J. EDUCATIONAL COMPUTING RESEARCH, Vol. 7(1) 1-24, 1991

WHO IS LIKELY TO ACQUIRE PROGRAMMING SKILLS?*

VALERIE J. SHUTE
Air Force Human Resources Laboratory

ABSTRACT

The purpose of this study was to investigate the relationship between programming skill acquisition and various measures of individual differences, including: 1) prior knowledge and general cognitive skills (e.g., word knowledge, information processing speed); 2) problem solving abilities (e.g., ability to decompose a problem into its constituent parts); and 3) learning style measures (e.g., asking for hints versus solving problems on one's own). Subjects (N = 260) received extensive Pascal programming instruction from an intelligent tutoring system. Following instruction, an online battery of criterion tests was administered measuring programming knowledge and skills acquired from the tutor. Results showed that a large amount (68%) of the outcome variance could be predicted by a working-memory factor, specific word problem solving abilities (i.e., problem identification and sequencing of elements) and some learning style measures (i.e., asking for hints and running programs). Implications of the findings for the development of a theoretical framework on which to base programming instruction are discussed.

What are the characteristics of individuals who acquire programming skills efficiently and effectively? Can we successfully predict who is more likely to pick up programming skills from a computer programming curriculum? Can we improve the design of effective computer programming curricula? The research discussed in this article investigates the relationship between programming skill acquisition and various measures of individual differences, including: 1) prior knowledge and cognitive skills (e.g., word knowledge, information processing speed); 2) specific word problem solving abilities (e.g., ability to decompose a problem into its

*The research reported in this article was conducted at the Air Force Human Resources Laboratory, Brooks Air Force Base, Texas. The opinions expressed in this article are those of the author and do not necessarily reflect those of the Air Force.

doi: 10.2190/VQJD-T1YD-5WVB-RYPJ
http://baywood.com

constituent parts); and 3) learning style behaviors (e.g., asking for hints versus solving problems on one's own). Determining the correlated knowledge, skills, abilities, and traits for computer programming may ultimately provide educators with an explicit framework on which to base instruction.

Paralleling the prevalence of computers in our society (and the concurrent importance of computer programmers), research exploring underlying abilities of computer programmers has recently flourished [1-7]. While these studies differ on many dimensions (e.g., cognitive ability being measured, programming language being instructed/learned), three general problem-solving abilities have emerged as potentially important to programming skill acquisition: understanding, method-finding, and coding. *Understanding* is the identification of basic elements in a problem. This involves determining properties and relations of problem elements, establishing the initial and final problem states, and hypothesizing the operations needed for achieving the problem solution. *Method-finding* involves decomposing and sequencing problem elements into an outline of the problem solution. That is, what are the relevant operators or commands and how should they be arranged in the solution of a programming problem? *Coding* is the process of translating the natural language solution (from the previous stages) into programming code. Although this model has not been extensively tested, various researchers have found support for the abilities of understanding [2, 7], method-finding [2, 4, 5, 8, 9], and coding [2, 4] underlying programming skill acquisition.

The purpose of this study was to verify the existence and explicate the nature of the relationship between specific problem-solving abilities and programming skill acquisition from an intelligent tutoring system instructing Pascal programming. In other words, could success in learning programming skills be predicted from measures of specific problem-solving abilities (i.e., understanding, method-finding, and coding) beyond individual differences accounted for by more basic cognitive abilities?

The approach used to study these relationships involved estimating prior knowledge and skills from two sources. The first data source consisted of scores on the Armed Services Vocational Aptitude Battery [10] covering a broad range of knowledge and skills such as vocabulary knowledge and mathematical skills. The second data source consisted of cognitive process measures obtained from scores on a battery of computerized tests developed in the Learning Abilities Measurement Program (LAMP) at the Air Force Resources Laboratory [11]. The cognitive tests used in this study gauged working-memory capacity and information processing speed in each of three domains: quantitative, verbal and spatial.

Finally, an algebra word problem test battery was created [12] to estimate the specific problem-solving abilities, discussed above.[1] The statistical approach involved partialling out cognitive abilities from a learning outcome measure and

[1] The second ability of method-finding was divided into two subcomponents: 1) decomposition of problem parts; and 2) sequencing parts into an outlined solution.

testing if there was any remaining influence on outcome attributable to problem-solving abilities. In other words, is there anything unique to the problem-solving abilities (estimated from performance on an algebra word problems test) that can predict who will succeed in learning to program a computer?

Another goal of this research was to ascertain additional variables that may relate to efficient and effective programming skill acquisition. Included in this category are various learning style behaviors such as asking for hints from the system and actually running programs which was possible during phases 2 and 3 of the tutor.

METHOD

Subjects

The subjects in this study consisted of 260 males and females participating in a seven-day study on the acquisition of Pascal programming skills from an intelligent tutoring system. The gender distribution in the sample was approximately 80 percent males and 20 percent females. All subjects were high school graduates (or equivalent) with a mean age of 22.4. Subjects were recruited and selected from San Antonio colleges and technical schools. None of the subjects had any prior Pascal programming experience. All subjects were paid for their participation consisting of forty hours of testing and learning.

Materials

The Pascal programming intelligent tutoring system (Pascal ITS) used in this study was originally developed at the Learning Research and Development Center, University of Pittsburgh [13] and extensively modified at the Air Force Human Resources Laboratory. This system runs on a Xerox 1186 computer and was designed to help non-programmers learn how to program in Pascal.

The curriculum consisted of twenty-five programming problems of increasing difficulty. Initial problems involved simple write and read functions (e.g., Problem 2—*Write a program that prints out your name and phone number*). Intermediate problems involved simple "for" or "while" loops (e.g., Problem 13—*Write a program that asks the user if he or she would like to see a line of stars. If so, print a line of stars. Continue the above until the user does not want to see stars anymore*). Later problems involved more complex "while" or "repeat until" loops (e.g., Problem 25—*Write a program that asks the user if he or she wants to square an integer. If so, read in an integer, square it, and print out the result. Keep a running total of all the squares that are computed. Continue the above until the user doesn't want to square an integer any longer. Then print out the running total of all the squares*).

For each of the twenty-five problems in the tutor's curriculum, there were three learning phases. Each phase was designed to teach different skills associated with

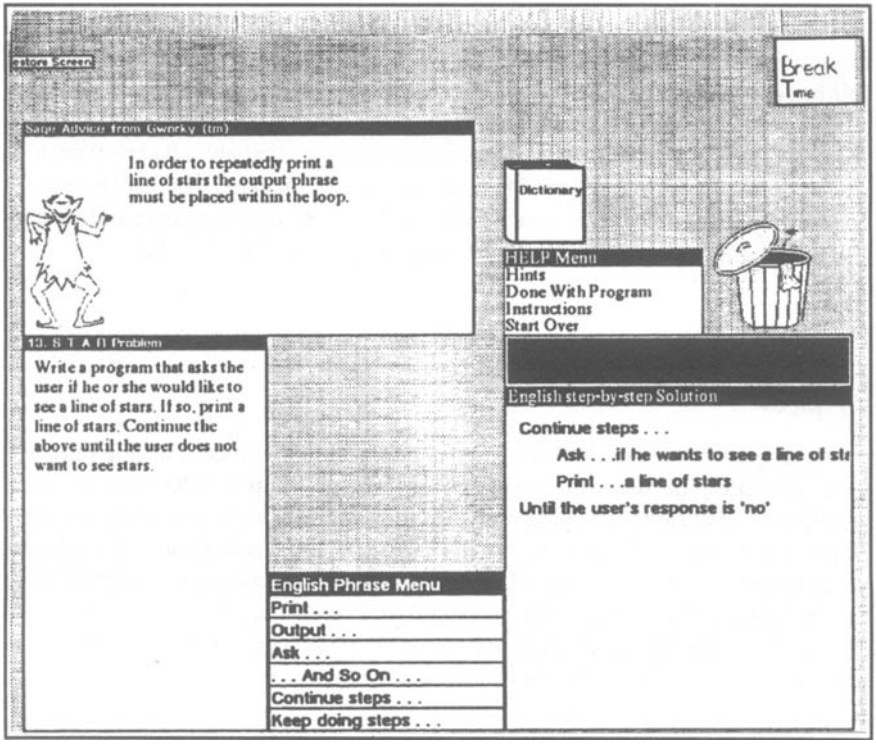

Figure 1. Screen of the Pascal tutor with a Phase 1 solution.

programming. In phase 1, subjects had to generate natural language solutions to programming problems. In phase 2, subjects expressed their phase 1 solutions in the form of "programming plans" [14]. These plans were then arranged into flowcharts (connecting like jigsaw puzzle pieces) for each problem. In phase 3, subjects translated their phase 2 visual solutions into Pascal code. Figures 1, 2, and 3 show the screen for the three phases in the solution of an intermediate problem type. This problem (#13) involved printing or not printing stars, depending on the user's response. For any given problem, subjects were required to proceed sequentially through the three phases. They could not go on to the next phase until they had succeeded in the current phase. Subjects could take as long as they needed to go through the entire curriculum. Up to thirty hours were allowed for interaction, and no subject required additional time.[2]

[2] The mean time spent on the tutor was 12.2 hours, with a 5.2 hour standard deviation. The minimum time to complete the entire ITS was 2.8 hours and the maximum time was 29.2 hours (N = 260). The data were normally distributed.

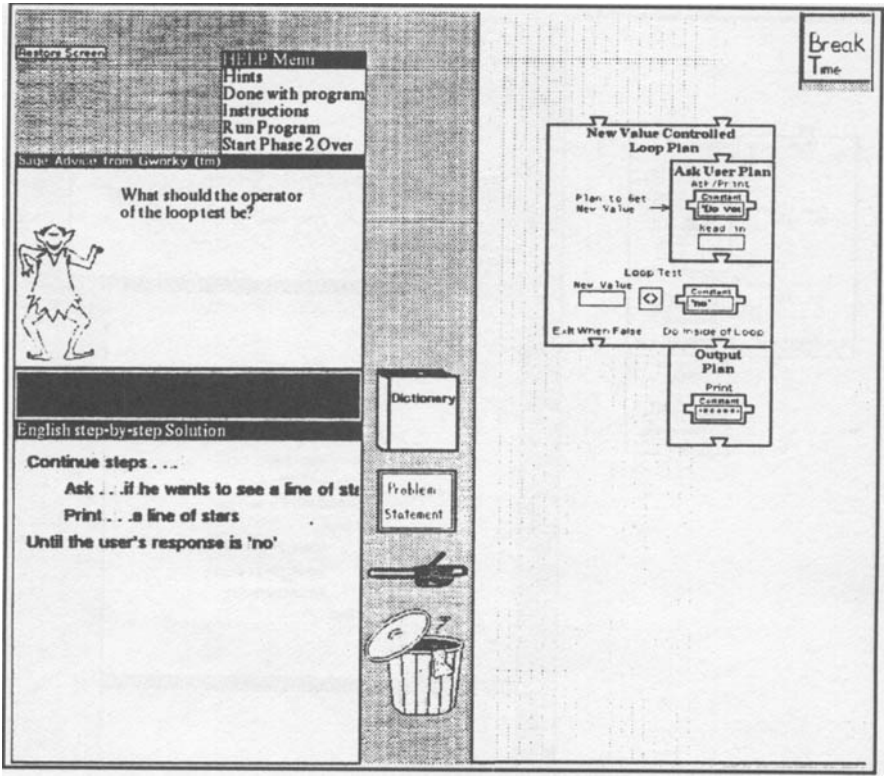

Figure 2. Screen of the Pascal tutor with a Phase 2 solution.

Subjects could ask for unlimited hints from the tutor and there were three levels to each hint, from more thought-provoking to more straightforward. Subjects were also free to run their solutions in phases 2 and 3. In phase 2, running a "program" would result in a dynamic flow of control as various plans (jigsaw puzzle pieces) lit up as they became active or relevant. The same flow of control could be witnessed in phase 3 where each line of code was highlighted as it became active.

A criterion posttest battery was created and administered online that tested the breadth and depth of knowledge and skills acquired from the Pascal ITS. This battery consisted of three tests, each one requiring progressively more complex programming skills. The first test involved the detection of errors in simple Pascal programs. Subjects were given a problem statement, presented with some Pascal code, and asked to determine if an error was present. If an error was thought to be present, they had to indicate what type of error it was (e.g., unnecessary line, misplaced line, missing line). Subjects then had to identify the part of the program (the line) which manifested the error. The second test involved decomposing and

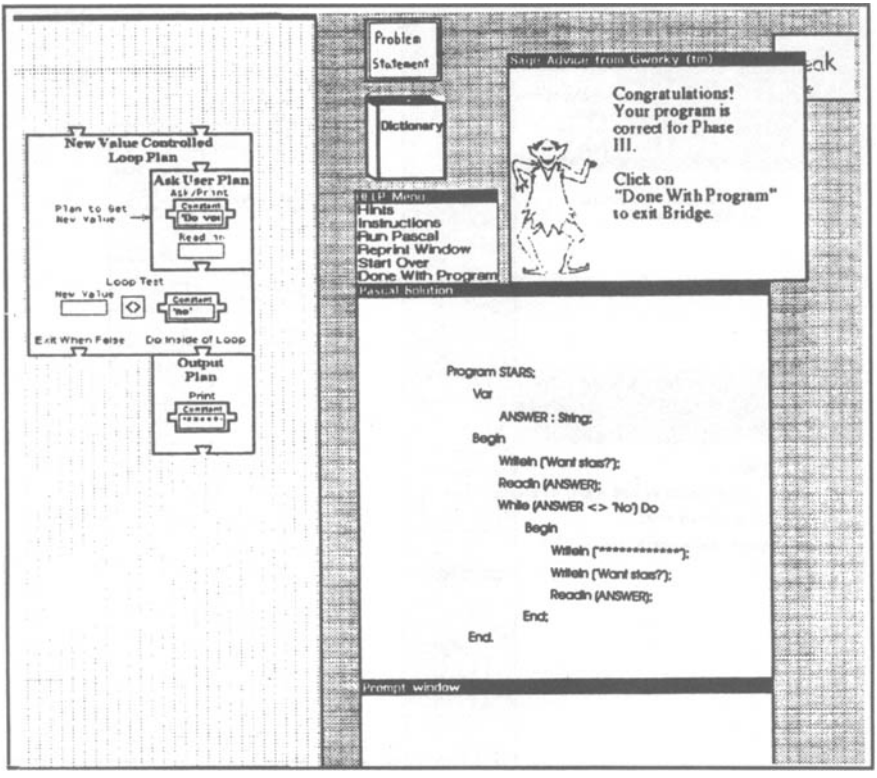

Figure 3. Screen of the Pascal tutor with a Phase 3 solution.

ordering Pascal commands into a problem solution. Here, subjects received a problem statement, then a menu of possible commands (e.g., Readln, Write, If...Then...). They selected and arranged the commands to solve the programming problem in an adjacent window. For the third test, subjects generated and wrote Pascal code from scratch in response to programming problem statements. There were twelve problems per test, comparable to the problem types encountered during the tutor. Subjects could take as long as they needed to complete the tests. For each test, accuracies per problem as well as response latencies were automatically tallied and recorded by the system.[3]

The algebra word problems test battery [12] consisted of four tests measuring the following specific problem-solving abilities: 1) Problem type identification; 2) Problem decomposition into relevant arithmetic operators; 3) Sequencing of

[3] Scoring of the third test (i.e., writing Pascal code) was done offline by two scorers, and interscorer reliability was high ($r > .90$).

the operators; and 4) Translation of verbal problem statements into algebraic equations. The first test (problem identification or understanding), consisted of twelve possible problem types: triangle, distance/rate/time, average, scale conversion, ratio, interest, area, mixture, probability, number, work, and progressions. Subjects initially read definitions of each problem type,[4] then for ensuing problems, had to select the most appropriate classification. For the second and third tests, algebra word problems were presented and the subject was required to specify the arithmetic operators involved, and the proper ordering of those operators, respectively. The last test required translation of word problems into mathematically correct equations. An example item from the translation test is: "Lester Lanning, a band leader, pays his drummer $500 for each 4-hour job. One month, the drummer worked 68 hours. How much did the drummer make?" (a) (68 + 4) × 500; (b) (68 × 500); (c) (500/4) × 68;[5] (d) (4/68) + 500.

As stated earlier, this test battery was created to map onto the proposed abilities believed to be related to learning to program: understanding parallels problem identification in that the goal of the understanding phase is to identify what type of problem needs to be solved. Method-finding maps onto decomposing and sequencing steps in a problem because it results in a program outline where each problem step is explicitly identified and placed in a particular order. Finally, coding involves the translation of natural language solutions into programming codes, comparable to the translation of arithmetic operators into an algebraic equation. This paper and pencil test took approximately thirty minutes for subjects to complete and consisted of ten problems per section.

The Armed Services Vocational Aptitude Battery (ASVAB) was included in this study to see how incoming knowledge and skills related to the acquisition of a new skill—Pascal programming. The ASVAB consists of ten separate tests: General Science, Arithmetic Reasoning, Word Knowledge, Paragraph Comprehension, Numerical Operations, Coding Speed, Auto Shop, Mechanical Comprehension, Math Knowledge, and Electronics Information. This paper and pencil battery took 3.5 hours to complete and is used by the various armed services for selection and classification of enlisted personnel. All subjects in this study were administered the ASVAB prior to the Pascal ITS. An example test item is presented from the Word Knowledge test. This requires the examinee to choose an alternative word whose meaning is most like the meaning of a word underlined in a phrase: "It was a *small* table." (a) sturdy; (b) round; (c) little; (d) cheap.

[4] Definitions of problem types were kept simple. For example, "A *scale conversion* problem consists of converting a quantity from one unit of measurement to another, like from 1 yard to 36 inches."

[5] There are, of course, other representations of this correct response. But of the four alternatives, just one is right. For this item, a correct alternative representation is: (68/4) × 500.

The Learning Abilities Measurement Program (LAMP) at the Air Force Human Resources Laboratory conducts basic research on individual differences in cognitive abilities and skill acquisition. As part of this program, different computerized tests, administered on Zenith 248 microcomputers, have been developed during the past five years that measure the cognitive attributes of working-memory capacity (WM) and information processing speed (PS) in each of three different domains: quantitative, verbal, and spatial [15]. Working memory tests require an individual to maintain some information in temporary storage while simultaneously processing new information. The degree to which individuals can handle this dual tasking without becoming overloaded reflects their working-memory capacity. Processing speed tests require an individual to answer various items as quickly as possible without sacrificing accuracy. A test battery was created [11] that consisted of three tests for each of the six categories—WM and PS in the verbal, quantitative and spatial domains (18 tests in all). Also included in the battery were several measures of general knowledge (see Appendix 1 for descriptions of all tests in this battery).

Learning behaviors (or indicators) were extracted from each student's history list. This extensive list represents all actions taken during tutor involvement. Sixty-six learning behaviors were tallied by the computer for each of the twenty-five problems, for each phase. Indicators were summed at the end of the tutor for total count measures, and slopes were derived across the twenty-five problems showing changes in behaviors over time. There are two parts to interpreting slope measures: 1) the size of the slope, and 2) the direction of the sign. Larger slopes imply more change from beginning to end of the tutor while smaller (or flat) slopes imply less (or no) change. Negative slopes imply a reduction in some behavior while positive slopes imply an increase in a behavior over time.

The learning behaviors investigated in this article included: Total number of hints requested (Hints), the slope of hints requested (Slope-Hints), number of times a program was run (Runs), and the slope of runs (Slope-Runs). These measures were selected as being representative of volitional types of behaviors (i.e., those actions under the learner's control). Preliminary findings have suggested that the hint-asking indicator is a potent predictor of Pascal programming skill acquisition [16].

Procedure

Subjects were tested in groups of approximately fourteen persons, and there were twenty groups tested all together. Each group spent seven days (nearly six hours per day) in this study. Subjects began the study being tested on the basic cognitive process measures. Over successive days, they were administered the ASVAB, followed by the algebra word problems test battery, the ITS (up to 30 hours), and the criterion test battery.

RESULTS

The data were analyzed in two ways. First, to reduce the number of incoming knowledge and skill measures, I computed a factor analysis combining ASVAB and LAMP test data. Second, to test incremental predictive validities of the various sets of measures, hierarchical regression analyses were computed. The three sets of measures that were tested included: 1) incoming knowledge and skills factors derived from the factor analysis; 2) specific problem-solving ability measures from scores on the algebra word problems tests; and 3) particular learning behaviors, obtained from the student history lists.

Results from the factor analysis (principal axis factoring with varimax rotation) yielded five factors accounting for 57 percent of the variance. Following are the interpretations of each factor: Factor 1 (Working Memory); Factor 2 (General Knowledge); Factor 3 (Information Processing Speed); Factor 4 (Technical Knowledge); and Factor 5 (Perceptual Speed). Descriptive statistics for each one of the ASVAB and LAMP tests are presented in Table 1. Factor loadings for each test can be seen in Table 2.

Hierarchical regression analyses were computed testing incremental validities predicting overall learning outcome from the Pascal ITS. The learning criterion measure was the average score from the three criterion tests.[6]

The first set of measures to be entered into the equation predicting programming skill acquisition were the five factor scores measuring incoming knowledge and skills: Working memory, general knowledge, information processing speed, technical knowledge, and perceptual speed. These were entered first as they represent more fundamental cognitive abilities [17-20]. The theoretical framework supporting this position was derived from the organization of aptitudes outlined by Cattell [21].

All five factors were entered into the equation. After backwards elimination, only Factor 5 (perceptual speed) dropped out. The four remaining knowledge and skills factors (i.e., working memory, general knowledge, information processing speed, and technical knowledge) accounted for 50 percent of the learning outcome variance (Multiple R = .71). Keeping those four variables in the equation, the measures of problem-solving ability were added next. Backwards elimination of this set of variables resulted in decomposition being removed first[7] followed by translation. So, by step two, the equation contained the four cognitive factors plus two problem-solving abilities—understanding (or problem identification) and sequencing (Multiple R = .75). The two problem-solving abilities contributed an additional 7 percent to the prediction of learning outcome, a significant increase ($p < .001$) beyond the first set of factors. Finally, the third set of variables

[6] Separate regression analyses were computed for each of the three tests. Results showed no major differences in predictors for individual tests in comparison with the overall measure, so the overall outcome measure is reported.

[7] The removal of the decomposition skills variable was due to its very high (r = .90) correlation to sequencing, thus not contributing any new variance to the equation.

Table 1. Descriptive Statistics for ASVAB and LAMP Tests (N = 260)

Variable	*M*	*SD*	*Minimum*	*Maximum*
Arithmetic reasoning (ASVAB)	52.14	9.19	30.00	66.00
Math knowledge (ASVAB)	52.53	8.98	33.00	68.00
Word knowledge (ASVAB)	53.00	6.98	26.00	61.00
Paragraph comprehension (ASVAB)	52.46	7.53	29.00	62.00
Mechanical comprehension (ASVAB)	52.95	9.86	24.00	70.00
Electronic information (ASVAB)	52.93	9.16	30.00	70.00
Auto shop (ASVAB)	53.06	8.94	24.00	69.00
General science (ASVAB)	52.83	8.81	30.00	68.00
Numerical operations (ASVAB)	52.79	8.40	24.00	62.00
Coding speed (ASVAB)	52.05	7.96	27.00	72.00
Working Memory				
ABC recall (LAMP)	46.11	26.14	0.00	100.00
Mental math (LAMP)	47.60	25.47	0.00	100.00
Slots test (LAMP)	60.20	19.01	0.00	95.00
Word span (LAMP)	59.82	26.24	0.00	100.00
Reading span (LAMP)	62.21	24.02	0.00	100.00
ABCD test (LAMP)	37.09	24.95	0.00	93.33
Spatial visualization (LAMP)	29.39	20.65	0.00	100.00
Figure synthesis (LAMP)	76.12	10.87	41.67	94.44
Ichikawa (LAMP)	73.50	9.20	34.44	95.56
Information Processing Speed				
Number fact reduction (LAMP)	3.75	1.50	0.28	8.00
Odd-Even test (LAMP)	1.35	0.41	0.19	3.58
Larger-Smaller test (LAMP)	0.54	0.12	0.37	1.72
Meaning identity (LAMP)	1.28	0.40	0.77	3.31
Semantic relations (LAMP)	1.86	0.45	0.22	3.22
Category identification (LAMP)	1.13	0.30	0.27	2.96
String matching (LAMP)	1.38	0.35	0.25	2.55
Santa's figures (LAMP)	0.88	0.23	0.45	1.97
Palmer's figures (LAMP)	1.22	0.31	0.29	2.87
General Knowledge				
General knowledge survey (LAMP)	60.87	20.09	0.00	98.00
Meaning identity (LAMP)	93.78	5.55	60.42	100.00

Note: Means with single digits refer to latency scores (in seconds) while means with double digits refer to the percent correct.

and associated slope measures were added to the equation. These included hints, slope-hints, runs, and slope-runs. All four learning style variables remained in the equation following backwards elimination (Multiple R = .82). The amount of unique variance (11%) accounted for by these new variables beyond the other two sets was significant ($p < .001$).

Table 2. Five Factor Solution from Principal Axis Factor Extraction with Varimax Rotation

Variable	*Factor 1*	*Factor 2*	*Factor 3*	*Factor 4*	*Factor 5*
ABC recall	**.81**	.13	−.16	.18	.14
Arithmetic reasoning	**.66**	.32	−.14	.37	.22
Math knowledge	**.64**	.20	−.18	.38	.26
Mental math	**.63**	.14	−.09	.10	.18
Figure synthesis	**.63**	.10	−.02	.20	.12
Reading span	**.56**	.22	−.13	.00	.15
Slots test	**.56**	.22	−.15	.11	.42
Spatial visualization	**.55**	.23	−.13	.41	.04
Word span	**.55**	.41	−.17	.23	.25
Ichikawa	**.54**	.00	−.20	.12	.11
ABCD test	**.52**	.32	−.12	.25	.04
Word knowledge	.32	**.72**	−.11	.36	−.03
Meaning identity (latency)	−.13	**−.68**	.44	−.12	−.20
General knowledge survey	.39	**.65**	−.23	.33	−.03
Paragraph comprehension	.32	**.61**	−.01	.30	.21
Meaning identity (percent correct)	.31	**.59**	.08	.13	.11
Semantic relations (latency)	.05	**−.50**	.44	−.05	−.12
String matching	−.08	−.03	**.71**	−.00	−.08
Number fact reduction	−.32	−.14	**.61**	−.03	−.41
Odd-Even test	−.18	−.16	**.60**	−.13	−.08
Santa's figures	−.03	.12	**.58**	.06	.08
Palmer's figures	−.19	−.12	**.57**	−.10	−.06
Larger-Smaller test	−.21	−.18	**.53**	−.13	−.21
Category identification	.03	−.43	**.52**	.08	−.13
Auto shop	.08	.17	.03	**.76**	.09
Mechanical comprehension	.41	.15	−.16	**.74**	.02
Electronic information	.30	.22	−.08	**.73**	.10
General science	.38	.48	−.18	**.58**	−.10
Numerical operations	.36	.04	−.20	.02	**.63**
Coding speed	.32	.15	−.16	.07	**.60**

In summary, a large amount of criterion variance (68%) was accounted for by three sets of variables: incoming knowledge and skills, problem-solving abilities, and learning behaviors. Table 3 shows the final solution.

The WM factor was the best predictor of Pascal programming skill acquisition. With all of the other variables in the equation, this was the only one of the original cognitive factors that remained significant. The algebra word problem solving

Table 3. Hierarchical Regression Solution Predicting Overall Learning Outcome (Multiple R = .82)

Variable	*Beta*	*T*	*Significance*
Knowledge and Skills			
Working memory factor	0.26	5.00	0.001
General knowledge factor	0.06	1.31	0.191
Processing speed factor	–0.06	–1.15	0.149
Technical skills factor	0.04	0.81	0.418
Problem Solving Abilities			
Problem identification	0.12	2.55	0.011
Sequencing	0.18	3.33	0.001
Learning Behaviors			
Hints	–0.23	–2.70	0.007
Slope-hints	–0.18	–2.09	0.038
Runs	0.07	1.77	0.078
Slope-runs	0.09	2.21	0.028

abilities of identifying problems and sequencing solutions were also important predictors of programming skill acquisition. Both of these variables showed positive relationships to learning outcome. And of the learning behaviors, asking for hints from the system was associated with significantly lower outcome scores [16] while running programs was a productive behavior (but only marginally significant). The slope measures remained in the equation along with the total count measures. For slope-hints, the more negative slopes (i.e., progressively fewer hints requested over time) predicted learning outcome. But for slope-runs, the more positive slopes (i.e., progressively more programs run over time) predicted learning outcome.

From the results above, a logical conclusion would be that hint-asking, overall, is a suboptimal behavior. This is particularly disturbing since one of the main features of intelligent tutoring systems is their ability to provide assistance to learners. The last analysis examined hint-asking behavior more closely, to disambiguate hint-users from hint-abusers. That is, some individuals may have requested a lot of hints because they really needed help. Others may have asked for hints due to a particular learning style. Additionally, some individuals who actually needed help may not have asked for it, resulting in "floundering" behaviors,[8] while others may not have asked for hints because they were proceeding satisfactorily. Table 4 sums up these four hypothetical categories of subjects.

[8] "Floundering" in this context is measured by making a lot of errors during tutor interaction. Five error types were summed for this measure: errors involving sequencing, omission, logic, data typing, and incorrect operators. These errors were tallied within each phase for each problem.

Table 4. Four Hypothetical Subject Categories

Errors	*Hints*	*Expected Outcome*	*Category*
Few	Few	High	Productive
Few	Many	Low	Hint-abusers
Many	Few	Low	Counter-productive
Many	Many	High	Hint-users

Table 5. Regression Solution Predicting Overall Learning Outcome (Multiple R = .84)

Variable	*B*	*T*	*Significance*
Hints	2.87	1.19	0.237
Errors	–22.51	–9.58	0.001
Hints x Errors	5.84	7.87	0.001

This categorization of subject types was based on a rational analysis of needs. If a subject does not make errors, he or she has little *need* for hints, but if a subject makes a lot of errors, then there is a clear *need* for assistance. To test the main effects of hints and errors on learning outcome (and particularly the interaction between them), a regression analysis was computed. The solution is presented in Table 5. Results indicated that the main effect of errors was significant as well as the interaction between hints and errors (Multiple R = .84). The main effect of hints on learning outcome was not significant.

To illustrate this interaction, rather than present scatterplots of the data, I computed median splits for the main effects. This resulted in four groups of subjects. Figure 4 represents these data. The four groups included: 1) Few Errors, Few Hints (N = 115), 2) Few Errors, Many Hints (N = 13), 3) Many Errors, Few Hints (N = 11), and 4) Many Errors, Many Hints (N = 103).[9]

The disordinal interaction depicted in Figure 4 supports the conclusion that hints are differentially good/bad for different individuals. For individuals in the "few errors" group, learning outcome was significantly higher ($F_{1,126}$ = 18.29; p < .001) if they worked out solutions on their own (i.e., asked for fewer hints) than if they relied on the system for help (i.e., asked for many hints). But for individuals in the "many errors" group, just the opposite was true: more hints were associated with higher outcome scores ($F_{1,112}$ = 3.61; p = .06).

[9] There is an unequal number of subjects in each cell since 1) hints and errors are correlated and 2) some of the cells are more/less typical than others. For example, only eleven subjects comprised the "Many Errors, Few Hints" group. Most individuals making a lot of errors request hints. Note also that the significant interaction reported in the text was based on continuous data. When the regression was recomputed with the category data, the interaction remained significant.

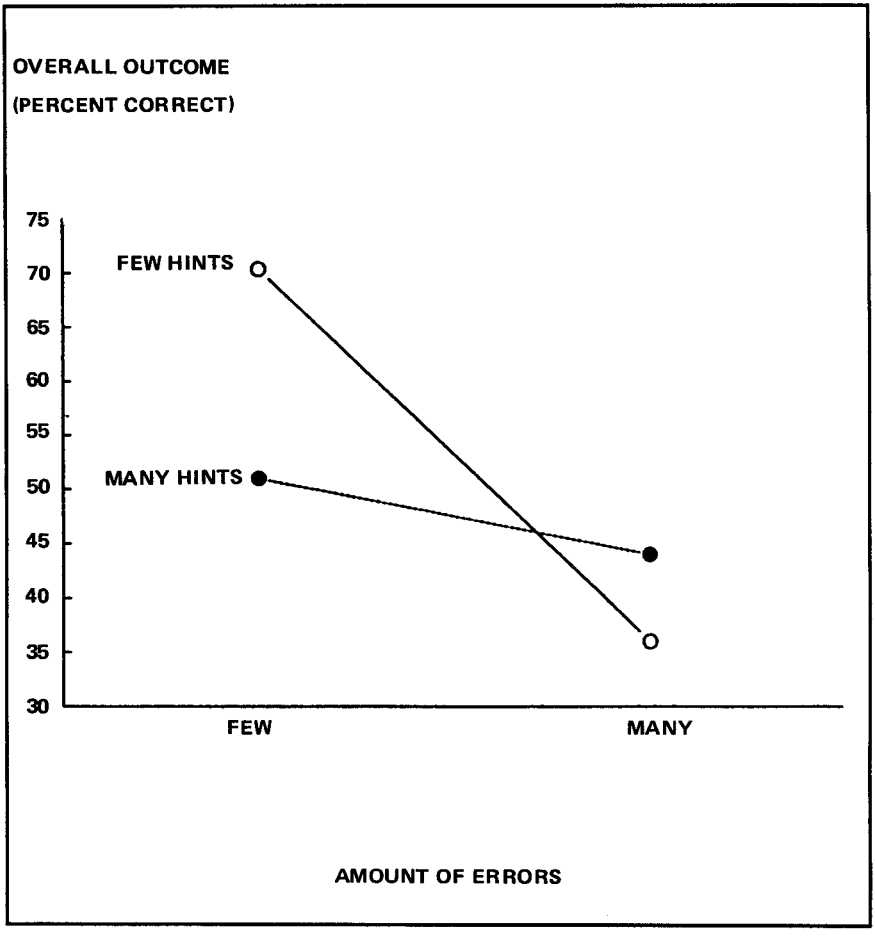

Figure 4. Interaction between Hints and Errors.

DISCUSSION

The question asked at the beginning of this article concerned the characteristics of individuals who successfully acquired programming skills. This issue was addressed in terms of the knowledge, skills, abilities and traits tested in this study. The literature on individual differences in skill acquisition is replete with studies showing that people acquire a new skill because of their incoming knowledge and cognitive skills [12, 19, 20, 22]. But there is much less in the individual differences literature concerning the role of problem-solving abilities and stylistic measures on new learning. The main research question in this article was: Once

the weighty influences of incoming knowledge and skills have been controlled, do problem-solving abilities or stylistic variables contribute anything new in predicting learning outcome? The appeal of investigating specific abilities is that they can be instructed and learned while more basic cognitive skills are less subject to change [23, 24].

First, it was shown that individuals with higher scores on the working memory factor evidenced higher scores on the criterion test battery. Working memory has been shown to predict both declarative learning and procedural learning [25]. The Pascal ITS used in this study involved both kinds of learning. Also, working memory has been shown to correlate very highly with general reasoning abilities [26] which has been shown to correlate highly with general intelligence [20]. So these results were not at all surprising and replicated others' findings on working memory and cognitive skill acquisition. The more fundamental question about the exact facets of working memory that cause improved learning goes beyond the scope of this article. But new learning does appear to be mediated by working-memory capacity.

Next, problem-solving abilities were estimated from scores on an algebra word problems test battery. Those measures were then used as predictors of programming skill acquisition *after* incoming knowledge and skills factors were entered into the equation. Results indicated that there was additional, unique variance accounted for by two of the problem-solving abilities. The two major points of interest here are that certain skills measured in one domain (algebra) predicted skill acquisition in another domain (computer programming). This kind of transfer of skills from one area to another is not often reported in the literature. Second, two problem-solving abilities (problem identification and sequencing) significantly predicted programming-skill acquisition after the effects of the cognitive factors were controlled. Translation and decomposition abilities did not predict learning outcome. Because sequencing and decomposition skills were highly correlated ($r = .90$), decomposition dropped out of the final equation as it was virtually indistinguishable from sequencing. The most probable explanation of why translation was not included in the final equation is that in this sample of non-programmers, the most difficult hurdle for most of them had to do with formulating and organizing an outline solution for a given problem. Once that structure was established (i.e., sequenced appropriately), translating it into Pascal code was relatively easy, especially in conjunction with the tutor's support. The pedagogical approach taken by the ITS emphasized the higher, conceptual level of programming (i.e., organizing a solution using programming "plans" or constructs) more than the lower level, syntactical aspects of Pascal coding. Thus, these findings based on the relative importance of the abilities of understanding and sequencing are not surprising in light of the design and implementation of the ITS.

All four of the learning style measures predicted the learning criterion. What this tells us is that there are certain behaviors which seem to be more efficacious

during learning a new skill than others. A particular learning style indicator, hint-asking, showed a significant and strong negative correlation with learning outcome ($r = -.64$). The obvious (but misleading) conclusion is that problem solving on one's own without excessive assistance from a tutor leads to better learning outcome. It was later shown that this statement is only true for a certain class of individuals (i.e., those making few errors). For others (i.e., those making more errors), it was better to seek tutorial assistance. Furthermore, hint-slope data showed that if one *had* to ask for hints, it was better to get assistance early in the tutor and try to avoid relying on help during the later stages of learning.

The four hypothetical subject categories and expected outcomes were thus supported by the data. Some people asked for hints due to their learning style ("hint-abusers"). Others asked for hints because they were having problems ("hint-users"). For the persons not using the hint option, those making few errors did not really need help, but there were some individuals who, for whatever reasons, chose not to get help from the system. This group did worse on the outcome measure than any other category of subjects.

The overall picture that emerges from these findings is that success in learning programming skills from the Pascal ITS used in this study can be predicted by working-memory capacity, good problem-solving abilities, and being an active learner (i.e., running programs, working out solutions on one's own, or seeking assistance when appropriate). It is important to keep in mind, though, that these results are mediated by the interaction involving learning environment, learner behaviors, and learning outcome measures employed in this study. The ITS environment differed between persons as a result of learners' activities. That is, if a person proceeded through the tutor by repeatedly asking for hints (whether help was really needed or not), then the environment was more structured or didactic. But for individuals relying on their own problem-solving abilities, the environment was less structured, eliciting more trial-and-error types of learning behaviors.[10] As illustrated in Figure 4 by the disordinal interaction, some environments are better suited for some individuals than others. The more didactic environment resulting from many hints being requested is better for those committing many errors, while the more trial-and-error environment resulting from minimal hints is better for subjects making fewer errors during learning.

What are the implications of these findings for programming teachers and ITS developers? Since certain problem-solving abilities, as outlined and tested in this study, are highly correlated with successful acquisition of programming skills, and since these same abilities are trainable [23], computer programming curricula may benefit from the inclusion of supplemental instruction on relevant problem-solving skills (e.g., part-task training of sequencing skills).

[10]For more on this topic, see [27] for a complete discussion of a taxonomy of learning skills based on the interaction of four dimensions: learning environment, learning outcome, subject matter, and learning styles.

Information about an individual's cognitive process measures may also be used to vary instruction in a principled manner, such as teaching smaller chunks of relevant knowledge for those with lower WM capacity. In other words, since WM capacity was shown to be an important predictor of programming skill acquisition, and the functional size of an individual's working memory cannot be directly manipulated, instruction may be varied based on differing units of knowledge for those individuals judged (by simple testing) to have smaller WM capacities.

Furthermore, if a person is making more than the usual number of errors[11] during learning, he or she could be encouraged to get help from the system, or the system could give help, unsolicited. If a person is judged to be making less than the average number of errors, the system could either impose a limitation on the number of hints a person may receive, or reinforce an individual's independence in the learning process, especially in the latter stages of learning.

In conclusion, a large amount of the criterion variance (68%) can be explained by just a few variables. This information can be used to enhance instruction, focusing on those variables impacting programming skills acquisition, or we can use the findings to predict who will acquire good programming skills for selection and classification purposes. If the designated variables can be instructed and/or trained, we can maximize instruction for more individuals, which is the purpose of ITS's, in particular, and education, in general. For purposes of selection and classification, our findings highlight the variables that allow us to predict *who is likely to acquire programming skills.*

APPENDIX 1: BATTERY OF *CAM* TESTS—VERSION 1.0

I. WORKING MEMORY (Percent correct)

A. Quantitative

ABC Recall: Subjects must learn and remember numeric values assigned to the letters A, B, and C. Statements (e.g., A = B/2) are presented one at a time, and subjects are permitted to look at each one for as long as desired before going on to the next statement. They are then asked to recall the values of the letters one at a time. Some of the problems are more difficult than others since values must be computed (e.g., A = 2 × 8 or, A = 16). Still other values cannot be computed until the value of another letter is known (e.g., B = A + 4). Even-odd reliability = .95.

Mental Math: This task requires subjects to calculate a subtraction or division problem mentally, and then choose the correct answer from 5 alternatives. A problem appears on the screen for 2 seconds (preceded by a warning asterisk) and

[11]This error count can be made by the ITS, maintained and monitored in a history list. If the number of errors exceeds some threshold value, that could invoke some tutoring, even if not explicitly requested.

then disappears. Subjects mentally solve the problem for as long as they wish. When they have the answer, they hit the space bar to see the five alternatives. They have 4 seconds to type in the number of the correct answer. Even-odd reliability = .88.

Slots Test: This test presents simple math equations (e.g., 5 + 2) in five sequential positions on the screen. Subjects must calculate the equations as they are presented and remember the answer for each position. Following the presentation of all equations, a question mark appears in one of the positions, and subjects must type in the corresponding answer. The five positions are marked by horizontal lines, one next to the other. Problems are presented from left to right, one at a time. Two rates of presentation exist (i.e., slow and fast) and before each trial, subjects are warned to get ready for either a slow or fast item. Each problem presents between 1 and 10 math equations. In the more difficult items, new math problems may be presented in a slot where a problem was already presented. Subjects are required to remember the most recent answer. Even-odd reliability = .91.

B. Verbal

ABCD Test: Subjects are presented with five general rules:

Rule 1—Set 1 = A and B.
Rule 2—Set 2 = C and D.
Rule 3—Set 1 can either precede or follow Set 2.
Rule 4—A can either precede or follow B.
Rule 5—C can either precede or follow D.

Each problem consists of three instructions presented one at a time concerning Sets 1 and 2. For example: 1) A precedes B. 2) Set 1 follows Set 2. 3) C precedes D. Each instruction is presented one at a time, and subjects may look at each one as long as desired before going on to the next one. They must determine the appropriate order of the letters and then hit the space bar to see the choices of answers. They then choose the number for the correct answer (e.g., CDAB). Even-odd reliability = .81.

Word Span Test: Subjects are required to memorize a short list of words and answer questions about them. A "Get Ready!!" warning precedes the words, which are presented one at a time. The questions are asked in an equation-like format. For example, if the list were 'neat, burp, inn', a possible question is 'neat + 1 = ?'. This question asks for the word which is one position after 'neat'. Answers are presented in a multiple choice format; alternatives are synonyms to the actual words on the list. Subjects must type in the number for the synonym which matches the word from the list. Any given word list is between 3 to 5 words long. Subjects answer three questions about each list, after which they are

told how many questions they had correct for that word list. Even-odd reliability = .93.

Reading Span: This task tests subjects' ability to classify true/false statements and their short-term memory capacity. Subjects are presented a list of sentences of general knowledge which they must determine to be true/like ('L') or false/different ('D'). Concurrently, they must memorize the last word in each sentence (this word is highlighted a different color from the other words). Sentences are presented one at a time, after which they are asked to type in the first two letters of each word in the order that they appeared. Subjects receive partial credit if the correct letters are typed in, but in the wrong sequence. Even-odd reliability = .93.

C. Spatial

Figure Synthesis: Two geometric figures are presented for subjects who are instructed to imagine the shape if the pieces were rearranged to form one figure. These figures are then replaced by a third figure. The subject must determine whether or not the third figure could be formed from the combined figures. Reaction time is presented when subjects give the correct response. Even-odd reliability = .65.

Spatial Visualization: This task requires 3-dimensional visualization. Subjects read descriptions of blocks and visualize how they appear before and after various manipulations (e.g., colors, initial size, ensuing size, number of blocks it may be cut into, etc.). The subject is allowed to study the description for 30 seconds before the first question is asked (although the description remains on the screen throughout the problem). Subjects work the problems mentally and then choose one of the multiple choice answers using the letters A through O. Subjects are given 60 seconds to respond, at which time they are told to enter their response (within another 10 seconds). If no response is entered during that time, the item is counted wrong. For each description, three or more questions may be asked in this multiple choice format. Even-odd reliability = .84.

Ichikawa: This test presents a 5 x 5 matrix of squares containing 7 asterisks. The placement of the asterisks is random. Subjects see a warning asterisk, the matrix filled with asterisks, and then a blank matrix with a question mark in one of the squares. Subjects are to determine whether or not an asterisk was in that square, and respond with 'L' (correct) or 'D' (not correct). Subjects have 3 seconds to respond, and then a new blank matrix appears with another question mark in it. For each matrix, three positions are questioned. The computer provides accuracy feedback. Subjects are allowed to study the initial matrix for 2 seconds, followed by a 1 second delay before questions are asked. Even-odd reliability = .73.

II. INFORMATION PROCESSING SPEED (Latencies)

A. Quantitative

Number Fact Reduction: Subjects are given four sets of simple arithmetic problems, each set containing only one type of operation (i.e., addition, subtraction, multiplication, or division). Each problem is preceded by a "Get Ready!!" warning and asterisk. Subjects must quickly determine whether or not the problem is correct (type in 'L') or incorrect (type in 'D'). The computer gives accuracy feedback. Even-odd reliability = .98.

Larger-Smaller Test: Subjects are presented with two single-digit numbers on separate sides of the screen to determine which of the two is larger. If the one on the right is greater, 'L' is the correct response; and if the one on the left is greater, 'D' is the correct response. Each set of numbers is preceded by a warning asterisk and a one second delay. The test contains four sets of 36 trials. Even-odd reliability = .98.

Odd-Even Test: In this test subjects must decide as quickly as possible whether two numbers presented are odd or even. The two numbers (between 1 and 20) are presented one above the other. Some numbers are presented as digits and others as English words (e.g., 5 or five). Subjects respond with 'L' if both numbers are either odd or even. If one number is odd and the other is even, then 'D' is the correct response. Reaction time is shown when a response is entered. Even-odd reliability = .97.

B. Verbal

Meaning Identity: Two words are presented and the subject must decide whether they have the same or different meanings. Subjects type in 'L' if they have the same meaning, and 'D' if they have different meanings. Some of the pairs of words are repeated exactly and some are repeated with different pairings. Each pair is preceded by a warning asterisk. The goal is to respond as quickly as possible and still try to get 95% of the items correct. After each set, the student's percent correct and average response time are shown. Even-odd reliability = .98.

Category Identification: Subjects are presented with three words: one on the left, one on the right, and one centered above the other two. The subjects must determine which of the lower words belongs in the same class or category as the word at the top. If it is the one on the left, 'D' is the correct response, and 'L' if it is the one on the right. Three warning asterisks are presented where the words will appear for the subjects to focus their attention. The computer responds with whether their answer is correct or incorrect in addition to reporting their reaction time. Even-odd reliability = .98.

Semantic Relations Verification Test: Subjects must determine whether or not simple sentences are true ('L') or false ('D'). Key words in the sentence are colored. For example, in the sentence 'Theft is a crime', '*theft*' and '*crime*' might be colored differently from the default color of white like the rest of the sentence. The computer responds with whether or not the subject made the correct response, and the reaction time if the response is correct. Even-odd reliability = .98.

C. Spatial

Santa's Figures: Subjects are presented with 2 sets of geometric figures to determine if they have the same parts ('L') or different parts ('D'). Each set consists of three geometric shapes or figures (e.g., circle, arrow, diamond, square) next to each other. The first set appears for 2 seconds, disappears, and then the second set appears. Subjects then make their decisions. The order of the figures is not important, only whether or not the two sets contain the same figures. Even-odd reliability = .93.

Palmer's Figure Comparison: Subjects are presented with two 3 × 3 dot-matrices with 5 interconnecting lines forming different geometric patterns. These matrices are presented side by side until a response is entered. The subject must respond as quickly as possible whether the two line figures are the same ('L') or different ('D'). The computer responds with either 'correct' or 'wrong'. At the end of each set the computer provides summary performance feedback. Even-odd reliability = .96.

String Matching: This task presents two strings of letters, symbols, or digits for subjects to determine if they are the same ('L') or different ('D'). The strings may be upper or lower case letters, numbers, or any other symbol from the keyboard (e.g., * & % $). Each string is between 2 to 5 characters long. The two stimuli comprising a comparison are always the same length and composed of the same character type (e.g., upper case with upper case, symbols with symbols, etc.). Strings may be made up of the exact characters, but if the sequence is different, then it is not a match. In all cases, the two strings will either be identical, transposed, or just one value may be different. Subjects are presented a warning asterisk and then the two strings. The strings stay on the screen until a response is entered. At that time, the computer responds with whether the response was correct and the reaction time. Even-odd reliability = .97.

III. GENERAL KNOWLEDGE (Percent correct)

A. Verbal

General Knowledge Survey: Subjects are asked general questions (e.g., *San Antonio is in what state?*), and must respond by typing in the first two letters of the

answer (e.g., 'TE' for Texas). The computer responds with 'correct' or 'wrong'. Even-odd reliability = .93.

Meaning Identity: Two words are presented and the subject must decide whether they have the same or different meanings. Subjects are to type in 'L' if they have the same meaning, and 'D' if they have different meanings. Even-odd reliability = .80.

Semantic Relations Verification Test: Subjects must determine whether or not simple sentences are true ('L') or false ('D') (e.g., 'Theft is a crime' would be a true sentence). The computer responds with whether or not the subject made the correct response, and the reaction time if the response is correct. Even-odd reliability = .84.

ACKNOWLEDGMENTS

I wish to sincerely thank Lisa Gawlick, Roy Chollman, Carmen Pena, Elena Selles and Rich Walker for help in data collection. I am very much indebted to Dan Woltz, Pat Kyllonen, Wes Regian, Ray Christal and Bill Tirre for their help and insightful suggestions during the conduct of this research. Finally, I would like to acknowledge the creative and industrious programmers who worked on this project: Tony Beauregard and Kym Costa.

REFERENCES

1. B. Adelson and E. Soloway, Methodology Revisited: The Cognitive Underpinnings of Programmers' Behavior, in *Human-Computer Interaction*, G. Salvendy (ed.), Elsevier, Amsterdam, pp. 181-184, 1984.
2. R. Brooks, Toward a Theory of the Cognitive Process in Computer Programming, *International Journal of Man-Machine Studies, 9,* pp. 737-751, 1977.
3. J. L. Dyck and A. P. Trent, *Learning to Program in LOGO as a Function of Instructional Method and Cognitive Ability*, Paper presented at the AERA, Boston, Massachusetts, April 1990.
4. R. E. Mayer, J. L. Dyck, and W. Vilberg, Learning to Program and Learning to Think: What's the Connection?, *Communications of the ACM, 29*:7, pp. 605-610, 1986.
5. M. C. Pena and W. C. Tirre, *Cognitive Correlates of the Early Stages of Programming Skill Acquisition*, Paper presented at the AERA, Boston, Massachusetts, April 1990.
6. R. T. Redmond and J. B. Gasen, PAST: Viewing the Programming Process, *Behavior Research Methods, Instruments, and Computers, 20*:5, pp. 503-507, 1988.
7. J. M. Faries and B. J. Reiser, Access and Use of Previous Solutions in a Problem Solving Situation, *Technical Report 29*, Cognitive Science Laboratory, Princeton University, Princeton, New Jersey, June 1988.

8. R. E. Snow, Aptitude Processes, in *Aptitude, Learning and Instruction, Vol. 1*, R. E. Snow, P. A. Federico, and W. E. Montague (eds.), Lawrence Erlbaum Associates, Hillsdale, New Jersey, pp. 27-63, 1980.
9. K. Swan and J. B. Black, The Cross-Contextual Transfer of Problem Solving Skills, *Technical Report*, Teachers College, Columbia University, New York, 1987.
10. Department of Defense, Test Manual for the Armed Services Vocational Aptitude Battery (DoD, 1340.12AA), *U.S. Military Entrance Processing Command*, North Chicago, Illinois, 1984.
11. P. C. Kyllonen and R. E. Christal, *Cognitive Abilities Measurement Battery, Version 1*, unpublished manuscript and computer program, 1990.
12. M. C. Pena, Cognitive Determinants of the Early Stages of Programming Skill Acquisition, unpublished Master's thesis, St. Mary's University, San Antonio, Texas, 1989.
13. J. G. Bonar, R. Cunningham, P. Beatty, and W. Weil, Bridge: Intelligent Tutoring with Intermediate Representations, Technical Report, LRDC, University of Pittsburgh, Pittsburgh, 1988.
14. E. Soloway, J. G. Bonar, and K. Ehrlich, Cognitive Strategies and Looping Constructions: An Empirical Study, *Communications of the Association for Computing Machinery, 26*, pp. 853-861, November 1983.
15. P. C. Kyllonen and R. E. Christal, Cognitive Modeling and Learning Abilities: A Status Report of LAMP, in *Testing: Theoretical and Applied Issues*, R. Dillon and J. W. Pellegrino (eds.), Freeman, San Francisco, 1989.
16. V. J. Shute, D. J. Woltz, and J. W. Regian, An Investigation of Learner Differences in an ITS Environment: There's No Such Thing as a Free Lunch, in *Artificial Intelligence and Education*, D. Bierman, J. Breuker, and J. Sandberg (eds.), IOS, Amsterdam, pp. 260-266, 1989.
17. J. R. Anderson, Skill Acquisition: Compilation of Weak-Method Problem Solutions, *Psychological Review, 94*, pp. 192-210, 1987.
18. A. Baddeley, *Working Memory*, Clarendon Press, Oxford, 1986.
19. D. J. Woltz, An Investigation of the Role of Working Memory in Procedural Skill Acquisition, *Journal of Experimental Psychology: General, 117*, pp. 319-331, 1988.
20. P. L. Ackerman, Determinants of Individual Differences during Skill Acquisition: Cognitive Abilities and Information Processing Perspectives, *Journal of Experimental Psychology: General, 117*, pp. 288-318, 1988.
21. R. B. Cattell, *Abilities: Their Structure, Growth, and Action*, Houghton-Mifflin, Boston, 1971.
22. P. C. Kyllonen and D. J. Woltz, Role of Cognitive Factors in the Acquisition of Cognitive Skill, in *Abilities, Motivation, and Methodology*, R. Kanfer, P. L. Ackerman, and R. Cudeck (eds.), Lawrence Erlbaum Associates, Hillsdale, New Jersey, pp. 239-280, 1989.
23. J. D. Bransford and B. S. Stein, *The IDEAL Problem Solver*, W. H. Freeman, New York, 1984.
24. V. J. Shute and R. Glaser, A Large-scale Evaluation of an Intelligent Discovery World: Smithtown, *Interactive Learning Environments, 1*, pp. 51-76, 1990.
25. P. C. Kyllonen and D. Stephens, Cognitive Abilities as Determinants of Success in Acquiring Logic Skill, *Learning and Individual Differences*, in press.

26. P. C. Kyllonen and R. E. Christal, Reasoning Ability Is (Little More Than) Working-Memory Capacity, *Intelligence*, in press.
27. P. C. Kyllonen and V. J. Shute, A Taxonomy of Learning Skills, in *Learning and Individual Differences*, P. L. Ackerman, R. J. Sternberg, and R. Glaser (eds.), W. H. Freeman, New York, pp. 117-163, 1989.

Direct reprint requests to:

Dr. Valerie J. Shute
Air Force Human Resources Laboratory
Cognitive Skills Assessment Branch
Brooks Air Force Base, TX 78235