Programmed Instruction for Teaching Java: Consideration of Learn Unit Frequency and Rule-Test Performance

Henry H. Emurian

Abstract

At the beginning of a Java computer programming course, nine students in an undergraduate class and nine students in a graduate class completed a web-based programmed instruction tutoring system that taught a simple computer program. All students exited the tutor with an identical level of skill, at least as determined by the tutor's required terminal performance, which involved writing the program and passing multiple-choice tests on the program's elements. Before entering and after exiting the tutor, students completed a test of rule-based performance that required applications of general programming principles to solve novel problems. In both classes, the number of correct rule answers observed before entering the tutor did not predict the number of learn units that students subsequently used to complete the tutor. However, the frequency of learn units was inversely related to post-tutor rule-test performance, *i.e.*, as the number of learn units used in the tutor increased over students, the number of correct answers on the post-tutor rule test decreased. Since time to complete the tutor was unrelated to learn unit frequency, these data suggest that high achieving students may have generated autoclitic learn units while using the tutor. Interteaching, as an occasion for generating and sharing interlocking learn units, may be an effective complement to programmed instruction in promoting optimal learning in all students.

Keywords*:* Programmed instruction, learn units, rule-governed performance, technology education.

Programmed instruction (PI) is an effective tool to teach students in information systems to write and to understand a Java computer program as a first technical exercise in a computer programming course[i]. A web-based PI tutoring system to accomplish that objective was presented in Emurian, Hu, Wang, and Durham (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). Since the initial prototype tutoring system was developed, the implementation and assessments have evolved over several updates, ranging from analyses of learning performance (Emurian, 2004) to considerations of the extent to which skill gained by completing the tutor fostered generalizable or rule-governed performances (Emurian, 2005, 2006a). More recently, interteaching (Boyce & Hineline, 2002; Saville, Zinn, Neef, Norman, & Ferreri, 2006) was shown to support and confirm the students' knowledge acquired initially from using the programmed instruction tutor (Emurian, 2006b, in press).

An obviously important consideration in the design of a programmed instruction tutoring system is the extent to which the knowledge or skill gained is generalizable to solve problems not explicitly taught or encountered with the tutor itself. Behavior analysts working in the areas of programmed instruction (*e.g.*, Tudor & Bostow, 1991) and instructional design (*e.g.*, Greer, 2002) recognize the importance of this issue. In a traditional formulation of learning, this is a transfer of training problem (Barnett & Ceci, 2002), and in behavior analysis, this is a rule-governed problem (Hayes, 1989). When learners apply knowledge successfully to novel situations, they are said to be demonstrating meaningful learning (Mayer, 2002). Generalizable rules, which may be the essence of meaningful learning, can be acquired by direct instruction and rehearsal or by induction, when many different situations are encountered that occasion the general rule (Kudadjie-Gyamfi & Rachlin, 2002). That recognition implies 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 former tactic is consistent with our instructional system design.

A tacit assumption in the design of programmed instruction is that all learners who exit the system do so having achieved equivalent skill. A framework for interpreting individual differences in the

learning process is given by the ***learn unit*** formulation of Greer and McDonough (1999). A learn unit is defined as "a countable unit of teacher and student interaction that leads to important changes in student behavior" (Greer & McDonough, 1999, p. 6). In the automated tutoring system under consideration, each successive component, or learn unit, within eight tutor stages, to be described below, 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*** (Greer, 2002, p. 18).

The importance of this framework is to be understood in terms of its assumed predictability of long-term outcomes. Although our previous work showed increases in the number of correct rule-test answers following completion of a PI tutoring system (Emurian, 2005; 2006a, b), in comparison to pre-tutor answers, no attempt was made to relate such tests of rule-governed performance to the frequency of learn units that students encountered until successfully completing the tutor. If generalizable knowledge is equivalent and if individual differences have been overcome by repetition of learn units until mastery has occurred, there should be no differences in rule-governed performances among students upon completion of the tutoring system, and the frequency of learn units should be unrelated to subsequent performance. This paper, then, presents an evaluation of learn unit frequency in relationship to students' rule-test performance observed (1) prior to using the programmed instruction tutoring system (baseline) and (2) after successfully completing the tutoring system.

METHOD

Subjects

Subjects were students in two classes of a course entitled *Graphical User Interface Systems Using Java*. The first class was for undergraduate students, and it met in the Fall semester of 2005. The class met twice each week for 75 min over a 14-week semester. The second class was for graduate students, and it met in the Summer of 2006. This class met three times each week for three hrs over a five-week semester. The technical course content was equivalent between the classes, but the graduate students were required to write more journal article reviews than the undergraduate students. The prerequisite for both classes was at least one prior programming course, and the course design was intended to provide instruction to information systems majors. The primary professional interests of those students are in the areas of systems analysis and technical management, in contrast to computer programming and algorithms.

For the Fall 2005 undergraduate class, nine of the 14 students produced usable records of tutor performance. There were six male and three female students (median age = 23 yrs, range = 19 to 28 yrs; median number of programming courses taken = 3, range = 2 to 5 courses). For the Summer 2006 graduate class, nine of the 13 students produced usable records of tutor performance. There were six male and three female students (median age = 26 yrs, range = 23 to 33 yrs; median number of programming courses taken = 3, range = 1 to 15 courses). Data were automatically recorded on a server at the transition points in the tutor, and occasional technical problems with the server would prevent all records from being recorded for a particular student. That accounts for the number of data records being fewer than the number of students in a class.

Material

At the first meeting for both classes, students completed a pre-tutor questionnaire (baseline) that included rule-based questions. Appendix A presents the 14 rule questions administered to the Fall 2005 students, and Appendix B presents the 12 rule questions administered to the Spring 2006 students. The

questionnaire changed over classes in relationship to updates to the Java tutor. Other questionnaires were also administered, to include demographic items and software self-efficacy (see Emurian, 2005). Following completion of the Java tutor, students in both classes repeated the rule-based questions in a post-tutor assessment.

The programmed instruction tutoring system taught a simple Java applet that would display a text string, as a JLabel object, within a browser on the World Wide Web[ii].
Table 1 presents the Java code for the tutor version presented to the Fall 2005 students. The program was arbitrarily organized into ten lines of code and 34 items of code. Each item occupies a cell in the table, and PI learn units were based upon cells and lines. Table 2 presents the Java code for the tutor version presented to the Summer 2006 students. The program was arbitrarily organized into 11 lines of code and 37 items of code. Each item occupies a cell in the table, and PI learn units were based upon cells and lines. The symbols in the tables may be cryptic on initial encounter, but the objective of the tutor was to teach students to understand and use the symbols in the tables, even if the student had never before written a Java computer program.

Table 1. The Java program for the tutor version presented to the Fall 2005 students.

import	**javax.swing.JApplet**	**;**			
import	**javax.swing.JLabel**	**;**			
public	**class**	**MyProgram**	**extends**	**JApplet**	**{**
JLabel	**myLabel**	**;**			
public	**void**	**init()**	**{**		
myLabel	**=**	**new**	**JLabel (“Java”)**	**;**	
myLabel	**.**	**setVisible(true)**	**;**		
getContentPane()	**.**	**add(myLabel)**	**;**		
}					
}					

Table 2. The Java program for the tutor version presented to the Summer 2006 students.

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)**	**;**		

The tutor and the open source code are freely available on the web, as referenced above. Running the tutor is the best way to understand its components and operation. In brief, 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 = 34 or 37), (3) learning to discriminate an item of code in a list (learn units = 34 or 37), (4) learning the semantics of an item of code (learn units = 34 or 37) and learning the syntax by typing the item by recall (learn units = 34 or 37), (5) learning to copy a line of code (learn units = 10 or 11), (6) learning to discriminate a line of code in a list (learn units = 10 or 11), (7) learning the semantics of a line of code (learn units = 10 or 11) and learning the syntax by typing the line by recall (learn units = 10 or 11), and (8) writing the entire program by recall (learn units = 1). Thus, for the Fall 2005 class, the minimum number of learn units to complete the tutor was 178; for the Summer 2006 class, the minimum number of learn units was 194. The multiple-choice tests for items and lines of code, which are embedded in the tutor, 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. For both tutor versions, the lines stage had no delay interval or confirmation window. Experience suggested that most students in our courses could complete the tutor within two to three hrs. 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.

At the completion of each tutor stage, a record of performance, which included stage duration and performance errors, was transmitted to a database on the server. The only identifier was the student's email username, similar to what appears in any email transmission on the Internet. Students were informed of this record's transmission, and the instructional protocol was exempt from informed consent. Network interruptions sometimes prevented a record from being written, and this paper includes only those students who had records for all eight stages in the tutor.

Procedure

For the Fall 2005 class, the first meeting provided orientation to the course. The pre-tutor questionnaire was administered and collected. As homework, students were instructed to complete the Java tutor before the next class meeting. At the completion of the tutor, students downloaded the post-tutor questionnaire from the course Blackboard site, completed it electronically, and returned it to the instructor by email attachment. For the Summer 2006 class, the procedure was similar, but it was the expectation that all students could complete the tutor within the three-hr class period. Thus, the post-tutor questionnaire was administered and collected as students completed the tutor in class. Only one student was not able to complete the tutor during class, and that student was not included in the data to be presented here because the records were not obtained.

RESULTS

Figure 1 presents scatterplots of total learn units and total correct rule-test answers for the pre-tutor baseline and post-tutor assessments for each of the students in the Fall 2005 class. Regression lines and Pearson correlation parameters are presented on the figure. The figure shows graphically that correct rule answers increased from pre-tutor baseline to post-tutor assessment for all nine students. The figure also shows that pre-tutor correct rule answers were not demonstrably related to subsequent learn unit frequency, and the test of the relationship failed to provide evidence of an orderly relationship. In contrast, the figure shows graphically that total learn units encountered during the tutor were inversely related to subsequent post-tutor correct rule answers, and the test of the relationship was significant. Students who required comparatively fewer learn units during the tutor achieved a higher number of post-tutor correct rule answers in comparison to students who required comparatively more learn units to complete the tutor.

Figure 2 presents scatterplots of total learn units and total correct rule-test answers for

the pre-tutor baseline and post-tutor assessments for each of the students in the Summer 2006 class. Regression lines and Pearson correlation parameters are presented on the figure. The figure shows graphically that correct rule answers increased from pre-tutor baseline to post-tutor assessment for eight of the nine students. The exception was a student who answered 11 correct rule questions on both occasions. The figure also shows that pre-tutor correct rule answers were not demonstrably related to subsequent learn unit frequency, and the test of the relationship failed to provide evidence of an orderly relationship. In contrast, the figure shows graphically that total learn units encountered during the tutor were inversely related to subsequent post-tutor correct

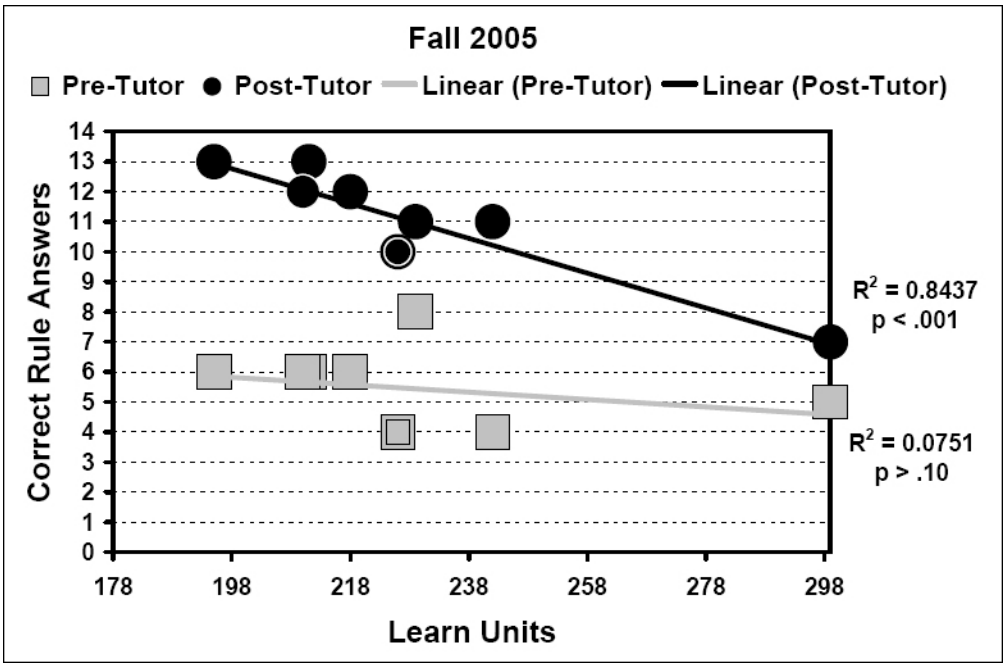

Figure 1. **Scatterplots of learn units and correct rule answers for pre-tutor and post-tutor assessments for nine students in the Fall 2005 class. Also presented are regression lines and R^2 values for each line.**

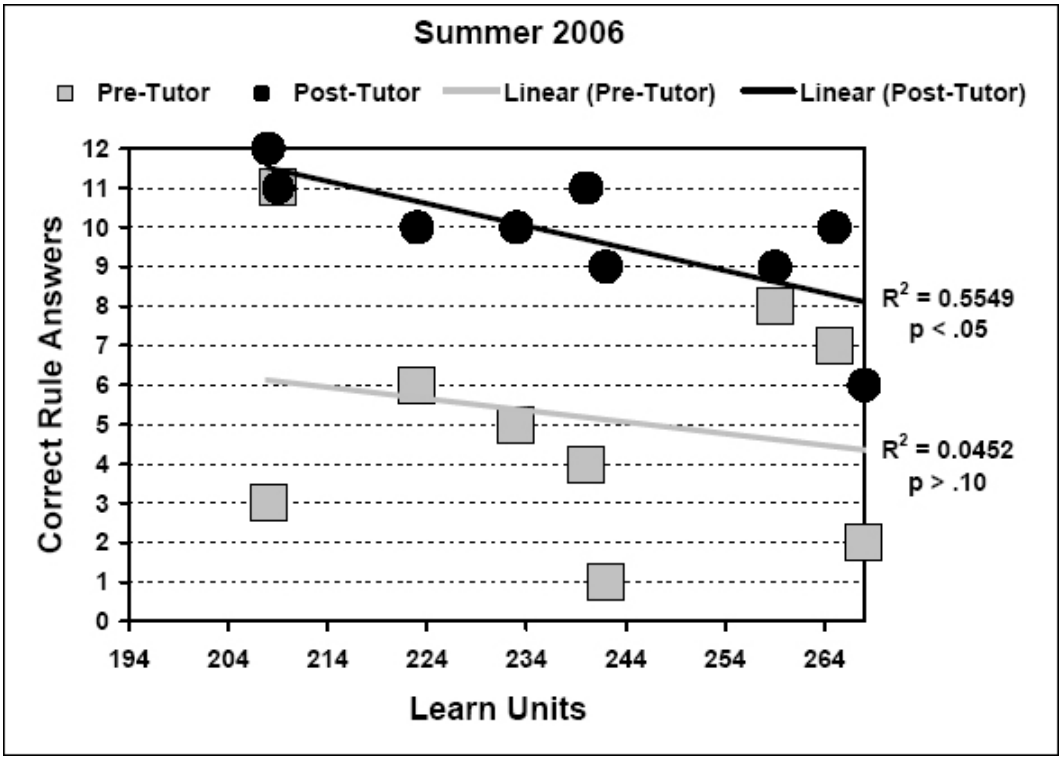

Figure 2. **Scatterplots of learn units and correct rule answers for pre-tutor and post-tutor assessments for nine students in the Summer 2006 class. Also presented are regression lines and R^2 values for each line.**

rule answers, and the test of the relationship was significant. Students who required comparatively fewer learn units during the tutor achieved a higher number of post-tutor correct rule answers in comparison to students who required comparatively more learn units to complete the tutor.

Figure 3 presents scatterplots of total learn units to complete the tutor and total time to complete the tutor for each of the students in the Fall 2005 class. The shortest time to complete the tutor was 60.5 min, and the longest time was 543.2 min. A Pearson correlation did not support a relationship between time and total learn units.

FIGURE 3, NEXT PAGE!

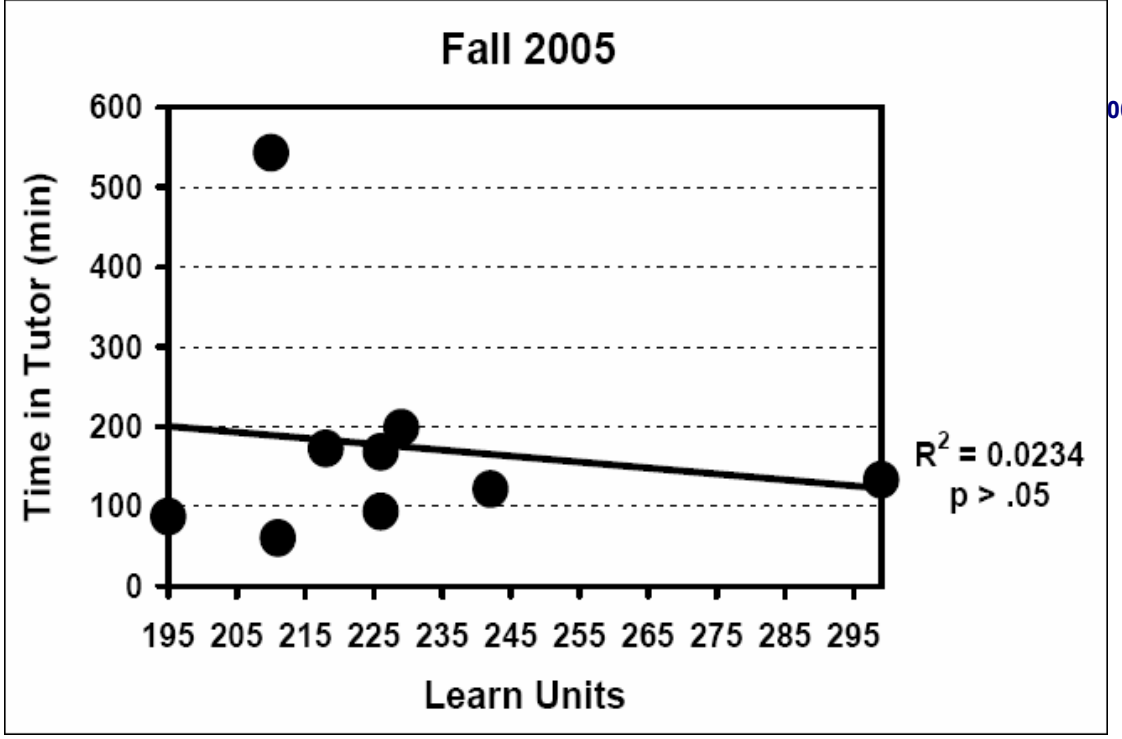

Figure 3. **Scatterplots of total time required to complete the tutor and total learn units required to complete the tutor for the nine students in the Fall 2005 class.**

Figure 4 presents scatterplots of total learn units to complete the tutor and total time to complete the tutor for each of the students in the Summer 2006 class. The shortest time to complete the tutor was 60.0 min, and the longest time was 143.1 min. A Pearson correlation did not support a relationship between time and total learn units.

DISCUSSION

The relationship between learn unit frequency and rule-test performance was orderly for students over two different classes. As the number of learn units to complete the tutor increased over students, the number of correct answers produced on the rule test decreased. However, the data did not support a similar relationship between baseline rule-test performance and subsequent learn unit frequency or between time in the tutor and learn unit frequency to complete the tutor. These effects were observed for undergraduate and graduate students, for two versions of the Java program, for two versions of the rule test, and for two ways to complete the tutor: (1) homework, which favored distributed learning, and (2) class work, which supported massed learning.

FIGURE 4, NEXT PAGE!

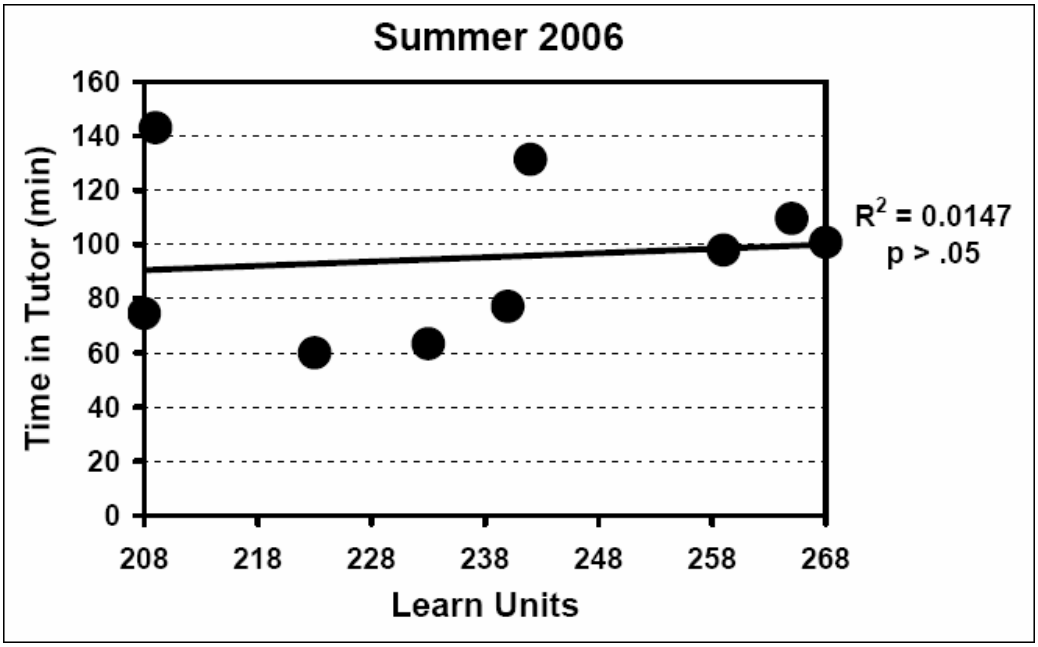

***Figure 4*. Scatterplots of total time required to complete the tutor and total learn units required to complete the tutor for the nine students in the Summer 2006 class.**

The similarity in outcomes shows the reliability of the behavioral processes under conditions of systematic replication (Sidman, 1960), thereby demonstrating the generality of the findings.

All students in the Fall 2005 class and eight of the nine students in the Summer 2006 class showed improvement in rule-test performance between the pre-tutor baseline and the post-tutor assessment. However, the knowledge transfer from the tutor to the rule test (*i.e.*, "far transfer") differed across students, despite the fact that all students exited the tutor with equivalent competency, at least as operationalized by tutor performance. In fact, only one student (Summer 2006) showed perfect performance on the post-tutor assessment. The orderliness observed, then, was to predict post-tutor rule-test performance by the number of learn units required to complete the tutor. Surprisingly, however, the evidence was insufficient to show a similar orderly relationship between pre-tutor baseline rule-test performance and the number of learn units later required to complete the tutor. This outcome is consistent with the observation that frequent testing and test feedback that is immediate and corrective, such as occurred within the Java tutor, may sometimes hinder, rather than help, knowledge acquisition by giving learners an overly optimistic opinion of their competency (Mathan & Koedinger, 2005). Although students may sometimes prefer item-by-item feedback (*e.g.*, Buzhardt & Semb, 2002), it is also acknowledged that multiple-choice recognition tests may assess only low-level learning processes and may lead to overconfidence in long-term retention (Halpern & Hakel, 2003).

The observation that total correct rule answers on the post-tutor assessment were generally lower for students who used a relatively greater number of learn units suggests that equivalent "knowledge" was not achieved by all students at the completion of the tutor. When multiple-choice tests are used to assess learning, as was the case in the present tutor, the opportunity for repetition may not always occasion the "studying behavior" essential to apply a principle to the selection of the correct alternative on a test. Although the term "learn unit" was applied to the tutor components, the failure of transfer to occur for all students indicates that the components did not always satisfy the definition of a learn unit for all students, since the interaction with the component events apparently did not change all students' behavior in the direction intended. As stated by Greer and McDonough (1999), "The learn unit is present when student

learning occurs in teaching interactions and is absent when student learning does not occur in teaching interactions" (p. 6). Although 17 of the 18 students in this study showed improvement in rule-test performance over pre-tutor baseline, this outcome does not support the conclusion that those same 17 students all encountered identical learn units. It was possible, for example, for a student to repeat a tutoring system component until a correct multiple-choice question was answered correctly simply by selecting a different answer until a transition occurred.

An automated instructional system is challenged to provide tests of learning complex rules, especially when the design of learn units requires correct performance for progress in the system. In the present tutor, for example, the display of textual descriptions and explanations of the semantics of Java code is followed by a multiple-choice test. A correct or corrective choice operationalizes the learn unit. The tactic is to present learn units until evidence of hierarchical relational networks (Hayes, Fox, Gifford, Wilson, Barnes-Holmes, & Healy, 2001) emerges and can be documented. Although a student may select the correct choice on a multiple-choice test, the challenge is to make certain that the choice is under the functional control of the rule and is evidence of the existence of equivalence relations. How a reader may come to such verbal expertise is addressed by behavior analysts in terms of milestones such as verbal mediation for problem solving (Greer & Keohane, 2006). Once such a milestone has been reached, presumably a reader would benefit from exposure to text that has been designed to facilitate the acquisition of rule-governed performances. One interpretation of these data, then, is that the textual information contained with the tutor frames did not adequately support the solution of rule-based problems for all students.

In that latter regard, the textual frames in the Java tutor follow many of the guidelines suggested by Mayer (2002) to promote meaningful learning: advance organizers; signaling; adjunct questions; immediate feedback for performance accuracy; tested understanding of facts, concepts, and rules; and sequential structure building as a terminal performance to organize the learning process within a single conceptual objective – the student's production and understanding of a Java applet. Moreover, that students may acquire skill to generate their own learn units is the rationale for much of the work in self-regulated learning (Kauffman, 2004) and reflection (Masui & DeCorte, 2005). The conditions for the manifestation of such repertoires, however, include a history of reinforcement necessary to sustain responding in relationship to the textual material under study, otherwise defined as a student's "interest" in the subject matter (Dornisch & Sperling, 2006). How learners might integrate motivational factors during learning with transfer performance has recently been addressed in the educational psychology literature (Pugh & Bergin, 2006).

The fact that insufficient evidence existed to support a relationship between learn units required to complete the tutor and time in the tutor suggests that the students' behavioral interactions with the textual material differed, which was manifested in the post-tutor rule-test performances. Those differences plausibly relate to generative learn units in excess of the ones present in the programmed instruction tutoring system. How to foster a student's autoclitic learn units, occasioned by reading and by the questioner (speaker) and the answerer (listener) being the solitary "thinker" (Catania, 1998; Skinner, 1957), is a challenge for behavior analysis. If teachers can be taught to modify their behavior to produce learn units that will change the behavior of their students (*e.g.*, Keohane & Greer, 2005; Ross, Singer-Dudek, & Greer, 2005), there is reason to believe that students may also be taught to generate learn units when the intended effect is on themselves (cf. Azevedo & Cromley, 2004), at least initially. Issues of self-editing and self-monitoring may arise, however, to insure that the intended consequences are supported when the listener is someone or some agency other than the speaker (Epting & Critchfield, 2006).

Learning is a process that includes the actions of study and practice (Swezey & Llaneras, 1997), sometimes for years (Ericsson & Lehmann, 1996), and the assessment of effectiveness as a change in the learner (Skinner, 1953, 1954), a change that might be observed and documented by others or even by the

learner as a self-evaluating authority (Eilam & Aharon, 2003; Pintrich, 2003; Zimmerman, 1994). Behavior analysts have proposed and evaluated instructional techniques to overcome individual differences *(e.g.*, Emurian, 2001), and these include the personalized system of instruction (Keller, 1968) and its descendents (Ferster & Perrott, 1968), direct instruction (Watkins & Slocum, 2004), precision teaching (Chisea & Robertson, 2000), and interteaching (Boyce & Hineline, 2002). The learn unit framework is promising as a technique to consider when developing programmed instruction for technology education. The challenges associated with operationalizing all relevant controlling variables within the context of an automated instructional system support the synergistic applications of interteaching, facilitated by rubrics to generate interlocking learn units, as a beneficial approach to overcoming individual differences in this area of technical education (*e.g.*, Emurian, 2006b, in press). As stated by Keohane and Greer (2005), "...it can be argued that most if not all of the applied research literature has direct or indirect application to the teaching process" (p. 252). Our objective is to apply that literature to assist our students to be skilled and knowledgeable in the domain of information technology.

REFERENCES

Azevedo, R., & Cromley, J.G. (2004). Does training on self-regulated learning facilitate students' learning with hypermedia? *Journal of Educational Psychology*, *96*(3), 523-535.

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.

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.

Buzhardt, J., & Semb, G.B. (2002). Item-by-item versus end-of-test feedback in a computer-based PSI course. *Journal of Behavioral Education, 11*(2), 89-104.

Catania, A.C. (1998). The taxonomy of verbal behavior. In K. A. Lattal & M. Perone (Eds.), *Handbook of Research Methods in Human Operant Behavior* (pp. 405-433). New York: Plenum Press.

Chiesa, M., & Robertson, A. (2000). Precision teaching and fluency training: Making math easier for pupils and teachers. *Educational Psychology in Practice*, *16*(3), 297-310.

Dornisch, M.M., & Sperling, R.A. (2006). Facilitating learning from technology-enhanced text: Effects of prompted elaborative interrogation. *The Journal of Educational Research*, *99*(3), 156-165.

Eilam, B., & Aharon, I. (2003). Student's planning in the process of self-regulated learning. *Contemporary Educational Psychology*, *28*(3), 304-334.

Emurian, H.H. (2001). The consequences of e-Learning. (Editorial), *Information Resources Management Journal*, Apr-Jun, 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., Holden, H.K., & Abarbanel, R.A. (in press). 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.

Epting, L.K., & Critchfield, T.S. (2006). Self-editing: On the relation between behavioral and psycholinguistic approaches. *The Behavior Analyst*, *29*(2), 211-234.

Ericsson, K.A., & Lehmann, A.C. (1996). Expert and exceptional performance: Evidence of maximal adaptation to task constraints. *Annual Review of Psychology*, *37*, 273-305.

Ferster, C.B., & Perrott, M.C. (1968). *Behavior Principles*. New York: Appleton-Century-Crofts.

Greer, R.D. (2002). *Designing Teaching Strategies: An Applied Behavior Analysis Systems Approach*. New York: Academic Press.

Greer, R.D., & Keohane, D.-D. (2006). The evolution of verbal behavior in children. *Journal of Speech -- Language Pathology and Applied Behavior Analysis*, *1*(2), 111-140.

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.

Hayes, S.C. (1989). *Rule-Governed Behavior: Cognition, Contingencies, and Instructional Control*. New York: Plenum Press.

Hayes, S.C., Fox, E., Gifford, E.V., Wilson, K.G., Barnes-Holmes, D., & Healy, O. (2001). Derived relational responding as learned behavior. In S.C. Hayes, D. Barnes-Holmes, & B. Roche (Eds.), *Relational Frame Theory: A Post-Skinnerian Account of Human Language and Cognition* (pp. 21-49). New York: Kluwer Academic/Plenum Publishers.

Kauffman, D.F. (2004). Self-regulated learning in web-based environments: Instructional tools designed to facilitate cognitive strategy use, metacognitive processing, and motivational beliefs. *Journal of Educational Computing Research*, *30*(1 & 2), 139-161.

Keller, F.S. (1968). Goodbye teacher... *Journal of Applied Behavior Analysis*, *1*, 79-89.

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.

Keohane, D.-D., & Greer, R.D. (2005). Teachers' use of a verbally governed algorithm and student learning. *International Journal of Behavioral Consultation and Therapy*, *1*(3), 252-271.

Kudadjie-Gyamfi, E., & Rachlin, H. (2002). Rule-governed versus contingency-governed behavior in a self-control task: Effects of changes in contingencies. *Behavioral Processes*, *57*(1), 29-35.

Mathan, S.A., & Koedinger, K.R. (2005). Fostering the intelligent novice: Learning from errors with metacognitive tutoring. *Educational Psychologist*, *40*(4), 257-265.

Masui, C., & DeCorte, E. (2005). Learning to reflect and to attribute constructively as basic components of self-regulated learning. *British Journal of Educational Psychology*, *75*, 351-372.

Mayer, R.E. (2002). *The Promise of Educational Psychology. Volume II. Teaching for Meaningful Learning*. Upper Saddle River, NJ: Pearson Education, Inc.

Pintrich, P.R. (2003). A motivational science perspective on the role of student motivation in learning and teaching contexts. *Journal of Educational Psychology*, *95*(4), 667-686.

Pugh, K.J., & Bergin, D.A. (2006). Motivational influences on transfer. *Educational Psychologist*, *41*(3), 147-160.

Ross, D.E., Singer-Dudek, J., & Greer, R.D. (2005). The teacher performance rate and accuracy scale (TPRA): Training as evaluation. *Education and Training in Developmental Disabilities*, *40*(4), 411- 423.

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.

Skinner, B.F. (1957). *Verbal Behavior*. New York: Appleton-Century-Crofts.

Skinner, B.F. (1954). The science of learning and the art of teaching. *Harvard Educational Review*, *24*, 86-97.

Skinner, B.F. (1953). *Science and Human Behavior*. New York: The Free Press.

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

Tudor, R.M., & Bostow, D.E. (1991). Computer-programmed instruction: The relation of required interaction to practical application. *Journal of Applied Behavior Analysis*, *24*, 361-368.

Watkins, C.L., & Slocum, T.A. (2004). The components of direct instruction. *Journal of Direct Instruction*, *3*(2), 75-110.

Zimmerman, B.J. (1994). Dimensions of academic self-regulation: A conceptual framework for education. In D.H. Schunk & B.J. Zimmerman (Eds.), *Self-Regulation of Learning and Performance* (pp. 3-21). Hillsdale, NJ: Erlbaum.

Appendix Follows

Appendix A

1. Which of the following lines most likely would be used to create a shorthand notation for the compiler to locate the JFrame class, which is built-in to Java?
   a. import ../class/JFrame;
   b. access JFrame.class;
   c. import javax.swing.JFrame;
   d. import java.awt.JFrame.class;
   e. append javax.swing.JFrame;

2. Which of the following lines most likely would be used to construct an instance of the JButton class?
   a. Button = new JButton("Hello");
   b. myJButton = new JButton("Click Me");
   c. Button = new Button("Hello");
   d. myButton = Button.class("Hello");
   e. myButton = new JButton("Click Me").

3. Which of the following lines most likely would be used to add a Checkbox object to a content pane?
   a. getContentPane.Add(myCheckBox);
   b. container.Add(CheckboxObject);
   c. add(container.Checkbox);
   d. getContentPane().add(myBox);
   e. add(myCheckBox);

4. Which of the following lines most likely overrides a method that is contained in the Applet class?
   a. public Void stop{} { lines of Java code here }
   b. public void Stop(){ lines of Java code here }
   c. public void stop() {lines of Java code here }
   d. Public Void Stop() (lines of Java code here)
   e. Public void stop() { lines of Java code here }

5. Which of the following sequences is correct?
   a. declare a JTextField object, construct a JTextField object, add a JTextField object to a container.
   b. construct a JTextField object, declare a JTextField object, add a JTextField object to a container.
   c. declare a JTextField object, add a JTextField object to a container, construct a JTextField object.
   d. add a JTextField object to a container, declare a JTextField object, construct a JTextField object.
   e. add a JTextField object to a container, construct a JTextField object, declare a JTextField object.

6. Given the line, **public class MyTextArea extends JTextArea {**, which of the following statements is correct?
   a. JTextArea is a subclass of MyTextArea.
   b. MyTextArea is a superclass of the extends class.
   c. JTextArea is a superclass of MyTextArea.
   d. MyTextArea is a subclass of the JText class.

e. JTextArea is a class of MyTextArea.

7. Which one of the below lines declares myList as a potential instance of the JList class?
   a. myList JList;
   b. JList myJList;
   c. JList myList;
   d. myJList JList;
   e. JList myList.

8. Given the following code: **public class MyJSlider extends JSlider { …**
   which one of the below would be the name of the file that contains this program for compilation?
   a. MyJslider.java
   b. JSlider.java
   c. MyJSlider.javax
   d. myJSlider.java
   e. MyJSlider.java

9. Which of the following lines would most likely add a JScrollPane object to a JPanel object?
   a. JPanel.add(JScrollPane);
   b. JPanel.add(myJScrollPane);
   c. myJPanel.add(JScrollPane);
   d. JScrollPane.add(JPanelObject);
   e. myJPanel2.add(myJScrollPane1);

10. A Java JApplet program has two methods written in the class. The methods are not nested. What is the total number of braces, { and } added together, that are needed for this program.
   a. 9
   b. 6
   c. 3
   d. 4
   e. 2

11. A programmer intends to use the TableColumn class, which is located in the table package. Which statement below is correct to use in the program?
   a. import javax.swing.table.TableColumn
   b. javax.swing.table.TableColumn;
   c. import javax.swing.TableColumn;
   d. import javax.swing.table.TableColumn;
   e. import TableColumn;

12. Which of the following most likely would be used to make a JButton object invisible?
   a. JButton.setBackground(invisible);
   b. myButton.setVisible(opaque);
   c. myButton.setVisible(false);
   d. JButton.SetBackground();
   e. MyButton.setInVisible(true);

13. Which of the following is a correct class definition?
   a. public class StandardJFrame extends JApplet }
   b. public StandardJFrame is-a JFrame {
   c. public class MyLabel extends jlabel {

d. class public MyFrame extends JFrame {
e. public class StandardJFrame extends JFrame {

14. Which of the below shows a correct form of a method?
a. public void int start() {statements}
b. public class stop() {statements}
c. public void Init() {statements}
d. public void stopRunning() {statements}
e. public init() destroy() [statements]

Appendix B

1. Which of the following lines most likely would be used to create a shorthand notation for the compiler to locate the JFrame class, which is built-in to Java?
   a. import ../class/JFrame;
   b. access JFrame.class;
   c. import javax.swing.JFrame;
   d. import java.awt.JFrame.class;
   e. append javax.swing.JFrame;

2. Which of the following lines most likely would be used to construct an instance of the JButton class?
   a. JButton = new JButton("Hello");
   b. myButton = new JButton("Click Me");
   c. Button = new JButton("Hello");
   d. myButton = JButton.class("Hello");
   e. myButton = new JButton("Click Me").

3. Which of the following lines most likely would be used to add a JCheckBox object to a content pane?
   a. getContentPane.Add(myJCheckBox);
   b. container.Add(JCheckBox.Object);
   c. add(container.JCheckBox);
   d. getContentPane().add(myBox);
   e. add(myJCheckBox);

4. Which of the following lines most likely overrides a method that is contained in the Applet class?
   a. public Void stop{} { lines of Java code here }
   b. public void Stop(){ lines of Java code here }
   c. public void stop() {lines of Java code here }
   d. Public Void Stop() (lines of Java code here)
   e. Public void stop() { lines of Java code here }

5. Which of the following sequences is correct?
   a. declare a JTextField object, construct a JTextField object, add a JTextField object to a container.
   b. construct a JTextField object, declare a JTextField object, add a JTextField object to a container.
   c. declare a JTextField object, add a JTextField object to a container, construct a JTextField object.
   d. add a JTextField object to a container, declare a JTextField object, construct a JTextField object.
   e. add a JTextField object to a container, construct a JTextField object, declare a JTextField object.

6. Given the line, **public class MyTextArea extends JTextArea {**, which of the following statements is correct?
   a. JTextArea is a subclass of MyTextArea.
   b. MyTextArea is a superclass of the extends class.
   c. JTextArea is a superclass of MyTextArea.
   d. MyTextArea is a subclass of the JText class.

e. JTextArea is a class of MyTextArea.

7. Which one of the below lines declares myList as a potential instance of the JList class?
   a. myList JList;
   b. JList myJList;
   c. JList myList;
   d. myJList JList;
   e. JList myList.

8. Given the following line in a program: **public class MyJSlider extends JSlider { …** , which one of the below would be the name of the file that contains this program for compilation?
   a. MyJslider.java
   b. JSlider.java
   c. MyJSlider.javax
   d. myJSlider.java
   e. MyJSlider.java

9. Which of the following lines would most likely add a JScrollPane object to a JPanel object?
   a. JPanel.add(JScrollPane);
   b. JPanel.add(myJScrollPane);
   c. myJPanel.add(JScrollPane);
   d. JScrollPane.add(JPanelObject);
   e. myJPanel2.add(myJScrollPane1);

10. A Java JApplet program has two methods written in the class. The methods are not nested. What is the total number of braces, { and } added together, that are needed for this program.
   a. 9
   b. 6
   c. 3
   d. 4
   e. 2

11. A programmer intends to use the **TableColumn** class, which is located in the **table** package. Which statement below is correct to use in the program?
   a. import javax.swing.table.TableColumn
   b. javax.swing.table.TableColumn;
   c. import javax.swing.TableColumn;
   d. import javax.swing.table.TableColumn;
   e. import TableColumn;

12. Which of the following most likely would be used to make a JButton object red?
   a. JButton.setIt(Red);
   b. myJButton.setBackground(red.Color);
   c. myJButton.setBackground(red);
   d. myJButton.setBackground(Color.red);
   e. MyJButton.setBackground(Color.red);

[i] The programmed instruction tutoring systems are freely available on the web, together with instructional course material, interteaching reports, and free open source software for the tutor. Each stage of the tutor is separately accessible, and the best way to understand the tutor's design is to run it in a browser. To run

the tutor, your browser must be enabled with Java JRE 5, which may be downloaded from Sun Microsystems, Inc. How to do that is explained within the online material that supports the Java tutor, as given here: http://nasa1.ifsm.umbc.edu/learnJava/tutorLinks/TutorLinks.html

[ii] The link to run the program is given in the tutor instructions, and it is given here: http://userpages.umbc.edu/~emurian/learnJava/swing/tutor/v2/start/MyProgram.html

Contact Information:

Henry H. Emurian
Information Systems Department
College of Engineering and Information Technology
UMBC
ITE 420
1000 Hilltop Circle
Baltimore, Maryland 21218
Email: Emurian@umbc.edu
Web: http://nasa1.ifsm.umbc.edu/
Voice: 410-455-3206

Advertising in the Behavior Analyst Today

Advertising is available in The Behavior Analyst Today. All advertising must be paid for in advance. Make your check payable to Joseph Cautilli. The ad copy should be in our hands at least 3 weeks prior to publication. Copy should be in MS Word or Word Perfect, RTF format and advertiser should include graphics or logos with ad copy.

The prices for advertising in one issue are as follows:

1/4 Page: $50.00 1/2 Page: $100.00 Full Page: $200.00

If you wish to run the same ad in both issues for the year, you are eligible for the following discount:

1/4 Pg.: $40 - per issue 1/2 Pg.: $75 - per issue Full Page: $150.00 - per issue

An additional one-time layout/composition fee of $25.00 is applicable

For more information, or place an ad, contact Halina Dziewolska by phone at (215) 462-6737 or e-mail at: halinadz@hotmail.com

Copyright of Behavior Analyst Today is the property of Joseph D. Cautilli and its content may not be copied or emailed to multiple sites or posted to a listserv without the copyright holder's express written permission. However, users may print, download, or email articles for individual use.