Programmed Instruction and Interteaching Applications to Teaching Java™:
A Systematic Replication

Journal of Behavioral Education, in review

Henry H. Emurian¹
&
Peng Zheng

Information Systems Department
College of Engineering and Information Technology
UMBC
1000 Hilltop Circle
Baltimore, Maryland 21218

Voice: 410-455-3655
Fax: 410-455-1073
Email: Emurian@umbc.edu

RUNNING HEAD: Programmed Instruction and Interteaching

¹ Corresponding author.
Programmed Instruction and Interteaching Applications to Teaching Java™: A Systematic Replication

ABSTRACT

Sixteen undergraduate students in a Java computer programming course completed a programmed instruction tutor and an interteaching session in furtherance of mastering an applet program as a first technical training exercise. The tutor instructional frames were enhanced to support rule governed learning. Prior to the interteaching session, students completed a brief rule tutorial designed to provide exemplars of a test of rule governed performance that was administered three times over the course of this initial learning. Students showed improvements in test performance and software self confidence, although the relative gains observed during interteaching did not always transfer to a subsequent quiz for individual students. The reported history of the students was found to relate to the knowledge that students acquired from the several instructional tactics. This systematic replication extends the generality and reliability of the previous work, and it shows the value of using several different instructional media successively to help students achieve mastery.

KEYWORDS: Programmed instruction, interteaching, Java training.

INTRODUCTION

This paper reports a replication of instructional tactics presented in Emurian (2007a) to teach undergraduate students a simple Java applet as a first technical exercise in a mid-level computer programming course. An applet is a Java computer program, and the version of the applet in the present work displayed a text string in a browser. The previous study was based upon classroom applications of programmed instruction, which is individualized instruction (Molenda, 2008), and interteaching, which is a type of
collaborative peer tutoring (Boyce & Hineline, 2002). Emurian, Holden, and Abarbanel (2008) offer arguments in support of such a multi-phased and design-based program (Wang & Hannafin, 2005) in instructional technology. The two instructional tactics to teach Java are also described in Emurian (2008), Emurian, Wang, and Durham (2003), and Emurian and Durham (2003). The current study constitutes a systematic replication (Sidman, 1960) of our previous work in that it is intended to show the generality and dependability of the behavioral processes under investigation. The title of the undergraduate course in the current study was “Graphical User Interface Systems Using Java.” The prerequisite was one prior computer programming course, and most students satisfied that requirement with an introductory Java course required by majors in Information Systems at UMBC.

In the Emurian (2007a) study, rule governed learning was assessed by the administration of 12 multiple-choice test items on the following five successive occasions: pre-tutor, post-tutor, lecture, interteaching, and a graded quiz. Although improvement was observed across the first four occasions without feedback on test accuracy, after the interteaching session, feedback was given to the students regarding the correct answers on the 12 test items. This occasioned further improvement in rule governed test performance observed on the graded quiz. Although such feedback may occasion improved future performance on a direct replication of a test, the fact that any test error occurred following programmed instruction may be taken as evidence that the tutor’s instructional frames together with the embedded multiple-choice tests did not adequately occasion the learning of principles that could be applied to solve novel problems, which the 12 rule questions were intended to assess.
Davis, Bostow, and Heimisson (2007) reported the inclusion of abstract statements of a behavioral relation (i.e., a rule) in many frames of a programmed instruction tutor designed to promote generalization of what was taught in the tutor. Accordingly in the present study, rather than giving students feedback on the rule questions, the instructional frames of information in the tutor were revised to ensure that “rules” were explicitly stated. An example of a rule is as follows: *A class file in Java* always begin with a capital letter. Figure 1 presents an instructional frame that states a rule for naming a class. Additionally, at the conclusion of the Items stage of the tutor, which taught the syntax and semantics of 37 items in the program, a figure with exemplars was presented to assist the learner in being able to classify seven groups of terms in the Java programming language, such as *keyword*, *method*, *operator*, and *separator*. The figure is presented in Figure 2. These tactics were implemented to foster “same” relations of terms (e.g., O’Hora, Barnes-Holmes, Roche, & Smeets, 2004) encompassed within the context of a hierarchical relational frame (Hayes, Fox, Gifford, Wilson, Barnes-Holmes, & Healy, 2001, p. 37).

The replication in the present study was undertaken with these revisions to the content of the Java programmed instruction tutor.
The replication’s instructional tactics were also influenced by a growing body of research involving testing. The impact of frequent testing on learning and retention by students (i.e., the “testing effect”) is now well documented (Karpicke & Roediger, 2008; McDaniel, Roediger, & McDermott, 2007; Roediger & Karpicke, 2006a). This evidence suggests a shift from testing as only an assessment of learning to testing as a technique of information retrieval that enhances learning (Roediger & Karpicke, 2006b). Even testing with multiple-choice tests without immediate feedback has a positive effect on retention (Butler, Karpicke, & Roediger, 2007).

In that latter regard, our previous work showed that providing feedback on the fourth of five successive administrations of an identical multiple-choice test may have assisted learners in mastering all test items, although the programmed instruction and collaborative peer tutoring (i.e., interteaching) were also provided to facilitate that outcome over successive learning and testing occasions (Emurian, 2007a). Moreover, our prior classroom applications were intended to provide a series of instructional experiences, beyond the “testing effect,” that would lead all students to complete mastery of the items on a test of rule governed learning (e.g., Emurian, Holden, & Abarbanel, 2008). The replication, then, builds upon this stream of work to include consideration of the potential transfer of learning from a brief tutorial having content and multiple-choice test items similar, but not identical, to subsequent tests of rule governed learning that assessed identical concepts but that were administered without benefit of feedback (cf. Chan, McDermott, & Roediger, 2006). The brief tutorial used an “answer-until-correct” approach similar to the programmed instruction tutor. However, in contrast to Butler et al. (2007), in which a multiple-choice test was sometimes repeated until the correct answer
was selected, when a test choice was incorrect in the present study, the student was recycled for an additional opportunity to study the content before the test was again administered. The assumption was that an incorrect choice on a test was an indication of inadequate studying, and that performance needed repetition as well as the subsequent action of selecting an answer on the test. Finally, this brief tutorial was required as homework prior to the interteaching session in the course.

The replication, then, used an exemplar test with immediate feedback under the assumption that such skill will transfer to a similar, but not identical, test where no feedback was given over three assessment occasions (i.e., pre-tutor, post-tutor, and graded quiz).

METHOD

Materials

Materials used in the study are freely available on the web, and links to that material are presented in Appendix A. The programmed instruction tutoring system has been described in detail elsewhere (Emurian, 2008), and the best way to understand that part of the procedure in this study is to run the online tutoring system, which is designed for learners having minimal prior programming experience. There were 23 unique elements in the program, which led to 37 items of code and 11 lines of code to produce the applet that was taught as the first technical exercise.

Regarding the 12 rule questions that appeared on the pre-tutor and post-tutor questionnaires, the students were informed that those questions were eligible to appear on the first quiz. The following statement appeared on the interteaching questionnaire with respect to the 12 rule questions and other requirements: “The below identical or similar
questions may appear on the next quiz. The questions embedded in the Java tutor are also eligible to appear on the next quiz.”

Subjects

Undergraduate students in an elective course in Java participated in this research, which was determined to be exempt from informed consent requirements. All course requirements were posted on the syllabus, which was available online to prospective students. The class, in which 22 students were enrolled, met in the Spring of 2008 for 2.5 hrs each week across a 14-week semester. Only the 16 students who completed all parts of the initial instructional tactics and assessments, which took place over the first three class periods, are included in the data analysis. Reasons for a student’s omission include network connection failures for data transmission, failure to complete a questionnaire, and missing a class. There were 12 males (mean age = 22.3, range = 18-27) and 4 females (mean age = 20, range = 19-21) included in the analysis.

Procedure

Prior to working on the Java programmed instruction tutoring system, each student completed a background questionnaire giving sex, age, and number of prior computer programming courses taken. Students also rated their experience in overall computer programming and Java programming on a 10-point Likert-type scale where 1 = No experience. I am a novice. to 10 = Extensive experience. I am an expert. Figure 3 presents boxplots of the number of programming courses taken previously, Java experience, and overall programming experience reported by the students. The importance of these data, as they relate to the outcomes of programmed instruction, interteaching, and rule test performance will be presented below.
Ratings of software self-efficacy were based upon the following question for each of the 23 unique elements in the program: “How confident are you that you understand and can use the following symbol now to write a Java program?” A 10-point Likert-type scale was used where 1 = No confidence. to 10 = Totally confident. I am an expert. For each rule question, the student rated his or her confidence that the selected answer was correct, where 1 = No confidence to 10 = Total confidence.

Table 1 presents a summary of the contents of the first three classes in the course.

In the present study, a brief rule tutorial was required as homework before the second class. This brief tutorial was similar in functionality to the Items learning stage of the programmed instruction tutor. However, there were only 12 frames, and each frame presented an explanation of a principle that could be applied to solve the 12 multiple-choice rule questions that appeared in the pre-tutor and post-tutor questionnaires and on the quiz. The 12 multiple-choice questions that were embedded in the brief rule tutorial were similar in semantics (“meaning”) to the other questions, but the syntax (“topography”) differed. Students received course credit for completing the brief rule tutorial, and that requirement was on the course syllabus. The order of the questions and the location of the correct answer differed across all testing occasions.
Table 2 presents a question on the rule test that appeared on the pre-tutor and post-tutor questionnaires and on the quiz. Also presented is a similar question that appeared within the brief rule tutorial and on the interteaching record. The order of the questions differed across those occasions.

Non-parametric or robust tests were used in the data analyses, and Z scores were sometimes presented for ease of interpretation. SPSS was used for the calculations.

RESULTS

Figure 4 presents boxplots of errors observed on the 12 rule test questions across successive assessment occasions. Since the brief rule tutorial required the learner to repeat a frame until a corresponding multiple-choice test had been passed, the boxplot for the brief rule tutorial may reflect more than one failed attempt to complete a particular frame. The figure shows graphically that median errors declined over the first four occasions and increased somewhat for the quiz, at least in comparison to interteaching. A Friedman test was significant (chi-square = 38.314, p = 0.000). To assess the magnitude of the changes over successive occasions, a difference score, Di, was computed for each student (Di = 1, n) for the three sets of differences (Dj = 1, 2, 3) obtained over the four successive assessment occasions, not including the brief tutorial errors. A Welch robust test was significant (W = 45.223, p = 0.000). Planned pairwise comparisons, Bonferroni corrected, were significant for Di1 and Di2 (p < .05) and Di2 and Di3 (p < .05). A Wilcoxon Signed Ranks test comparing errors between the interteaching session and the
quiz was not significant (Z = -1.634, p = .102. A similar test comparing errors between the post-tutor occasion and the quiz was significant (Z = -2.692, p = .007).

Figure 5 presents total correct answers on the rule test questions for pre-tutor, post-tutor, and interteaching sessions and for the quiz for the pairs of students who participated together in the interteaching session. Correct answers were used in this figure to represent graphically the correlations under consideration. The figure graphically shows that the two students who worked together on the questions during interteaching generally were identical in the correct answers, with student pairs 2 and 7 being the exceptions. Additionally, the rule test scores during interteaching often did not transfer to the quiz for one or even two of the previously paired students, and sometimes the scores on the quiz were comparatively lower for one or two of the students (pairs 2, 3, 5, 6, 7). Supporting these interpretations are the correlations between the pairs of students: pre-tutor (r = .53, p = .180), post-tutor (r = .32, p = .438), interteaching (r = .98, p = .000), quiz (r = .64, p = .086). Only the correlation during interteaching was significant.

Figure 6 presents boxplots of total errors on the items and lines multiple-choice tests for the tutor and quiz occasions. The figure shows graphically that median errors

Insert Figure 5 About Here

Insert Figure 6 About Here
decreased for items and lines between the programmed instruction tutor and the quiz.

Wilcoxon Signed Ranks tests were significant for items (Z = -3.409, p = .001) and for lines (Z = -3.524, p = .000). The correlations in errors between the two occasions were as follows: items (r = .837, p = .000) and lines (r = .649, p = .007).

**Figure 7** presents boxplots of software self-efficacy ratings across the pre-tutor, post-tutor, and interteaching occasions. Reliability of the scales based upon Cronbach’s Alpha was as follows: pre-tutor (alpha = .99, p = .000), post-tutor (alpha = .99, p = .000), and interteaching (alpha = .98, p = .000). A Friedman test was significant (chi-square = 21.686, p = 0.000). A comparison of Di1 and Di2 was significant (W = 17.093, p = 0.000). The figure shows graphically the large increase in the median rating between the pre-tutor and post-tutor occasion. The range of individual ratings from a low of 1 to a high of 10 is also evident on the pre-tutor occasion.

**Figure 8** presents boxplots of confidence ratings that a selected rule test answer was correct for both correct (“Right”) and incorrect (“Wrong”) answers across the pre-tutor, post-tutor, and interteaching occasions. The figure shows graphically that confidence increased over occasions for both correct and incorrect answers. Because the number of students answering both correctly and incorrectly differed across conditions,
the Kruskal-Wallis test was used. For ratings across the three occasions, a Kruskal-Wallis
test was significant for correct answers (chi-square = 13.692, p = .001) and for incorrect
answers (chi-square = 9.473, p = .009). A Mann-Whitney U test comparing the all ratings
for correct and incorrect answers was significant (Z = -3.531, p = .000).

**Table 3** presents a correlation matrix based upon four measures of the history of
the students (Courses Taken, Java Experience, Programming Experience, and Software
--------------------------------
Table 3 About Here
--------------------------------
Self-Efficacy) and outcomes of rule tests and items tests on several occasions. Too few
errors were observed on the lines test during the quiz to include those data. These data
show that students’ self-reported background and knowledge at the beginning of the
course could predict, with the possible exception of Programming Experience, quiz
performance outcomes for rule test questions and items test questions even after the
several instructional experiences provided to the students.

**Figure 9** presents scatterplots of Java experience and errors on the rule test
questions for the pre-tutor and quiz assessment occasions. Regression lines are also
--------------------------------
Figure 9 About Here
--------------------------------
displayed. This figure graphically amplifies the significant correlations between these
variables presented in Table 3. The figure shows graphically the linear relationship
between the measurements on both occasions. Even though the errors on the quiz
occasion are graphically lower than observed on the pre-tutor occasion, the linear relationship between Java experience and errors continued to hold. In general, as self-reported Java experience at the beginning of the course increased, errors on the rule test decreased on both occasions, although total errors dropped across the two occasions.

**Figure 10** presents scatterplots of pre-tutor software self-efficacy and errors on the items test questions observed on the programmed instruction tutor and on the quiz assessment occasions. Regression lines are also displayed. This figure graphically amplifies the significant correlations between these variables presented in Table 3. The figure shows that as pre-tutor software self-efficacy increased, errors on the items questions in the programmed instruction tutor and on the quiz decreased. The relationships were orderly across the two assessment occasions despite the fact that variability in errors and the magnitude of errors were greater during the tutor in comparison to the quiz.

**DISCUSSION**

The results of this study show that programmed instruction can impact student knowledge as evidenced by decreases in rule test errors and by increases in software self-efficacy as an immediate benefit of completing a tutoring system, with updated instructional frames, having the objective of teaching a simple Java applet. The history of completing multiple-choice tests on the items and lines of code that were embedded in the tutor clearly impacted the errors observed on identical tests administered on a quiz.
two weeks later: test errors were much lower on the quiz in comparison to errors while completing the tutor. Additionally, the study showed that completing a brief rule tutorial and an interteaching session with a set of similar, but not identical rule test questions, reduced rule test errors on the quiz, in comparison to the post-tutor occasion. In general, the several successive instructional experiences improved student performance over those occasions.

This study followed the tactic of systematic replication (Sidman, 1960) as a way to show the generality and dependability of a behavioral process operationalized by a set of antecedent conditions evaluated with respect to effectiveness in changing the behavior of a group of students in a manner consistent with the objectives of a course in the Java programming language. The replication was systematic in that modifications were introduced to the information frames in the programmed instruction tutor, and a separate brief rule tutorial was required as homework prior to interteaching. That latter tutorial also varied the exemplars that were taught in a way intended to foster the development of general principles related to Java. The fact that a new group of students was used also makes this replication systematic with respect to the generality of the outcomes. In fact, in the two classes reported in Emurian (2007a), the median Java experience by the students was 2 for both classes, and in Emurian et al. (2008), the median Java experience was 1 for students in the two classes. The dependability of the behavioral process, however, is indicated by the similarity of the present outcomes, where the median Java experience by the students was 5, to the previous classroom investigations in this stream of applied behavior analysis (e.g., Emurian et al., 2008; Emurian, 2007a).
How to operationalize “studying behavior” continues to pose challenges for educational research and applications, and both constructed recall tests and multiple-choice tests may contribute to learning in programmed instruction (Miller & Malott, 2006). Although retention rehearsal through structured testing may improve retention for the material practiced under a testing condition (Roediger & Karpicke, 2008; Roediger & Karpicke, 2006a), it is less certain that such a strategy benefits a student’s mastery of appropriate learning strategies. The latter include teaching students the techniques of self-regulated learning and practice recall in furtherance of achieving what has been designated as adaptive transfer (Bell & Kozlowski, 2008), an active learning approach that includes teaching how to modify a learned procedure to solve a novel problem. Such an approach may have the effect of teaching students to create and implement their own learn units (Emurian, 2007b; Greer & McDonough, 1999). The contents of a self-supported learn unit, perhaps, assumes that sufficient evidence exists to guide the construction of such a unit in an optimal fashion. However, research by Rohrer and Pashler (2007) and Rohrer, Taylor, Pashler, Wixted, and Cepeda (2005) question traditional assumptions about the importance of sustained practice recall on long-term retention, and they raise issues about what exactly a learner should be practicing. The ongoing development of programmed instruction requires consideration of this important evidence, while acknowledging the challenge of interpreting and synthesizing such a large and growing body of literature (Burton, 2005). In that regard, both programmed instruction and interteaching would perhaps benefit from consideration of the need for automated tutors, such as MediaMatrix (Appendix A), that adapt to the level of a student’s knowledge and skill and to transition students, collaborating or otherwise, in the
acquisition of skills whereby continued learning would not necessarily require such tactics at all (Ray & Belden, 2007). In fact, that is entirely compatible with the current use of programmed instruction and interteaching in our course.

The intent of the repeated practice effects occasioned by the current instructional tactics, to include repeated testing, is consistent with the conclusion of Karpicke and Roediger (2008): “Repeated retrieval induced through testing (and not repeated encoding during additional study) produces large positive effects on long-term retention” (p. 968). Although assessing long-term retention was not an objective in the current study, it was the case, however, that the transfer of learning from the brief rule tutor and interteaching to the quiz was not entirely successful. This was evidenced by the slight increase in median errors on the rule test observed on the quiz in comparison to the interteaching session. Although the difference in median errors was not statistically significant, Figure 4 clearly shows that the range of total errors was greater during the quiz. The explanation, perhaps, is given in Figure 5, which shows total correct answers for pairs of students. Pairs 2, 5, 6, and 7 all show one pair member whose correct answers dropped between interteaching and the quiz. It is plausible that during interteaching such pair members did not master the principle required for a correct quiz answer, and they were simply following the advice of the more informed student. How to insure the quality of interteaching and how to document equivalent knowledge within pairs of students is an ongoing challenge for instructional designers.

Peer tutoring has been defined as “people from similar social groupings who are not professional teachers, helping each other to learn, and learning themselves by teaching” (Topping, 1996, p. 322, cited in Smet, Keer, & Valcke, 2008). The academic
literature on this topic is now extensive, and there are milestone sources such as Slavin (1996) and Jehng (1997). The interteaching orientation of the present work, based upon the behavior analysis by Boyce and Hineline (2002), is readily encompassed within that general definition. The current classroom application used such collaborative peer tutoring on only one occasion, and that was during the interteaching session. In that regard, there may have been value in allowing students to work in pairs during the pre-tutor, post-tutor, and interteaching assessments, although individual performance changes would obviously be problematic to uncover. However, the results (Table 3, Figures 9 and 10) clearly reveal the discrepancy in the instruction’s efficacy between the experienced students and the less experienced peers. This observation, together with the finding by Smet et al. (2008) that peer tutors in an online setting exhibit different tutoring styles (i.e., “motivators,” “informers,” and “knowledge constructors”), complicates the interpretation that peer tutoring may always improve the performance of the less experienced student. When students present for a course with widely differing skills and backgrounds, beneficial peer tutoring may require balancing those factors within pairs and giving explicit instructions to all students regarding how peer tutoring should be undertaken. In that regard, Mayer (2002) suggests ways that a cooperative learning context may be designed to promote meaningful learning (p. 222).

With respect to rehearsal, the approach taken in our classroom applications deliberately incorporates different instructional media as a tactic to repeat the acquisition of knowledge and skill of a simple Java applet. This contrasts with much research on programmed instruction and interteaching that typically focuses upon investigating effects of different ways of managing a learner-tutor interaction in programmed
instruction (e.g., Davis, Bostow, & Heimisson, 2007) or comparing interteaching with other forms of instructional delivery, such as a lecture (e.g., Saville, Zinn, Neef, Norman, & Ferreri, 2006). To repeat a learning objective with different media is to follow the suggestions of Halpern and Hakel (2003) with respect to promoting retention and transfer and Fox and Hackerman (2003) with respect to socially supported interactions. A lecture was also used in the present research as a tool of repetition, but a separate assessment following the lecture was not undertaken here as it was in Emurian (2007a). From the perspective of behavior analysis, an implicit assumption in this multi-modal and replicative approach is that equivalence relations and the emergence of equivalence classes are prominently involved, and Perez-Gonzalez, Herszlikowicz, and Williams (2008) have listed several theoretical descriptions of such considerations that are relevant to interpreting the current work.

Although the role of information and communications technologies in education will continue to evolve even to the point of teaching such skills as collective problem resolution via mediated interactions (Dede, 2007), the need for focused individual skill development using programmed instruction should not be underestimated (Molenda, 2008). An ongoing challenge, however, includes the optimal design of the content of frames – the “explanations” (Wittwer & Renkl, 2008). A challenge for programmed instruction is to develop frames of information that are effective for learning, and that requires a conceptual framework for understanding the effectiveness of instructional explanations. Additionally, consideration must be given to the format of tests, whether short answer or multiple-choice, and the relationship of corrective feedback to long-term retention (Kang, McDermott, & Roediger, 2007).
Similar to our previous observations (Emurian 2007a; Emurian et al. 2008), students in the present study showed increases, over successive assessment occasions, in reported confidence that both the correct and incorrect rule test questions were accurate. It was the case, however, that such confidence was generally lower for the incorrect answers. In terms of a functional account of this descriptive autoclitic verbal performance (Skinner, 1957, p. 313), the student served as both speaker and listener even when the relationship between the antecedent condition (the multiple-choice question) and the reinforcer (self reinforcement) was defective for the incorrect answers. In this sense, the autoclitic serves as an indication of the strength, perhaps, of a future verbal performance such as writing a computer program. How a learn unit formulation (Greer & McDonough, 1999), to include consideration of self-regulated learning, may contribute to an account of such autoclitic verbal behavior has been discussed in Emurian (2007b), and Jaehnig and Miller (2007) suggest that elaboration feedback in programmed instruction, where a learner is given feedback to explain why a test answer is correct or incorrect, may improve learning. That latter feedback may be instrumental in shaping a relational frame of coordination (Hayes, Blackledge, & Barnes-Holmes, 2001) with respect to a descriptive autoclitic performance and the consequences of writing a computer program that is correct or incorrect. Although such feedback was provided for correct multiple-choice answers in the current programmed instruction tutor, an incorrect choice recycled the student through the instructional frame, rather than explaining “why” the choice was incorrect. In terms of instructional design, issues of feedback require additional study, as indicated by Jaehnig and Miller (2007).
Self reports of Java experience and software self-efficacy revealed orderly relationships between the history of the learners and extent of knowledge gained from the programmed instruction tutor, the brief rule tutorial, and the interteaching. Although all students exited the automated instructional systems with equivalent assessed mastery, it is clear from the data presented in Table 3 and in Figures 9 and 10 that the experienced students took away more knowledge from those instructional tactics, in comparison to less experienced students. Additionally, consideration of adaptive learning systems, such as discussed by Ray and Belden (2007), might be considered to offer additional instruction to learners who perhaps show more than one error on an embedded test in the programmed instruction tutor or brief rule tutorial.

The background skill and confidence gained by the programmed instruction tutor sets the occasion for the students’ continued learning through lectures and through learning how to seek out technical solutions to programming challenges. Additionally, the interteaching not only reinforces learning and understanding, but more importantly, perhaps, it also gives students experience in collaborating. In our course, we encourage students to help each other as they write, test, and debug computer programs throughout a semester’s work. It should be acknowledged that both programmed instruction and interteaching do have a role to play in helping many students to acquire skills, and dismissing either is unwarranted (Field, 2007). The design of effective programmed instruction may well require an extensive understanding of operant reinforcement contingencies, as suggested by Davis, et al. (2007), but the content of those contingencies would benefit by considering the contributions to educational tactics of the many
disciplines that may lie outside the scope of a traditional behavior analysis. The same holds true for interteaching.
REFERENCES


APPENDIX A

Links to the material used in this study are presented below.

Java programmed instruction tutoring system
http://nasa1.ifsm.umbc.edu/learnJava/tutorLinks/TutorLinks.html

Questionnaires, interteaching material, and the first quiz
http://www.umbc.edu/~emurian/2008study/

The brief rule tutor
http://userpages.umbc.edu/~emurian/learnJava/swing/tutor/v2/study2008/rules/Tutor.html

Explanations in the programmed instruction tutoring system
http://userpages.umbc.edu/~emurian/learnJava/swing/tutor/v2/study2008/explanations/Explanations.html

Explanations in the brief rule tutor
http://userpages.umbc.edu/~emurian/learnJava/swing/tutor/v2/study2008/rules/explanations/

MediaMatrix

MediaMatrix is an example of a commercial automated instructional that is adaptive and that provides a tactic to foster concept learning and relational networks. Importantly, the assessments of learning not only include multiple-choice tests, but they also include fill-in-blanks, and the writing and identifying of items grouped within a conceptual network:

Table 1

The Sequence of Activities

Class 1
- Pre-Tutor Questionnaire
- Programmed Instruction Tutor
- Post-Tutor Questionnaire

Homework
- Brief Rule Tutorial
- Access Interteaching Preparation Document

Class 2
- Lecture on the program
- Interteaching session on the program
  - Collaborative Questionnaire
  - Brief Rule Tutorial Questions

Class 3
- Quiz
  - Items and Lines Tests from the Java Tutor
  - Rule Tests from the Class 1 Questionnaires
Table 2

<table>
<thead>
<tr>
<th>Pre-Tutor, Post-Tutor, and Quiz</th>
<th>Brief Rule Tutorial and Interteaching</th>
</tr>
</thead>
<tbody>
<tr>
<td>Which of the following lines would most likely add a JScrollPane object to a JPanel object?</td>
<td>Which of the following lines would most likely add a JList object to a JPanel object?</td>
</tr>
<tr>
<td>a. <code>JPanel.add(JScrollPane);</code></td>
<td>a. <code>myBigJPanel5.add(JList);</code></td>
</tr>
<tr>
<td>b. <code>JPanel.add(myJScrollPane);</code></td>
<td>b. <code>myBigJPanel5.add(myLittleJList1);</code></td>
</tr>
<tr>
<td>c. <code>myJPanel.add(JScrollPane);</code></td>
<td>c. <code>JPanel.add(myLittleJList1);</code></td>
</tr>
<tr>
<td>d. <code>JScrollPane.add(JPanelObject);</code></td>
<td>d. <code>JList.add(JPanelObject);</code></td>
</tr>
<tr>
<td>e. <code>myJPanel2.add(myJScrollPane1);</code></td>
<td>e. <code>JPanel.add(JList);</code></td>
</tr>
</tbody>
</table>

An example of a rule test question across the two types of assessments. The underlying principle required to solve the problem is identical, and the principle was emphasized in the brief rule tutorial.
### Table 3
Correlation Matrix

<table>
<thead>
<tr>
<th></th>
<th>JE</th>
<th>PE</th>
<th>SSE</th>
<th>PreTRE</th>
<th>PosTRE</th>
<th>TIE</th>
<th>BTE</th>
<th>IRE</th>
<th>QRE</th>
<th>QIE</th>
<th>Courses</th>
</tr>
</thead>
<tbody>
<tr>
<td>JE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PE</td>
<td>.599*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSE</td>
<td>.528*</td>
<td>.552*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PreTRE</td>
<td>-.377</td>
<td>-.714**</td>
<td>-.416</td>
<td>-.767**</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PosTRE</td>
<td>-.345</td>
<td>-.661**</td>
<td>-.548*</td>
<td>-.875**</td>
<td>.900**</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIE</td>
<td>-.549*</td>
<td>-.534*</td>
<td>-.667**</td>
<td>-.636**</td>
<td>.762**</td>
<td>.715**</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BTE</td>
<td>-.288</td>
<td>-.454</td>
<td>-.368</td>
<td>-.443</td>
<td>.583*</td>
<td>.484</td>
<td>.703**</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IRE</td>
<td>-.523*</td>
<td>-.715**</td>
<td>-.461</td>
<td>-.378</td>
<td>.353</td>
<td>.351</td>
<td>.447</td>
<td>.751*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QRE</td>
<td>-.458</td>
<td>-.797**</td>
<td>-.464</td>
<td>-.610*</td>
<td>.572*</td>
<td>.668**</td>
<td>.524*</td>
<td>.411</td>
<td>.703**</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QIE</td>
<td>-.557*</td>
<td>-.608*</td>
<td>-.510*</td>
<td>-.598*</td>
<td>.844**</td>
<td>.791**</td>
<td>.837**</td>
<td>.490</td>
<td>.354</td>
<td>.662**</td>
<td></td>
</tr>
</tbody>
</table>

JE = Java Experience, PE = Programming Experience, SSE = Pre-Tutor Software Self-Efficacy, PreTRE = Pre-Tutor Rule Test Errors, PostTRE = Post-Tutor Rule Test Errors, TIE = Programmed Instructor Tutor Items Test Errors, BTE = Brief Tutor Test Errors, IRE = Interteaching Rule Test Errors, QRE = Quiz Rule Test Errors, QIE = Quiz Items Test Errors, and Courses = Number of Programming Courses Taken.
Figure 1

This figure shows an example of an instructional frame within the Items stage of the tutor that states an explicit rule in the Java programming language.
Figure 2

The below figure was presented at the conclusion of the Items stage of the programmed instruction tutor. The intent was to facilitate the student’s understanding of the various categories of terms in the Java programming language as they related to the particular program taught in the tutor.
Figure 3

Boxplots of previous courses taken and self-reported Java experience and overall programming experience. The lowest possible value of each boxplot was 1 because at least one programming course was required as a prerequisite.
Boxplots of total errors on the rule test across the five observational occasions. The shaded bars indicate errors on the second set of rule questions. During the brief rule tutorial, each question was repeated as needed in a learning cycle until it was answered correctly.
Figure 5

Total correct answers on the rule test questions across pre-tutor, post-tutor, interteaching, and quiz assessment occasions, respectively. Presented for all occasions are totals for the two students who were paired for the interteaching session, and the markers are black for that session.
Figure 6

Total errors on the items and lines multiple-choice tests observed at the completion of the programmed instruction tutor and on the subsequent quiz that presented the same questions, but in a different order.
Figure 7

Boxplots of confidence ratings of the 23 unique elements of code in the program across three assessment occasions.
Boxplots of confidence ratings in rule test answers that were correct (Right) and incorrect (Wrong) across three assessment occasions.

**Figure 8**

Confidence in Rule Test Answers

Answer
- Right
- Wrong

Rating

Pre-Tutor  Post-Tutor  Inter-Teaching
Figure 9

Scatterplots and regression lines for Java experience and total rule test errors for the pre-tutor and quiz assessment occasions. The size of the marker is increased proportionally where more than value was present at a particular point.
Figure 10

Scatterplots and regression line for software self-efficacy and items test errors observed at the completion of the programmed instruction tutor and on the subsequent quiz. The size of the marker is increased proportionally where more than value was present at a particular point.
Figure Captions

1. This figure shows an example of an instructional frame within the Items stage of the tutor that states an explicit rule in the Java programming language.

2. The below figure was presented at the conclusion of the Items stage of the programmed instruction tutor. The intent was to facilitate the student’s understanding of the various categories of terms in the Java programming language as they related to the particular program taught in the tutor.

3. Boxplots of previous courses taken and self-reported Java experience and overall programming experience. The lowest possible value of each boxplot was 1 because at least one programming course was required as a prerequisite.

4. Boxplots of total errors on the rule test across the five observational occasions. The shaded bars indicate errors on the second set of rule questions. During the brief rule tutorial, each question was repeated as needed in a learning cycle until it was answered correctly.

5. Total correct answers on the rule test questions across pre-tutor, post-tutor, interteaching, and quiz assessment occasions, respectively. Presented for all occasions are totals for the two students who were paired for the interteaching session, and the markers are black for that session.

6. Total errors on the items and lines multiple-choice tests observed at the completion of the programmed instruction tutor and on the subsequent quiz that presented the same questions, but in a different order.

7. Boxplots of confidence ratings of the 23 unique elements of code in the program across three assessment occasions.

8. Boxplots of confidence ratings in rule test answers that were correct (Right) and incorrect (Wrong) across three assessment occasions.

9. Scatterplots and regression lines for Java experience and total rule test errors for the pre-tutor and quiz assessment occasions. The size of the marker is increased proportionally where more than value was present at a particular point.

10. Scatterplots and regression line for software self-efficacy and items test errors observed at the completion of the programmed instruction tutor and on the subsequent quiz. The size of the marker is increased proportionally where more than value was present at a particular point.