IDEA GROUP PUBLISHING
701 E. Chocolate Avenue, Hershey PA 17033-1117, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com
ITP4152

Enhanced Learning On A Programmed Instruction Tutoring System For JAVA[1]

Henry H. Emurian
Information Systems Department, UMBC, Baltimore, Maryland, Tel: 410-455-3655, Fax: 410-455-1073, emurian@umbc.edu

Ashley G. Durham
Centers for Medicare adn Medicaid Services, Baltimore, Maryland, Tel: 410-786-5775, Fax: 410-786-0271, adurham@cms.hhs.gov

INTRODUCTION

Our previous research reported the classroom application of a web-based tutoring system for teaching a Java Applet, which is a computer program that runs in a browser, to information systems majors (Emurian & Durham, 2001). The tutoring system was based upon principles of programmed instruction that leads the learner, through a process of successive approximations, from understanding atomic elements of a program to mastery of a serial stream of Java items that constitute an error-free program. The research showed that experience with the tutoring system, together with subsequent classroom instruction, produced dependable improvements in students' self-reports of confidence in the use of Java symbols. It was the case, however, that the students' writing of an error-free Java Applet did not always carry over from the tutor itself to a later assessment occasion. In fact, only two of 12 learners were able to write the program correctly immediately after completing the tutor. This outcome was observed despite the fact that one error-free production of the program was required to exit the tutor. Although the tutor presented explanations of the code, together with multiple-choice tests of the meaning of individual items of code and rows of code, a more robust transfer of training was anticipated between the tutor and subsequent assessments of retention of the program.

The present study intends to enhance student learning and retention of the Java Applet under consideration by using the previous study results as a baseline for comparison. This methodology to programmatic modifications and evaluations exemplifies systematic replication (Sidman, 1953), in which selected independent variables are adjusted to potentiate an effect that has practical rather than statistical significance. In the present study, the objective of the modifications is to improve all students' performance across four assessment occasions during a semester-long course. This methodology also substantiates the generality of the prior findings, when the tutor is administered to a different group of learners, and it demonstrates the reliability of the previous learning effects, when observed under a somewhat different, but related, set of conditions.

TUTORING SYSTEM DESIGN

Details about the tutoring system have been presented elsewhere (Emurian, Hu, Wang, & Durham, 2000), and the system is freely available on the Web (http://webct.umbc.edu/public/JavaTutor/index.html). The system consists of a series of Java applets embedded within the WebCT course management software, and illustrations of the several user interfaces and learning stages are presented in Emurian and Durham (2001). The performance data in the latter study also serves as the baseline for interpreting the present modifications to the system.

The objective of the programmed instruction tutoring system is to teach a learner to construct a stream of 33 items of Java code that constitutes a program to display a text string in a Netscape browser window. The program under investigation in the present study is identical to the program presented in our previous work, with the modification that some atomic elements (i.e., items or units) in the previous study were combined to yield 24 atomic units. An atomic unit was the smallest Java symbol or group of symbols that was presented for learning and testing. For example, in the previous tutor, the ***add*** symbol and the ***(myLabel)*** symbol were separate items, and they were combined to ***add(myLabel)*** in the present version of the tutor. The rationale was to improve the explanation of an item by grouping a method name and its argument into a single item to be learned.

The pedagogical approach taken is first to specify the program to be learned, and then to craft a series of programmed instruction steps that progress to that goal. The outcome of these successive approximations to mastery (Sulzer-Azaroff & Mayer, 1991) is a learner's tested understanding of the meaning of each of the 24 atomic units and the interrelationships among the 33 total items in the writing of the Java program.

Overview

The learner progresses through the tutor in six stages: (1) symbol familiarity, (2) symbol identification, (3) item learning, (4) row familiarity, (5) row learning, and (6) program learning. The item learning and row learning interfaces were modified, as described next.

Figure 1 presents an example of the item interface. Interactive events are sequenced by the learner's use of the buttons, at the bottom of the interface, that are selectively enabled and disabled to regulate progress through this interface. First the learner is shown the Java symbol to enter ("Show Java"). Next are presented an item's explanation ("Explain it") and a multiple-choice test on the item's meaning ("Test"). If the learner then enters the item correctly in a keyin box, the next item is displayed. Otherwise, the cycle repeats until the item is learned.

Figure 1: An example of the item interface. The learner enters the Java item, by recall, in the keyin box

When the learner enters the last item correctly, the row familiarity interface is presented, followed by the row learning interface. That latter interface, which is almost identical to the row interface in the previous study, requires three iterations in each of which ten successive lines of code are entered. In the present modification, however, the third iteration requires the entire ten lines to be entered correctly in order to progress to the final interface. That is, whenever the learner selects an option to observe the code in a row in order to enter the row correctly, all rows are reset, and the learner starts again from the first row. This contrasts with the previous row interface in which an error could be corrected on a row without the requirement to repeat the code from the beginning. After the error-free iteration, the learner progresses to the final interface in which the entire program is entered in a text editor emulation window.

PROCEDURE

The tutor was presented as the first exercise in a 7-week graduate course (Summer, 2001) entitled "Graphical User Interface Systems Using Java." The three-hour class met twice each week, for 14 periods. There were 17 graduate students in Information Systems in the course (eight females, median age = 25.5; nine males, median age = 28). Prior to using the tutor, each student completed a questionnaire that presented two 5-point rating scales. The first scale assessed the student's prior experience with Java, where the scale anchors were ***1 = No experience. (I am a novice in Java.)*** to ***5 = Extensive experience. (I am an expert in Java.)***. The second scale assessed the student's confidence in being able to use each of the 24 Java items to write a Java computer program, where the scale anchors were ***1 = Not at all confident. I do not know how to use the symbol.*** to ***5 = Totally confident. I know how to use the symbol.*** The student was also asked to write a Java Applet, entered into a WebCT text area that was saved for analysis, to display a text string as a Label object in a browser window.

At the conclusion of the three hours that were allotted to the tutoring system or whenever a student finished the tutor prior to that time, a post-tutor questionnaire was completed. This questionnaire repeated the above confidence assessment, and it also included writing the Java Applet into the WebCT text area. A third 5-point rating scale assessed the student's overall reaction to the tutor, where the scale anchors were ***1 = Totally negative. I did not like the tutor.*** to ***5 = Totally positive. I liked the tutor.*** The students were then dismissed from the class, and the tutor continued to be available for those students who were motivated to access the tutor outside of class.

During the second period, which occurred five days later, the instructor discussed the Applet code with the students using a lecture format ("chalk and talk"). The students entered the code into a ***Unix*** text editor at the time the items were presented and discussed. The ***www*** directory tree and ***HTML*** file were also presented and discussed. The students then compiled the Java code and ran the Applet in a Netscape Communicator browser by accessing the ***HTML*** file as a ***URL*** on the Web. To foster collaborative learning, the students were encouraged to help each other and to seek help from the instructor and course assistant as needed. This part of the classroom experience was based upon a modification of the Personalized System of Instruction (Keller, 1968), which includes interpersonal interactions as a further means of learning and competency testing (Ferster & Perrott, 1968). After all students ran the Applet on the Web, they again completed the confidence ratings and the writing of the Applet code in the WebCT text area. This identical assessment was repeated during the fourteenth period.

The modification to the tutoring system also included the addition of a ***brief row tutor***. This interface was similar to the first pass through the row learning interface in the full tutoring system. Whenever an error was made on any row in the brief row tutor, the learner was given the opportunity to view the correct code for that particular row and to enter the code repeatedly until the row code was accurate. The purpose of this interface was to provide additional rehearsal and overlearning of the Java Applet at different temporal occasions throughout the course. This tutor was administered on periods three, seven, and ten. It was administered at the beginning of these classes, and 30 minutes were allotted, sufficient for all students to complete this tutor.

RESULTS AND DISCUSSION

At the conclusion of the time allotted for completing the full Java tutor during the first period, 11 students ("Completers") had finished all parts of the tutor, and six students ("Non-Completers") were working on the row learning (stage 5) or program learning (stage 6) interface. The data analysis is reported as a between-group comparison between these two groups of students.

Non-parametric techniques were applied to self-reported ordinal data, and parametric techniques were applied to ratio data and difference scores. The self-report scales are Likert-like scales (Aiken, 2000). The statements in a scale were selected by their face validity, intended as a structured interview rather than as components of a psychometrically rigorous assessment instrument. This approach is consistent with the use of such scales to solicit preference information.

A comparison of the experience ratings did not support differences between the group of Completers (median = 1, interquartile range = 1-2) and the group of Non-Completers (median = 1, interquartile range = 1-2), Kruskal-Wallis chi-square = 0.01, $p > .90$. The tutoring system provided records of errors made on the first interactive interface, which was the symbol familiarity interface requiring the learner to copy a displayed Java item into a keyin box. This interface was intended to provide practice on the actions of typing the code correctly, and an error was recorded whenever an entered item differed from the displayed item to be copied. **Figure 2** presents box-plots of errors on this interface for the two groups. The difference in mean errors was marginally significant between the Completers (mean = 0.8) and the Non-Completers (mean = 2.0), $F(1,15) = 3.44$, $p < .09$.

Figure 2: Box-plots of errors on the symbol familiarity interface for completers and non-completers

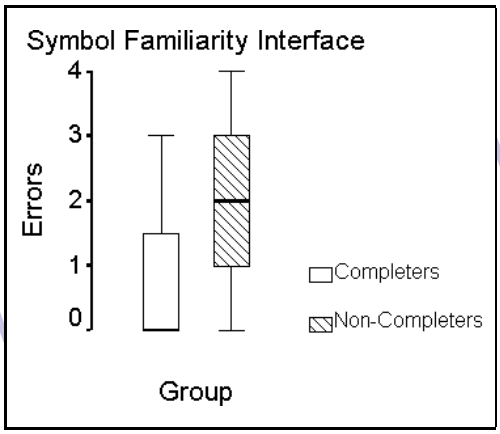
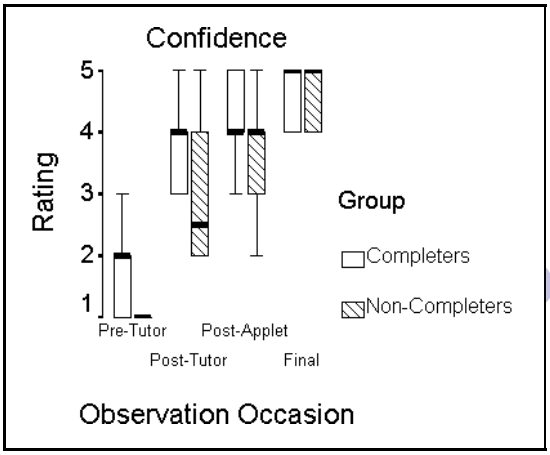


Figure 3 presents box-plots of confidence ratings for both groups across the four assessment occasions. The figure shows graphically that the ratings of confidence increased for both groups across all occasions. This is evidenced graphically by the increases in median confidence and by shifts in the ranges across all occasions. A test of trend, undertaken by comparing the six successive differences between medians with a population of zeros, was significant, $F(1, 10) = 8.57$, $p < .02$. The figure suggests that the Completers showed greater confidence ratings, in comparison to the Non-Completers, over the first three occasions. The difference between the medians on the Pre-Tutor assessment was significant, Kruskal-Wallis chi-square = 4.57, $p < .04$. The difference between the medians on the Final assessment was not significant, Kruskal-Wallis chi-square = 0.22, $p > .50$. These delimited tests were undertaken because the sample size is small and because

Figure 3: Box-plots of confidence ratings for completers and non-completers across the four observation occasions



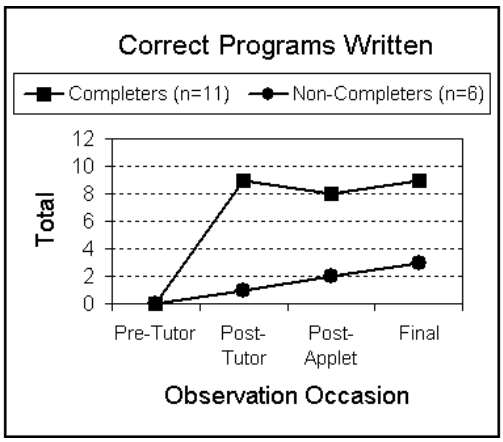
parametric tests may not be appropriate for ordinal data in a factorial design, i.e., Group by Occasion (Maxwell & Delaney, 2000).

Although the use of self-reported data are controversial in behavior analysis (Critchfield, Tucker, & Vuchinich, 1998), conceptualizing these data as verbal behavior (Skinner, 1957) at least allows the communication of contextual information that may be useful in the interpretation of a learner's prior history and current status. Furthermore, by obtaining similar ratings on several different occasions, the change in a "descriptive autoclitic" response, where a speaker estimates the strength of a skill, may be noted in relationship to the learner's history with the tutoring system presented here (Catania, 1998).

Figure 4 presents the total number of correct Java programs that were written into the questionnaire over the four assessment occasions for both groups. The figure categories are discrete, and lines are drawn for clarity of interpretation. Program accuracy was judged independently by two Java instructors who showed 100% agreement. There was no feedback given for incorrect responses, and there was no opportunity to compile the code to test for compilation errors and to revise the code. The assessment, then, consisted only of the learner's skill in entering a correct serial stream of Java items that would produce a Label object in an Applet container.

Figure 4 shows that no learner wrote a correct program during the Pre-Tutor assessment. Immediately after completing the tutor, which required one accurate construction of the entire program, nine

Figure 4: Total number of correct Java programs written into the questionnaire by completers and non-completers across the four assessment occasions



Completers entered the code correctly during the Post-Tutor assessment, and one Non-Completer entered the code correctly. After receiving classroom instruction, which involved writing and compiling the program and running the Applet, eight Completers and two Non-Completers entered the code correctly during the Post-Applet assessment. At the end of the course, nine Completers and three Non-Completers entered the code correctly during the Final assessment. There were only two instances in which a correct program written by a learner on an early assessment was not written correctly on a later assessment.

The assessment of writing the code correctly was undertaken without benefit of observing compile-time or run-time errors, which are obviously important ingredients of a professional's program development and testing. Nevertheless, the present modifications to the tutoring system resulted in robust increases in correct serial constructions in contrast to our previous study (Emurian, et al., 2001) where very few learners showed similar transfer of learning between the tutoring system and subsequent assessment occasions. The change in skill observed in the present study was most pronounced between the Pre-Tutor and Post-Tutor occasions, and only a few learners thereafter were not able to produce the serial stream accurately. This improvement in performance was observed in relationship to the modifications to the tutoring system that included three presentations of the brief row tutor.

Rather than concluding that this outcome is problematic, it may be more reasonable to suggest that even a professional programmer's "baseline" of code construction contains errors that might be readily overcome with the feedback provided by a system during program development, possibly with the assistance of an Integrated Development Environment (IDE). In contrast to such an approach, this study focused on the serial stream of code as it might be mastered in a textual learning paradigm (Li & Lewandowsky, 1995), and it would be informative to compare the present outcome with a baseline of errors exhibited by expert programmers.

The ratings of overall satisfaction with the tutor for the Completers were as follows: median = 5, interquartile range = 3-5. Ratings for the Non-Completers were as follows: median = 5, interquartile range = 3-5. The difference between the medians was not significant, Kruskal-Wallis chi-square = 0.00, $p > .99$. In general, the students reported high levels of satisfaction with the tutoring system.

As indicated previously, the present study followed the methodology of systematic replication (Sidman, 1953). Rather than use a "control group" for comparison, the tutoring system as an "independent variable" was modified by focusing on factors that were considered instrumental to enhance the learning effects, and all prior observations are considered the "control" observations. The greatest change in learning and self-reported confidence occurred in the present study between the Pre-Tutor and Post-Tutor assessment occasions, at least for the Completers, and this outcome is consistent with the power function of learning (Lane, 1987). This outcome was also observed in the previous study, and the effect was stronger in the present case. This shows the reliability of the learning effect over a broader population of learners.

Since the learners in both studies were similar in terms of experience and demographics, the effects observed in the present study are confidently attributable to the tutoring system rather than to between-group bias. In fact, even following a demonstration of relative effectiveness in a statistical paradigm of decision making, any further observation of the utility of a tutoring system is by definition undertaken with a different learner at a different point in time. It is for this reason that systematic replication is favored as an economical and demonstrably effective research methodology when a series of programmatic modifications is intended.

The confidence ratings for both Completers and Non-Completers were similar at the conclusion of the course as were the ratings of satisfaction with the tutor. It was also the case, however, that the behavior of the Non-Completers differed in important ways from that of the Completers. By the end of the course, only three of the six

Non-Completers were able to write the Java Applet correctly during the final assessment. That the group of Non-Completer students came to the course with less readiness for learning Java than did the Completer students was indicated by the observation that the former group showed more errors on the first symbol familiarity interface. That interface required only copying the Java item into a keyin box, showing the importance of basic data entry skills that may set the occasion for more advanced learning of the meaning of Java items. It should not always be assumed, then, that students with similar intellectual abilities are equally prepared to master a programming language. There are fundamental skills that all learners must possess, and programmed instruction provides a series of cumulative experiences that allow adequate preparation for each and every successive stage in the learning process.

Research in the behavior of computer programming and program comprehension ranges from early evaluations of conditional constructions (Sime, Green, & Guest, 1973) to recent simulations of memory representations by expert programmers (Altmann, 2001). These studies, to include evaluations of computer-based tutorials for the development of programming skill (Anderson, Corbett, Koedinger, & Pelletier, 1995), often assume that the learner brings at least some prior familiarity with the knowledge domain to the task or tutorial. In contrast, the present work falls within the stream of instructional technology that is based on programmed instruction, which assumes minimal prior history and which leads the learner to a competency criterion through successive incremental steps to task mastery (Anger, Rohlman, Kirkpatrick, Reed, Lundeen, & Eckerman, 2001; Bitzer, Braunfelf, & Lichtenberger, 1962; Holland, 1960; Skinner, 1958). This latter approach intends to generate the necessary prior history as an integral component in the instructional design technology, thereby making the competency outcome accessible to a wide range of learners possessing different degrees of initial familiarity with the knowledge domain. An important extension of this work, then, is the application of programmed instruction technology to populations of learners who may select career paths other than information technology and resource management. When the steps involved in the development of a skill can be enumerated, a computer-based tutoring system may serve as a generic teacher for anyone who is motivated to acquire the skill.

ENDNOTE

1 This paper represents the author's views and not those of the Centers for Medicare and Medicaid Services (CMS).

REFERENCES

Aiken, L.R. (2000). *Psychological Testing and Assessment* (pp. 250-251). Needham Heights, MA: Allyn and Bacon, Inc.

Altmann, E.M. (2001). Near-term memory in programming: A simulation-based analysis. *International Journal of Human-Computer Studies*, *54*, 189-210.

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

Anger, W.K., Rohlman, D.S., Reed, R.R., Lundeen, C.A., Eckerman, D.A. (2001). cTRAIN: A computer-aided training system developed in SuperCard for teaching skills using behavioral education principles. *Behavior Research Methods, Instruments, & Computers*, *33*, 277-281.

Bitzer, D.L., Braunfield, W.W., & Lichtenberger, W.W. (1962). PLATO II: A multiple-student, computer-controlled, automatic teaching device. In J.E. Coulson (Ed.) *Programmed Learning and Computer-Based Instruction* (pp. 205-216). New York: Wiley & Sons.

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.

Critchfield, T.S., Tucker, J.A., & Vuchinich, R.E. (1998). Self-report methods. In K.A. Lattal & M. Perone (Eds.), *Handbook of Research Methods in Human Operant Behavior* (pp. 435-470). New York: Plenum Press.

Emurian, H.H., & Durham, A.G. (2001). A personalized system of instruction for teaching Java. In M. Khosrowpour (Ed.) *Managing Information Technology in a Global Economy* (pp. 155-160). Hershey: Idea Group Publishing.

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.

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

Holland, J.G. (1960). Teaching machines: An application of principles from the laboratory. *Journal of the Experimental Analysis of Behavior*, *3*, 275-287.

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

Lane, N.E. (1987). *Skill Acquisition Rates and Patterns: Issues and Training Implications*. New York: Springer-Verlag.

Li, S., & Lewandowsky, S. (1995). Forward and backward recall: Different retrieval processes. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *21*, 837-847.

Maxwell, S.E., & Delaney, H.D. (2000). *Designing Experiments and Analyzing Data*. Mahwah, NJ: Lawrence Erlbaum Associates.

Sidman, M. (1953). *Tactics of Scientific Research*. New York, NY: Basic Books.

Sime, M.E., Green, T.R.G., & Guest, D.J. (1973). Psychological evaluation of two conditional constructions used in computer languages, *International Journal of Man-Machine Studies*, *5*, 105-113.

Skinner, B.F. (1958). Teaching machines, *Science*, *128*, 969-977.

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

Sulzer-Azaroff, B., & Mayer, G.R. (1991). *Behavior Analysis for Lasting Change*. Orlando, FL: Holt, Rinehart and Winston, Inc.