INTERNATIONAL JOURNAL OF BUSINESS, 2(2), 1997 ISSN:1083-4346

A Study of If-Then-Else Structures in End User Programming

Houn-Gee Chen

Information technology assumed a role of growing importance in organizations during the 1980s. It is no longer used exclusively by small groups of computer specialists; but, rather, office automation and end user computing (EUC) are used by workers at all levels and in all areas of an organization. This study attempts to gain an understanding of factors associated with the use of EUC. The present study is an empirical study, using one end user programming construct (IF-THEN-ELSE). The major findings are: 1) the constraint of human short-term memory has a significant impact on the use of a programming construct, 2) the individual aptitude and personality traits are strongly related to the use of IF-THEN-ELSE constructs, and 3) the format of the construct could also affect the user's performance.

I. INTRODUCTION

Steven Anderson, the head loan officer of a mid-sized bank, has 15 senior and junior loan officers reporting to him. Steven has a strong computer background and plans to develop a step-by-step computer program, which could be used by all lending officers to evaluate loan applications. Although anxious to develop such a program, Steven is not certain about how to structure the program, given the various levels of computer knowledge in his department. The case of Steven Anderson is just one example of end user computing (EUC), one of the today's fastest-growing computer applications [8,29,39,40] in information technology.

Rockart and Flannery [40] were among the first to address fundamental issues concerning EUC. They classified end users into six distinct classes, according to the degree of programming effort. Up to 75% of end users were ranked in the End-User Programming Mode--those users who develop, implement, maintain, and utilize their own computer applications, using high-level procedural languages, fourth generation languages (4GL), or a language that is part of a software package. Evidence has shown that easy-to-use programming constructs can not only enhance EUC's productivity [1,15] and satisfaction [12] but also keeps the learning barriers low, which encourages the flow of applications from one department to another [40].

Although EUC has been seen in various applications from sophisticated decision support systems to structured operating systems [11], the nature and determinants of EUC's success, however, have not yet been investigated [5,8,10,40]. An investigation on human cognitive skills, information processing capacity, and programming constructs is necessary for improving the use of EUC [8,39].

One often-addressed end user programming construct is the IF-THEN-ELSE,

Houn-Gee Chen, Graduate Institute of Information Mnagement, National Chung-Cheng University, Chia-Yi, Taiwan, R.O.C.

Copyright©1997 by SMC Premier Holdings, Inc. All rights of reproduction in any form reserved.

which allows a statement or a group of statements to be selected for execution on the basis of meeting prescribed conditions. If the testing condition is true, statements after THEN are executed. Otherwise, statements after ELSE are executed. Nested IF-THEN-ELSE is a construct in which an IF-THEN-ELSE is set inside another IF-THEN-ELSE construct. Nested IF-THEN-ELSE provides a way to program multi-level decision problems and has often been applied for complicated decision processes in which one condition depends on another condition. The nesting may occur at any desired level. This construct has been applied to represent knowledge in recent expert/knowledge-based system implementations. Sets of IF-THEN-ELSE constructs are used to define the knowledge source—the knowledge base. The reasoning process of these IF-THEN-ELSE statements will support the problem solving in deriving the final solution.

This paper presents the results of an empirical study that identifies factors associated with the use of the nested IF-THEN-ELSE structure in EUC applications. It also addresses the effects of individual differences, cognitive limits, and programming constructs.

II. RELATED STUDIES

Through a study of business information systems, Lucas [25] identified five main determinants for voluntary use of computerized systems: user attitudes and perception, situational and personal factors, technical supporting quality of the system, user decision style, and management support. Among those factors, user attitudes and personal/situational factors were reported to be strongly related to implementation success of most systems, followed by decision styles (quantitative or qualitative styles), supporting quality of the system, and management support. In a similar study [54], the difficulty in implementing management information systems has also been reported to be the result of cognitive style mismatches between user and information design. For a better use of information system design, Zmud [54] suggested research into user aspiration level, anxiety, risk-taking propensity, self-esteem, and individual attitudes and values. Studies by Bariff and Lusk [2,3] confirmed the suitability of inclusion of these behavior profiles into the design guidelines and implementation procedures of information systems. Other studies for the EUC of information systems are summarized as follows:

User Attitudes

Several studies have demonstrated that user attitudes can dramatically impact the learning and performance of interactive information systems [28,34,36,48]. Novices with negative attitudes towards computers learned editing tasks more slowly and made more errors [50]. Anxiety, generated by fear of failure, may reduce short-term memory capacity and inhibit performance. If users are insecure about their ability to use the system, worried about destroying files or the computer itself, overwhelmed by a volume of details or pressured to work rapidly, their anxiety will lower performance [37]. Programmers who must meet a deadline tend to make more errors as they frantically patch a program in a desperate attempt to finish the job. Of course, mild pressure can act as a motivator; but if the pressure becomes too strong, then the

resultant high levels of anxiety could interfere with competent work [37]. In recent related studies, computer anxiety was found to be closely related to the success of EUC [19,23] and microcomputer usage [22]. Chen and Vecchio [7], however, concluded that individual difference and programming attitudes are important when determining the use of IF-THEN-ELSE constructs.

Situational Factors

Situational factors cover a broad spectrum of personal characteristics, such as experience, training, sex, age, professional orientation, and organizational level. They were reported to have a significant influence on a user's problem-solving activities. Rivard and Huff [39] found that computer background had significant effects on the success of EUC. Other results also indicated that computer experience and training were positively related to system usage [8,10,23]. Although some studies found no gender differences in attitudes toward computer systems [19], others reported that women and older individuals tended to have unfavorable attitudes [9].

Personal Factor and User Cognitive Styles

Important factors related to the success of an information system may include locus of control, ambiguity, and extroversion/introversion [18,28,42,54]. Extroverted subjects were found to retrieve information stored in their own minds more quickly and to retain information better over short intervals, but not for long intervals, when compared with introverts [14]. However, in another study, the aggressive/humble dimension and introversion/extroversion did not significantly affect the subject's grade in an introductory programming course nor the subject's perceived ability in programming [32]. Personality was reported to be a major factor in determining results on a simulated man-machine decision information system [53].

Many studies have revealed [25,26,27,54] that cognitive style is one of the important factors affecting the success of MIS/DSS. Soloway [45] showed that a programming construct that has a closer "cognitive fit" with an individual preferred cognitive model is easier to be used effectively. Cognitive style represents the characteristic mode of functioning, shown by individuals in their perceptual and intellectual response. Perception involves the input, filtration, and condensation of cues from the environment. Intellect involves analyzing and reaching a conclusion based on what is received. Although such responses are dependent on tasks and situational elements, many individuals exhibit pervasive tendencies toward particular cognitive functioning, and consistent individual differences can be observed. For example, subjects with a higher level of intelligence may be observed to process information faster, select information more effectively, retain information better, make decisions faster [46], and organize information better [21]. Subjects with higher quantitative abilities utilize more short-term memory than those subjects with lower quantitative abilities [21], and subjects with greater verbal abilities possess enhanced short-term memory when compared with subjects having lesser verbal skills [20].

Cognitive Limitation (short-term memory)

Studies [6,42,51] have also suggested a cognitive model of long-term/short-term memory to explain human information processing. Information from the outside world (e.g., a description of the problem) first enters the human cognitive system via short-term memory. The information is then integrated with existing information to form the basis of the problem solution. EUC, however, can be viewed as a cognitive process in which end users develop their own applications via a mapping between the input problem description and supporting programming constructs. Since the human cognitive process is often constrained by the *size* of the short-term memory [42,51], the effective and/or efficient use of programming constructs may be confined by this limitation. A well-known research finding [30] revealed that human short-term memory can handle about seven, plus or minus two, "chunks" of information at any given time. This implies that the human information processing capacity is extremely small [42].

Formats of IF-THEN-ELSE Constructs

Researchers have attempted to study the effects of various formats to improve the readability of the IF-THEN-ELSE construct. Works by Sime, Green, and Guest [43,44] and Green [16,17] were the pioneering studies on the IF-THEN constructs. Across several experiments, they found evidence to support the nested IF-THEN-ELSE (used in ALGOL, PL/I and PASCAL) versus IF-GOTO (used in FORTRAN and BASIC). Among the nested constructs, their results seemed to support the superiority of the negation of the condition (NOT A=B). Nested IFs may be better when the tested items are unrelated and when forward comprehension questions are asked. Deeply nested IF-THEN-ELSE statements are difficult to comprehend, but the alternative use of complex Boolean conditions is not necessarily more appealing [44]. In a later study, Vessey and Weber [49] found no clear-cut evidence favoring nested IF-THEN-ELSE over IF-GOTO. One important conclusion derived from these results is that the suggested superiority of the nested IF-THEN-ELSE over the IF-GOTO is equivocal.

The CASE construct has received much attention in recent years [49]. The advantage of the CASE statement is that a multiple branch is accomplished in a single well-organized statement that has an explicit END statement. Shneiderman [42] showed that, in some instances, the multi-level decision logic of nested IF-THEN-ELSE is not easily represented by a CASE statement. Embley [13] proposed a hybrid construct by combining the CASE and iteration in a simple manner. In an empirical experiment, the groups working with Embley's construct resulted in a significantly better comprehension question than those questions of the groups working with CASE and IF-THEN-ELSE. The differences between CASE and IF-THEN-ELSE were not significant.

In fact, two different forms of nested IF-THEN-ELSE exist: bushy and thin. It has been frequently and persuasively argued that we should avoid the "bushy" form [4]; instead, the "skinny" form is the appropriate form to use [24,38,47]. The reason for this is that the skinny form "seems to more closely reflect how we think." Green [17], along with other researchers [33], examined reasoning for programming statements expressed in different forms. He used a set of conditionals in the thin forms, while only one conditional was applied in the bushy structure. The results indicated that for the forward questions (given a conditional to induce the actions), there was no

significant difference between thin and bushy forms. For the backward questions (given actions to identify the conditionals), the thin form was slightly better than the bushy form.

While these conflicting results have existed in previous studies, we argue that there are some important factors (e.g., individual characters and cognitive limitation) that may make these results equivocal. In addition to the measurement of performance (e.g., syntax errors, debugging efforts) that had been the primary concern in the above studies, our goal was to examine the individual difference factors on the use of nested IF-THEN-ELSE constructs. This study attempts to do so by examining four often-addressed nested IF-THEN-ELSE formats: bushy, long thin, GOTO, and revised-long thin.

III. A MODEL

The above discussion suggests that end user programming ability is at least partially a function of three important sets of variables: individual differences, cognitive limitations, and formats of language construct (Fig. 1). The individual difference consists of important factors, such as individual personality traits and programming aptitude. The programming aptitude is believed to be comprised of several important aptitudes, including abstract reasoning, ability to follow computer language, and ability to deduce an appropriate outcome from information presented in a serial form. In terms of personality, previous research suggests that there are two important attributes: introversion-extroversion and computer anxiety. Introversion is now generally seen as being related to computer programming ability, as a consequence of introverted individuals, being better able to focus their attention on demanding tasks than extroverted individuals are able to do. Computer anxiety should be related to programming ability in that higher levels of anxiety will interfere with the programming ability to focus on a demanding task.

This study argues that the programming ability is likely constrained by the limitation of human short-term memory. This suggests that there exists an upper limit of nesting levels on an effective use of nested IF-THEN-ELSE constructs. In addition, we argue that an end user should have comparative advantages in developing EUC applications when he/she is supported by easy-to-use programming constructs.

Figure 1
A Model of End User Programming

In addition to testing more general propositions, the present study seeks to address the following questions concerning nested IF-THEN-ELSE constructs:

(1) What is the appropriate nesting level in designing nested IF-THEN-ELSE constructs?

Since three-level and four-level IF-THEN-ELSE constructs carry up to 8 (2^3) and 16 (2^4) chunks of information, respectively, the following hypothesis is tested:

Hypothesis: When the nesting level is larger than three, the end user's programming ability will decrease significantly.

(2) Under a reasonable nesting level, which IF-THEN-ELSE formats are easy-to-use? and

(3) What are the important individual difference factors associated with the effective use of various IF-THEN-ELSE constructs?

IV. RESEARCH METHODOLOGY

The following research methodology was used.

Testing Problem

A simple credit-evaluation example was chosen as the testing problem. A department store offered credit cards for its customers. The credit limits were determined according to the applicants' incomes. The following table summarizes the income categories, credit limits, and the corresponding monthly incomes.

Category	Credit limit	Monthly Income
(1)	0	$ 500 or less
(2)	$ 100	$ 501-750
(3)	$ 200	$ 751-1000
(4)	$ 300	$1001-1250
(5)	$ 400	$1251-1500
(6)	$ 600	$1501-2000
(7)	$1000	$2001-3000
(8)	$1500	$3001-4000
(9)	$2000	$4001-5000
(10)	$2500	$5001 or more

Performance was assessed by tallying the *number of correct answers* and The credit limits were displayed using various IF-THEN-ELSE formats. This experiment studied the way that subjects received information from the displays. Subject the *response times* given to the various types of programming construct

presentations.

Experimental Prototype

An interactive menu-driven computer program was developed to conduct the experiments. This prototype was running in the network and was accessed through a file server. Three sections were included. In the first section, subjects registered by entering geographical data. This was followed by a tutorial section where subjects went through several menu-driven examples to familiarize themselves with various keystroke functions and locations. A question-and-answer section was the third section. Here information was presented using various IF-THEN-ELSE formats on the left side of screen while the possible solutions were summarized in a menu on the right side of screen. Each time, a number was given to indicate an applicant's monthly income. Subjects were asked to determine the credit limit and to perform an appropriate keystroke by looking up the information. A sample display is given in Fig. 2.

Figure 2
Display of Thin Format (I: Income)

A monitor was built in to trace the individual subject's responses. This monitor
was activated to record the individual subject's answers and response times once a question was posted. The detailed step-by-step instructions were also incorporated so that subjects could control the pace of the experiment. This prototype was run on a PC.

Subjects

Subjects were students in a required, sophomore-level introductory business computer course taught in a university. This course introduced the basic concepts of computer systems and computer applications software such as spreadsheet, data base management, and computer programming languages. The subjects possessed several characteristics in common; that is, they anticipated a continuing involvement with computers while not being experts in programming.

Experiment Design

Experiments were conducted during the middle of the semester; and, by then, the subjects had been introduced to basic concepts of the computer. A lecture regarding the content of the experiments was given. A computer lab was reserved for conducting the experiments. Participation in those experiments was a course requirement.

In a regular class meeting, all subjects completed booklets containing questions related to demographic data (gender, age, prior computer coursework, college-entrance exam scores) and the following instruments.

(a) The extroversion scale from the Myers-Briggs Type Indicator [31]. Two different item formats are represented in the scale: force-choice responses for activities and self-descriptive adjectives and true-false responses to self-descriptive statements;

(b) A 10-item computer anxiety scale created by modifying items of the Mathematics Attitude Inventory [41] to reflect computer anxiety rather than math anxiety (sample items: "It scares me to have to study computers." and "Solving a computer problem is fun." (Reversed); Responses: 4=strongly agree, 3=agree, 2=disagree, 1=strongly disagree.);

(c) The programming language questions (problem set 3) from the Wolfe Programming Aptitude Test-Form W [52]. These questions require that the respondent translate a number of language-like,symbolic expressions into simplified values (answers) based on a set of given definitions of terms and operations;

(d) The series completion aptitude scale from the CALIP (Computer Aptitude, Literacy and Interest Profile form [35]). The series task, which requires completion of a number and/or letter sequence, identifies individuals who can detect internal and highly structured order (sample item: Complete the following, A,1,D,4,J,10,M,_).

One hundred forty-three students participated in the experiments. Most subjects were first-time users of computers.

Nesting Level

Four IF-THEN-ELSE constructs were designed (Fig. 3) in this experiment. Among them, Seven IF-THEN-ELSES and Eight IF-THEN-ELSES were three-levels constructs; Nine and Ten IF-THEN-ELSES were four-levels constructs. Subjects were equally assigned to each sequence. At *each construct*, five cases (634, 5373, 1212,

2736, 3879) were generated and tested. At each test, subjects were asked to determine the credit limit by referring to a presented IF-THEN-ELSE construct.

Figure 3
Nested IF-THEN-ELSE Structures

Figure 4
Various Nested IF-THEN-ELSE Formats (I: Income; Y: Yes; N: No)

Display Formats

Four display formats were designed (Fig. 4) in this experiment. In the long thin format (Thin), all of the nesting occurred within a THEN block or an ELSE block. The revised long thin format was similar to the long thin format except that it explicitly stated conditions for each IF statement. The bushy format (Bushy) was opposite to the long thin format in a way that an IF-THEN-ELSE might be contained within either a THEN block or an ELSE block. In the GOTO format, jump (GOTO) constructs were incorporated into the THEN or ELSE blocks such that the program logic flow didn't necessarily follow a "sequential" fashion.

At *each construct*, five cases (2687,1341,5054,3742,712) were generated and tested. Subjects were asked to determine the credit limit by referring to a presented IF-THEN-ELSE construct for each test. A sample display is given in Fig. 5.

Figure 5
Display of Cond Format (I: Income)

V. RESULTS

A breakdown on the subject's demographic information (Table 1) indicated that the college-entrance exam scores were close to the university's average. Although most of the students had taken college-level mathematics courses in their freshman year, very few had learned computer programming before so that most subjects were computer

novices.

Table 1
Demographic Data

Number of Subjects	
male	84
females	59
total	143

College-entrance Exam Scores

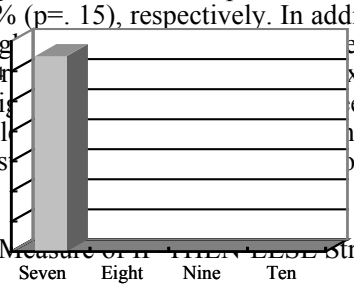
verbal(mean)	69.72
math(mean)	58.02

Computer courses taken before

number	count	percentage
0	9	6.3
1	89	62.2
2	22	15.4
3	16	11.2
4	4	2.8
5	3	2.1

Effects of Short-Term Memory

The subjects' performances, using various nesting-levels of IF-THEN-ELSE constructs, were tallied (Figure 6). The quality of answers measures the correctness of subjects' answers, and the "response time" is the time for the subject to answer all of the questions. The results indicate that the quality of answers has a significant 4.8% (p=. 05) drop from Eight to Nine. The drops from Seven to Eight and Nine to Ten are 2.5% (p=. 27) and 3.6% (p=. 15), respectively. In addition, the response time increases significantly from Eig en (a 20% and 4.4% increase, respectively). These xists a significant performance difference between Eig e-level construct) and Nine IF-THEN-ELSES(a four hat for a better use of IF-THEN-ELSE construct the nes th



Performance Measure of IF-THEN-ELSE Structure (Nesting Levels)

Effects of IF-THEN-ELSE Formats

Figure 7 depicts the subjects' performances on various IF-THEN-ELSE formats. Table 2 presents the results of an analysis of variance test for the performance (i.e., comparing the average level of performance across four nested IF-THEN-ELSE formats). This result was followed by paired comparison tests contrasting each of the types of nesting formats. The order of quality of answers followed the sequence of "Cond-Thin-GOTO-Bushy" with Cond having the highest quality. The order of response times followed the same sequence of "Cond-Thin-GOTO-Bushy". With the exception of the Thin-Cond contrast, all contrasts yielded significantly different results ($p<.01$). The results suggest that Cond and Thin are less complex formats than GOTO and Bushy.

Figure 7
Performance Measure of IF-THEN-ELSE Structures (Display Formats)

Table 2
ANOVA of IF-THEN-ELSE Formats

Source	Sum of Squares	Degree of Freedom	Mean Squares	F	p>F
Between formats	200.76	3	66.92	27.71	0.000
Error	1371.85	568	2.42		

Total	1572.61	571

Effects of User Attributes

To further understand the effects of individual differences on various IF-THEN-ELSE formats, bivariate correlations of individual difference measures with the dependent measures (i.e. nested IF-THEN-ELSE formats) were calculated (Table 3). The results indicate that the computer anxiety (inversely), lang test, and series aptitude were consistently related with performance on the programming ability (i.e., quality of answers). The computer anxiety and series aptitude were also related with performance on the response time. As the present study was most centrally concerned with the issue of whether personality and aptitude uniquely predicated programming ability, a hierarchical multiple regression approach was adopted. With this approach, the unique effect of each independent variable (in terms of incremental criterion variance accounted for) is calculated and tested for statistical significance, following comparisons of the R^2 value for an equation that contains all five independent variables and the R^2 for an equation that excludes the independent variable of interest. The results of such tests for each dependent measure of programming ability are displayed in Table 4.

Table 3
Intercorrelations Among Variables (decimal points omitted)

Table 4
Results of Tests for Unique Effects of Independent Variables

Rresults in Table 4 suggest that each independent variable does contribute to explaining the performance (i.e.,quality of answers) in the programming test. Among the independent variables, computer anxiety, series aptitude, and lang test have the most significant impacts. The evidence for explaining the response time is partially supported by the lang test and computer anxiety. In addition, extroversion subjects perform extremely well on GOTO format.

Another question concerns whether personality traits and aptitudes have separate effects. In order to test this issue, multiple regression equations were created for each dependent measure for the *set of personality tests* (extroversion and computer anxiety) and the *set of aptitudes* (language aptitude and series aptitude). The unique effects of each set were then calculated by calculating the R^2 for the complete equation (which included both the personality and aptitude measures) with equations that included only the set of personality measures or the set of aptitude measures. The results of statistical tests of significance and changes in R^2 are reported in Table 5. These results suggest that the unique effects of each set of independent variables do explain the performance on programming ability tests (i.e., quality of answers). The set

of aptitudes also contributes to the explanation of performance on the response time tests.

Table 5
Results of Tests for Unique Effects of Personality and Aptitude Variables

VI. DISCUSSIONS

The present study confirms the importance of aptitude and personality factors in influencing performance (i.e, quality of answers) on programming tasks. For example, individuals who are lower on computer anxiety generally achieved a higher score on the programming tasks. Specific aptitudes were also positively correlated with performance across tasks as well. Similarly, individuals who were low on extroversion tended to perform better than the more extroverted individuals on certain programming tasks.

From a practical perspective, these results suggest that prospective end users might be assessed for both personality and aptitude characteristics. Certain of these characteristics are not likely to be amenable to modification via training. However, computer anxiety may be responsive to hands-on experience and education. Given available evidence that computer anxiety may play a key role in influencing managerial attitude toward using microcomputers [23], the targeting of computer anxiety for "treatment" seems warranted. Thus, in the case introduced at the start of this paper, Steven Anderson should request more computer training programs to assist

end users in exploring productive applications.

In addition, this study confirms that the constraint of human short-term memory has a notable impact on programming ability. To ensure the success of EUC, Steven should be aware of this and avoid any "overload" misconduct in programming.

The differential results for the types of "nesting" suggest that performance (quality of answers) in the Cond format was the highest of the four formats. Formal contrast of the means of the four formats revealed that performance with the Cond and Thin formats was significantly higher than performance for each of the remaining formats. These differential results suggest the relative superiority of the long thin and revised-long thin formats for instructional purposes. However, the continuous supporting of easy-to-use programming constructs will be a goal for future information systems development.

Because the present study employed a sample of college students, the results cannot be readily generalized to programmers in general. However, the present sample offers the potential advantage that broader ranges were likely to be observed on the variables of interest, as individuals seeking a career in programming may represent only a limited portion of the possible ranges. In short, the sample enabled an establishment of the probable upper limit of likely associations among variables.

In conclusion, the results suggest that programming ability can be predicted from personality and aptitude measures. The demonstration that programming ability is sensitive to individual differences further strengthens a growing perspective that programming ability is not an independent trait.

ACKNOWLEDGEMENTS

The author would like to thank Drs. Sing-Ling Lee, Nai-Wei Lin, Dong-Her Shih, and D. T. Zhang for their supports in conducting the experiments.

REFERENCES

[1] Baily, J. and S. W. Pearson, (1983). "Development of a Tool for Measuring and Analyzing Computer User Satisfaction," *Management Science*, 29(5), 530-545.

[2] Barriff, M. L. and E. J. Lusk, (1977). "Cognitive and Personality Test for the Design of Management Information Systems," *Management Science*, 23(8), 820-829.

[3] Barriff, M. L. and E. J. Lusk, (1978). "Designing Information System for Organizational Control: The Use of Psychological Tests," *Information and Management*, 1 (3), 113-121.

[4] Barron, D. W. (1977). *An Introduction to the Study of Programming Languages*, Cambridge: Cambridge University Press.

[5] Benson, David (1983). "A Field Study of End User Computing: Findings and Issues," *MIS Quarterly*, (December), 35-45.

[6] Brooks, R. (1977). "Towards a Theory of the Cognitive Processes in Computer Programming," *International Journal of Man-Machine Studies*, 9, 737-751.

[7] Chen, H. G. and R. Vecchio, (1992). "Nested IF-THEN-ELSE Constructs in

End-User Computing: Personality and Aptitude as Predicators of Programming Ability," *International Journal of Man-Machine Studies*, 36, 843-859.

[8] Cheney, P. H., R. I. Mann, and D. L. Amoroso, (1986). "Organizational Factors Affecting the Success of End-User Computing," *Journal of Management Information Systems*, 3(1), 65-80.

[9] Dambrot, F. H., M. A. Watkin-Malek, M. S. Silling, R. S. Marshall, and J. A. Garver, (1985). "Correlates of Sex Differences in Attitudes Toward and Involvement with Computers," *Journal of Vocational Behavior*, 27(1), 71-86.

[10] DeLone, W. H. (1988). "Determinants of Success for Computer Usage in Small Business," *MIS Quarterly*, 12(1), 51-61.

[11] Dickson, G. W., R. L. Leitheiser, J. C. Wetherbe, and M. Nechis, (1984). "Key Information Systems Issues for the 1980's," *MIS Quarterly*, 8(3), 135-159.

[12] Doll, W. and G. Torkzadeh, (1988). "The Measurement of End-User Computing Satisfaction," *MIS Quarterly*, (June), 259-274.

[13] Emply, D. W. (1978). "Emperical and Formal Language Design Applied to a Unified Control Structure," *International Journal of Man-Machine Studies*, 10, 197-216.

[14] Eysenck, M. W. (1977). *Human Memory: Theory, Research and Differences*, Oxford: Pergamon Press.

[15] Goodwin, N. C. (1987). "Functionality and Usability," *Communications of the ACM*, 30(3), 229-233.

[16] Green, T. R. G. (1977). "Conditional Program Statements and Their Comprehensibility to Professional Programmer," *Journal of Occupational Psychology*, 50, 205-216.

[17] Green, T. R. G. (1980). "IFs and THENs: Is Nesting Just for the Birds?' *Software-Practice and Experience*, 10, 373-381.

[18] Helander, M. Ed. (1988). Amsterdam: North-Holland. *Handbook of Human-Computer Interaction*

[19] Howard, G. S. and R. Smith, (1986). "Computer Anxiety in Management: Myth or Reality?," *Communications of the ACM*, 29(7),611-615.

[20] Hunt, E., N. Frost, and C. Lunneborg, (1973). "Individual Differences in Cognition: A New Approach to Intelligence," in G. H. Bower,(ed.) *The Psychology of Learning and Memory*, New York: Academic Press.

[21] Hunt, E., and M. Lansman, (1975). "Cognitive Theory Applied to Individual Differences," in W. K. Estes,(ed.) *Handbook of Learning and Cognitive Processes*, Hillsdale, N. J: Lawrence Erlbaum Associates.

[22] Igbaria, M., F. N. Pavri, F. N. and Sid L. Huff, (1989). "Microcomputer Applications: An Empirical Look at Usage," *Information and Management*, 16, 187-196.

[23] Igbaria, M. and S. Parasuraman, (1989). "A Path Analytic Study of Individual Characteristics, Computer Anxiety and Attitudes Toward Microcomputers," *Journal of Management*, 15(3), 373-388.

[24] Kernighan, B. W. and P. M. Plauger, (1974). "Programming Style:Examples and Counter-examples," *Computer Surveys*, 6, 303-319.

[25] Lucas, H. C. (1978). "Empirical Evidence for a Descriptive Model of

Implementation," *MIS Quarterly*, 2(2),27-41.
[26] Lucas, H. C. and N. Nielsen, (1980). "The Impact of the Mode of Information Presentation on Learning and Performance," *Management Science*, 26(10), 982-993.
[27] Mason, R. O. and I. I. Mitroff, (1973). "A Program for Research on Management Information System," *Management Science*, 19(5), 475-487.
[28] Matta, K. M. and G. Kern, (1991). "Interactive Videodisc Instruction: The Influence of Personality on Learning," I*nternational Journal of Man-Machine Studies*, 35, 541-552.
[29] Mclean, E. R. (1979). "End User as Application Developers," *MIS Quarterly*, (December), 37-46.
[30] Miller, G. A.(1956). "The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information," *Psychol. Rev.*, 63, 81-97.
[31] Myers, I. B. and M. H. McCaulley, (1988). *Manual: A Guide to the Development and Use of M.B.T.I.* Palo Alto, CA: Consulting Psychologists Press.
[32] Newsted, P. R. (1975). "Grade and Ability Prediction in an Introductory Programming Course," *ACM SIGLSE Bulletin*, 7(2), 87-91.
[33] Ormerod, T. C., K. I. Manktelow, E. H. Robson, and A. P. Steward, (1986). "Content and Representation Effects with Reasoning Tasks in Prolog Form," *Behavior and Information Technology*, 5 (2), 157-168.
[34] Pask, G. (1976). "Styles and Strategies of Learning," *British Journal of Educational Psychology*, 46, 128-148.
[35] Poplin, M. S., D. E. Drew, and R. S. Gable, (1984). *CALIP-Computer, Aptitude, Literacy, and Interest Profile*, Austin, TX: Pro-ed.
[36] Pracht, W. E. and J. F. Courtney, (1988). The Effects of an Interactive Graphics-Based DSS to Support Problem Structuring, *Decision Science*, 19,598-621.
[37] Raub, A. C. (1981). *Correlates of Computer Anxiety in College Students*. Unpublished Ph.D. Dissertation, University of Pennsylvania, PA.
[38] Richards, M. (1976). "Programming Structure, Style and Efficiency", in *Structured Programming: An Infotech State-of-the-Art Report*, 13-28, Maidenhead: Infotech.
[39] Rivard, S. and S. L. Huff, (1988). "Factors of Success for End-User Computing," *Communications of the ACM*, 31(5), 552-561.
[40] Rockart, J. F. and L. S. Flannery, (1983). "The Management of End User Computing," *Communications of the ACM*, 26(10), 776-784.
[41] Sandman, R. S. (1979). *Mathematics Attitude Inventory*, Minnesota: Research and Evaluation Center, University of Minnesota.
[42] Shneiderman, B. (1980). *Software Psychology* Little, Brown and Company.
[43] Sime, M. E., T. R. G. Green, and D. J. Guest, (1973). "Psychological Evaluation of Two Conditional Constructions Used in Computer Languages," *International Journal of Man-Machine Studies*, 5, 123-143.
[44] Sime, M. E., T. R. G. Green, and D. J. Guest, (1977). "Scope Marking in Computer Conditionals-- A Psychological Evaluation," *International Journal of Man-Machine Studies*, 9, 107-118.
[45] Soloway, E., J. Bonar, and K. Ehrlich, (1983). "Cognitive Strategies and

Looping Constructs: An Empirical Study," *Communications of the ACM*, 26(11), 853-860.

[46] Taylor, R. N. and M. D. Dunnette, (1974). "Relative Contribution of Decision-Maker Attributes to Decision Process," *Organizational Behavior and Human Behavior*, 12, 286-298.

[47] Tracz, W. J. (1979). "Computer Programming and Human Thought Process," *Software-Practice and Experience*, 9, 127-137.

[48] Van der Veer, G. and Wolde J. van der Veer (1983). "Individual Differences and Aspects of Control Flow Notation," in T. R. G. Green, S. J. Payne and van der Veer (Eds.) *The Psychology of Computer Use*, London: Academic Press.

[49] Vessey, I. and Ron Weber, (1984). "Research on Structured Programming: An Empiricist's Evaluation," *IEEE Transactions on Software Engineering*, SE-10(4), 397-407.

[50] Walthner, G. H. and H. F. O'Neil, (1974). "On-Line User-Computer Interface: The Effects of Interface Flexibility, Terminal Type, and Experience on Performance," *Proceedings* of the National Computer Conference, AFIPS press, Montuale, NJ.

[51] Weinberg, G. M. (1971). *The Psychology of Computer Programming*, Van Nostrand Reinhold Company.

[52] Wolfe, J. M. (1982). *Wolfe Programming Aptitude Test,* Oradell, NJ: Wolfe Testing Ltd.

[53] Wynne, B. E. and G. W. Pickson, (1976). "Experienced Managers' Performance in Experimental Man-Machine Decision System Simulation," *Academy of Management Journal*, 18(1), 25-40.

[54] Zmud, R. (1979). "Individual Differences and MIS Success: A Review of the Empirical Literature," *Management Science*, 25(10),966-979.