educational Psychologist, 39*(4), 203-212.

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.

Kelly, A. F. (2002). Research as design. *Educational Researcher, 32*(1), 3-4.

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

Rittschof, K. A., & Griffin, B. W. (2001). Reciprocal peer tutoring: Re-examining the value of a co-operative learning technique to college students and instructors. *Educational Psychology, 21*(3), 313-331.

Roberts, E. (2004). Resources to support the use of Java in introductory computer science. In *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education* (pp. 233-234), Norfolk, Virginia. Retrieved September 29, 2005, from http://portal.acm.org/citation.cfm?id=971384

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

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

Sandoval, W., & Bell, P. (2004). Design-based research methods for studying learning in context: Introduction. *Educational Psychologist, 39*(4), 199-201.

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

Singer-Dudek, J., & Greer, R. D. (2005, Summer). A long-term analysis of the relationship between fluency and the training and maintenance of complex math skills. *The Psychological Record, 55*, 361-376.

Slavin, R. E. (1996). Research on cooperative learning and achievement: What we know, what we need to know. *Contemporary Educational Psychology, 21*, 43-69.

Swezey, R. W., & Llaneras, R. E. (1997). Models in training and instruction. In G. Salvendy (Ed.), *Handbook of human factors and ergonomics* (pp. 514-577). New York: Wiley.

Thomas, A., Benne, M. R., Marr, M. J., Thomas, E. W., & Hume, R. M. (2000). The evidence remains stable: the MBTI predicts attraction and attrition in an engineering program. *Journal of Psychological Type, 55*, 35-42.

Traynor, D., & Gibson, J. P. (2004). Towards the development of a cognitive model of programming: A software engineering approach. In *Proceedings of the 16th Meeting of the Psychology of Programming Interest Group*. Retrieved August 10, 2005, from http://www.cs.may.ie/~dtraynor/papers/PPIGarticle.pdf

U.S. Department of Education. (2003). *Identifying and implementing educational practices supported by rigor-*

ous evidence. Retrieved September 29, 2005, from http://www.excelgov.org/evidence

Williams, L. A., Kessler, R. R., Cunningham, W., & Jeffries, R. (2000). Strengthening the case for pair-programming. *IEEE Software, 17*(4), 19-25.

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.

Wormley, D. (2003). Engineering education and the science and engineering workforce. In M. A. Fox (Ed.), *Panorganizational summit on the U.S. science and engineering workforce: Meeting summary* (pp. 40-46). Washington, DC: National Academy of Science.

ENDNOTES

1 The tutoring system is freely accessible on the Web at http://nasa1.ifsm.umbc.edu/learnJava/tutorLinks/TutorLinks.html. Since this article was written, the tutor has been updated to teach Java Swing. Both versions of the tutor are accessible from the link provided. The source code for the Java tutor is freely available and may be obtained by contacting the author.

2 The program is presented in Appendix A.

3 The course description and online material are available on the Web at http://nasa1.ifsm.umbc.edu/IFSM413_613/

4 http://userpages.umbc.edu/~emurian/learnJava/ijicte1/

5 Data were missing for one T+I student in the Final occasion.

Henry Emurian is an associate professor in the Information Systems Department at UMBC. His background includes degrees in clinical psychology and computer science, and he is a licensed psychologist in Maryland. He is interested in exploring the applications of information technology to solve important problems related to education, health, and public service. His teaching and research interests include technology education strategies, Web-based tutoring systems, the empowerment of citizens via the Internet, and stress consequences of human-computer interactions. He has published in such journals as The Journal of the Experimental Analysis of Behavior, Journal of E-Commerce in Organizations, Computers in Human Behavior, Basic and Applied Social Psychology, *and* Aviation, Space, and Environmental Medicine. *He is a member of the American Psychological Association, the Information Resources Management Association, and the Association for Behavior Analysis.*

APPENDIX A.

Interteaching Report #1
Your name _____ Date _____
Your partner's name: _____
You should understand the components of the program below at a level given in the Java Tutor. Discuss these components with the intention to understand the specific item and any general principle that is reflected in an item or collection of items. An example of a general principle would be to begin the name of a class with a capital letter.

```
import java.applet.Applet;
import java.awt.Label;
public class MyProgram extends Applet {
Label myLabel;
public void init() {
myLabel = new Label("This is my first program.");
add(myLabel);
myLabel.setVisible(true);
}
}
```

How effective was this session in helping you to learn the material?

1 = Not at all effective. The session did not contribute to my learning of the material.
10 = Totally effective. The session contributed to my learning of the material.
(Not effective) **1 2 3 4 5 6 7 8 9 10** (Totally effective)

Enter one number that describes the effectiveness for you: _____.

How confident are you that you could answer all the questions correctly if you were tested on this program right now?

1 = Not at all confident. I could not answer any question correctly.
10 = Totally confident. I could answer all the questions correctly.
(Not confident) **1 2 3 4 5 6 7 8 9 10** (Totally confident)

Enter one number that describes your confidence: _____.