教育研究學報 1993, 第八卷, 68-73

Educational Research Journal 1993, Vol. 8, pp.68-73

Effect of Flowcharting on Program Composition Skill

Au Sai-Kit
SKH Leung Kwai Yee Secondary School

CHUNG Choi-man
The Chinese University of Hong Kong

Flowcharts were widely used as program organization tools and were assumed to have positive effects on program development. The purpose of this study was to investigate the effect of flowcharting on developing programs. When flowcharting was not required in program composition, subjects performed better in program logic, semantic correctness and syntactic correctness. Flowcharting seemed to be a redundant task in program development and it did not facilitate the construction of program logic. Male subjects generally performed better than female subjects in some aspects of programming performance. Home computer possession seemed to increase the familiarity with computer and lower the computer anixety, hence, subjects who possessed home computers generally performed better than those who did not.

在組織電腦程式的過程中，流程圖已被廣泛地應用，而且相信它能夠協助編寫程式。本研究之目的是探求編寫流程圖對程式編寫的影響。當受試者不需要繪畫流程圖時，他們在程式邏輯，語意及語法上，都有較佳的表現。男性受試者在程式編寫中，亦有較佳的表現。家中擁有電腦的受試者，因為對電腦的焦慮程度較低，在程式編寫的表現中，亦比家中沒有電腦的受試者為好。

Program flowcharts show the explicit flow of control and also prompt the completioh of the program logic. As a result, flowcharting seems to help the programmer to organize the program logic during the composition of programs (McAllister & Brock, 1990). Therefore, flowcharts are used as the major program design tools and they are also required to show the logic of program in the documentation. In the study of the effect of computer access on programming performance, for the moderate and higher ability students, it was found that the limited access group has a better performance than the free access group (McCormick & Ross, 1990). It was explained that the limited-access group were forced to think and plan before they went to the computers. For the unlimited-access group, the students were urged to key in lines of code when they got access to a computer. There was no organization stage in this group. The study revealed that an program organization stage was essential for a good programming performance.

Although flowcharts do not capture all the characteristics of good program design tools, they are the most commonly used program design tools in high schools. In Hong Kong, the students are asked to design the algorithm to solve the problem in flowcharts and code the flowcharts into BASIC programs before they go to the computer room. The students are asked to follow the problem solving procedures (see Appendix A) to solve complicated problems, and they are taught to use flowcharts as the organization tools in designing algorithms to solve the problems (Curriculum Development Committee, 1986).

Flowcharts seem to add direction to the programs of students during the construction of programs. Flowcharting has been considered as art requiring practice and that a flowchart should be developed before a program is coded. In an experiment on the effect of instruction on the misconceptions in BASIC language, it was observed that the subjects who were required to develop a flowchart, spent less time in developing and debugging a program (Stemler, 1989). It was explained that the use of flowcharting technique forces the student to look more at the solution of the problem in the very beginning stages of software development for the early identification and removal of errors.

Conflicting results were found by other researches (McCormick & Ross, 1990; Shneiderman et al., 1977). In the study of the effect of computer access on programming performance, half the subjects in the limited access group was required to submit flowcharts and the other half was not. It was found that the non-flowchart group performed better than the flowchart group (McCormick & Ross, 1990). It was explained that many students do not adequately master flowcharting skills. Flowcharting, because of its pictorial modality and unique symbol system, appears to be regarded as a separate task by most of the students. Some of the students may complete the program first and then draw the flowchart accordingly as to fulfill the requirement of the teachers. It may concluded that the subjects in the limited access group perform better because they are forced to plan before to key in their programs. The students may have their plan 'organized' abstractly in their mind or in their own set of codes and formats. This finding puts forth another question – Does flowcharting really help in the organization of program logic? This study tends to investigate the effect of flowcharting on the program composition skills.

TABLE 1
Characteristics of Subject

Characteristics		Number of Subjects
Sex	Male	112
	Female	62
Home Computer	Possessed	83
	Not possessed	91
Repeater	Repeater	15
	Non-repeater	159
Other Computer course attained	No	154
	Yes	20

Method

Procedure

The subject were 174 Form 5 students selected from 8 Anglo-Chinese coeducational schools (see Table 1). All the subjects had studied Computer Studies for about 16 months. A pretest which consisted of 30 multiple choice items, was first given to the subjects to assess their knowledge in BASIC language and flowcharts. The subjects were required to respond a questionnaire just before the pretest. Questions were set to collect the following information: name, sex, age, access to home computer, previous computer course attended, repeat F.5 or not, and other relevant information. The experiment were performed one week after the pretest. The subject were divided into two groups (Flowchart vs on-flowchart) of equal abilities according to the pretest scores. In the experiment, the subjects were required to write programs to solve two problems. The subjects in the Flowchart group were required to draw a flowchart and then write the program accordingly, while the subjects in the Non-flowchart group were asked to write the program directly without going through the process of flowcharting. The time given was 70 minutes which was quite sufficient since all the subjects finished the two problems within the given time.

Statistical Analysis

The experiment is a two by two factorial design with methods of writing program (Flowchart vs on-flowchart) and abilities of subjects (Low-ability vs High-ability) as between group factors. In the experiment, the subject has to write two programs, one is a high-difficulty problem and the other is a low-difficulty problem. The low-difficulty problem requires branching and a single looping control structure while the high-difficulty problem requires branching and a nested-looping control structure (see Appendix B).

The programming performance was assessed in three aspects: the program logic, the program semantics and the program syntax. The marking scheme was modified from that used by Van Merrienboer (1990). The marking of the program composition test was conducted in three steps. Firstly, the number of the number of correctly and incorrectly coded lines were counted. A syntax correctness score was expressed as the percentage of correctly coded lines. Secondly, the logic score was found by counting the number of logic elements present in the program. Finally, the marker scored the semantic correctness of each program on the five-point scale (see Appendix C) which was modifiedfrom that used in the research of Van Merrienboer, 1990.

The marking of scripts were conducted by the researcher. About one quarter of the raw scripts were photocopied and marked by an experienced computer studies teacher who was unfamiliar with the experiment. The rater gave only the logic scores and semantic correctness score for each of the problems, since these two scores were more subjective in the marking system. The interrater coefficients of the logic scores of the low-difficulty problem and the high-difficulty problem were 0.939 (N = 41) and 0.931 (N = 41) respectively, and those of the semantic correctness scores of the low-difficulty problem and the high-difficulty problem were 0.864 (N = 41) and 0.930 (N = 41) respectively.

Results and Discussion

Three different scores (logic score, semantic correctness score, and the syntactic correctness score) were given to each of the problems. The logic score ranged from 0 to 10, the semantic correctness score ranged from 0 to 5, and the syntactic correctness score which was a percentage of correctly coded lines, ranged from 0 to 100. The mean and standard deviation of the scores for different groups were shown in Table 2.

TABLE 2 .
Analysis of Variance by Method of Writing Program and Ability of Subjects

Variable	Mean of Non-flowchart (N = 87)	Mean of Flowchart (N = 87)	*df*	*F*
Low-difficulty problem				
Logic score	6.632(3.024)	6.437(2.692)	1/123	0.031 n.s.
Semantic correctness	3.356(1.257)	3.207(1.183)	1.123	0.358 n.s.
Syntactic correctness	91.16(15.73)	88.56(20.45)	1/123	0.282 n.s.
High-difficulty problem				
Logic score	3.333(4.195)	2.081(2.211)	1.123	4.151*
Semantic correctness	1.839(1.354)	1.276(1.148)	1/123	5.171*
Syntactic correctness	76.19(36.68)	62.74(41.61)	1/123	2.856 n.s.

* p <0.05
n.s. = non-significant
Standard deviation enclosed in brackets.

Generally speaking, the Non-flowchart group performed better than the Flowchart group in all the three scores in programming performance (see Table 2). In the analysis of variance, significant differences were only found in the logic scores F (1,123) = 4.151, mse = 12.835, $p < 0.05$) and semantic correctness scores (F(1,123) = 5.171, mse = 1.481, $p < 0.05$) of the high-difficulty problem.

For the high-difficulty problem, the Non-flowchart group performed better than the Flowchart group in the logic scores and the semantic correctness scores. This indicated that flowcharting did not really facilitate the construction of program logic. The students in the Flowchart group had to construct the flowcharts before writing the programs. Since most of the students did not handle flowchart well (Ramsey Atwood & Doreen, 1983; Darbey, tourniaire & Linn, 1986), this hindered the students in the construction of program logic and this was reflected in the logic scores. Difficulties might be found when the program was written according to the flowchart since the logic presented in the flowchart was not complete. This affected the overall semantics of the program and caused a low score in the semantic correctness.

There was no significant difference in the syntactic correctness scores between the two groups for both the low-difficulty and high-difficulty problems. The syntactic correctness of a program depended on the knowledge of BASIC of the students. Since the two groups were matched with pretest scores, both groups should have the same level of knowledge in BASIC language. Therefore, there should be no significant difference in the syntactic correctness scores between the two groups. The results showed that no significant interaction effect between groups was found in the three scores of programming performance of both the low-difficulty and high-difficulty problems.

As seen in Table 3, the male students performed generally better than the female students in all the three scores of both the low-difficulty and high-difficulty problems.

Previous studies revealed that male students generally had a more logical mind than the female students, and this was reflected in the better performance of male students in mathematics and programming (McCoy, 1991; Hall & Cooper, 1991). In this experiment, male subjects performed gerenally better than the female subjects but significant differences in F ratio were only obtained for the semantic correctness scores of both the low-difficulty (F = 4.132, df = 1, mse = 0.940,

TABLE 3
An Analysis of Covariance of Programming Performance by Sex with Pretest Score as Covriate

Variable	Mean of Male (N = 112)	Mean of Female (N = 62)	*df*	*F*
Low-difficulty problem				
Logic score	7.13(2.64)	5.47(2.95)	1/171	3.035 n.s.
Semantic correctness	3.54(1.11)	2.81(1.27)	1/171	4.132 *
Syntactic correctness	92.79(13.03)	84.56(24.29)	1/171	1.054 n.s.
High-difficulty problem				
Logic score	3.08(2.94)	2.03(4.05)	1/171	1.672 n.s.
Semantic correctness	1.77(1.37)	1.16(1.01)	1/171	4.605*
Syntactic correctness	72.95(38.09)	63.16(42.00)	1/171	0.499 n.s.

* p < 0.05. n.s. = non-significant Standard deviation enclosed in brackets.

p < 0.05) and high-difficulty problems (F = 4.605, df = 1, mse = 1.943, p < 0.05).

As seen in Table 4, the subjects with home computer performed generally better than the subjects without home computers. Significant differences were found in the logic scores (F(1,171) = 5.487, mse = 4.891, p < 0.05) and the semantic correctness scores (F(1,171) = 7.528, mse = 0.922, p < 0.01) of the low-difficulty problem.

Students possessing home computer certainly had more chance to get access to computers. The students might not use computers for programming activities, but, when they operated the computers, they had to use the language of the operating system. They should know the semantics and syntax of the command lines when they keyed into the computer. In order to complete a certain task, they had to key in several command lines in a logical order. This was the concept of programming. As a result, these students had more training in writing programs. On the other hand, students possessing home computers were more familiar with computers, thus, they had a low degree of anxiety towards computer. A low computer anxiety might cause a better programming performance (Loyd & Gressand, 1984), therefore, students with home computers should have better performance in programming.

Conclusion

The purpose of this study is to examine the effect of flowcharting on the programming composition skills of students. During the experiment, the subjects in the Flowchart group were asked to draw a flowchart first and then wrote the program accordingly. This was only the ideal working procedure for the Flowchart group. In some of the raw scripts, the BASIC program was written on the first page, and the flowchart was drawn on the second page. It suggested that, these subjects wrote the program first and then drew the flowchart separately, ignoring the logic developed by flowcharting. Some of the subjects even wrote the program first and then drew the flowchart according to the logic of the program as to fulfill the requirement. In this research, the ability of the subjects was most strongly and consistently related to programming performance which was measured in terms of the logic score, semantic correctness score and the syntactic correctness score. Gender differences among subjects were related to some aspects of programming performance, program logic and semantic correctness. It was found that male students generally performed better than female students in program composition. Home computer possession seemed to reduce the computer anxiety and led to a better performance in the experiment.

Results from this research indicate that flowcharts may not be the most suitable tool for the development of programs. Previous results showed that a program organization phase was essential for good programming performance. Students were encouraged to organize the program logic before writing the program. Most students tend to have their organized plan in mind, but it does not seem helphul in the development. Some kinds of tools should be used to figure out the logic and prompt a completion of logic. Flow-

TABLE 4
An Analysis of Covariance of Programming Performance by Home Computer Possession with Pretest Scores as Covariate

Variable	Mean of Not-possessed (N = 83)	Mean of Possessed (N = 91)	*df*	*F*
Low-difficulty problem				
Logic score	5.98(3.03)	7.04(2.60)	1/171	5.487 *
Semantic correctness	3.01(1.30)	3.53(1.09)	1/171	7.528**
Syntactic correctness	86.97(22.18)	92.49(13.28)	1/171	2.914 n.s.
High-difficulty problem				
Logic score	2.63(3.76)	2.78(3.05)	1/171	0.009 n.s.
Semantic correctness	1.48(1.14)	1.63(1.40)	1/171	0.2113 n.s.
Syntactic correctness	67.98(39.17)	70.82(40.32)	1/171	0.052 n.s.

** $p < 0.01$ n.s. = non-significant
* $p < 0.05$ Standard deviation enclosed in brackets.

charts seem to have these functions but they were outweighed by their shortcomings. Detailed flowchart are merely a redundant presentation of the information stored programs.

In summary, the experiments have demonstrated that flowcharting shows no significant effect on program composition. It is suggested that a better program organization tool should be searched, or a number of different organization tools should be introduced to the students such that the students can find an organization tool that suits them most. Further research should be carried out to investigate the effects of different program organization tools on programming abilities.

References

Curriculum Development Committee (1986). *Syllabuses for secondary schools – Computer Studies* (Forms IV – V), 7-8. Hong Kong.

Dalbey, J., Tourniaire, F., & Linn, M.C. (1986). Making programming instruction cognitively demanding: an intervention study. *Journal of Rearch in Science Teaching, 23*, 421-436.

Hall, J., & Cooper, J. (1991) Gender, experience and attributions to the computer. *Journal of Educational Computing Research, 7(1)*, 51-60.

Linn, M.C., & Dalbey, J. (1985). Cognitive consequences of Programming instruction: instruction, access and ability. *Educational Psychologist, 20*(4), 191-206.

Loyd, K.H., & Gressard, C. (1984, Winter), The effects of sex, age, and computer Experience on Computer Attitudes, *AEDS Journal*, 67-77.

Marcoulides, G.A. (1988). The relationship between computer anxiety and computer achievement *Journal of Educational Computing research 4*(2), 151-157.

Mayer, R.E. (1981). The psychology of how novices learn computer programming. *Computing Surveys, 13*(1), 121-141.

McAllister, D.W., & Brock, F. J., Jr (1990). Using the logic diagram in teaching programming: an overview and preliminary test results. *Journal of Research on Computing in Education 22*(3), 348-363.

McCormick D., & Ross S.M. (1990). Effects of computer access and flowcharting on students' attitudes and performance in learning computer programming. *Journal of Educational Computing Research, 6*(2), 203-214.

McCoy, L. P. (1991). Computer programming experience and mathematical problem solving. *Journal of Research on Computing in Education, 6* (1), 14-24.

Ramsey, H.R., Atwood, M.E., & Van Doren, JCR. (1983). Flowcharts versus .program design languages: An experimetal comparison. *Communications of the ACM, 26*(6), 445-449.

Sectional Committee X3, Computers and Information Processing, operating under the auspices and procedure of the American Standards Association (1963). Proposed American standard flowchart symbols for information processing. *Communications of the ACM, 6* (10), 601-604.

Shneiderman, B., Mayer, R., McKay, D., & Heller, B. (1977). Experiemental investigations of the utility of detailed flowcharts in programming. *Communications of the ACM, 20*(6), 373-381.

Stemer, L. (1989). Effects of instruction on the misconceptions about Programming in BASIC. *Journal of Research on Computing in Education, 22* (1), 26-34.

Van Merrienboer, Jeroen J.G. (1990). Strategies for programming instruction in high school: program completion vs program generation. *Journal of Educational Computing Research, 6*(3), 265-286.

Appendix A

Stages of Problem Solving Procedures (CDC Hong Kong, 1986)

1 Problem definition
2 Problem analysis
3 Design of an appropriate algorithm
4 Program coding
5 Program testing and debugging
6 Program documentation

Appendix B

1. Problem set for the Non-flowchart group
Low-difficulty problem:

Write a program to find the highest average mark and the lowest average mark of 10 students. The Chinese mark (CM), the English mark (EM) and the Mathematics mark (MM) of each student are inputted. The highest average mark (HM) and the lowest average mark (LM) should be outputted at the end of the program. The output should be corrected to one decimal place.

High-difficulty problem:

Write a program which accepts a number (NUM) and determines whether it is a prime number or not. It is a prime number, a sentence ("IT IS A PRIME NUMBER") should be printed, otherwise the product of prime factors should be printed.

2. Problem set for the Flowchart group
Low-difficulty problem:

Draw a detailed flowchart and then write a program accordingly to find the highest average mark and the lowest average mark of 10 students. The Chinese mark (CM), the English mark (EM) and the Mathematics mark (MM) of each student are inputted. The highest average mark (HM) and the lowest average mark (LM) should be outputted at the end of the program. The output should be corrected to one decimal place.

High-difficulty problem:

Draw a detailed flowchart and then write a program accordingly which accepts a number (NUM) and determines whether it is a prime number of not. If it is a prime number, a sentence ('IT IS A PRIME NUMBER') should be printed, otherwise, the product of prime factors should be printed.

Appendix C

Scoring sheet of the semantic correctness (Van Merrienboer, 1990)

0 Not attended.
1 The program is hardly recognizable as a "real" program.
2 The program can be recognized as a "real" program but obviously does not try to reach its goal.
3 The program clearly tries to reach its goal but includes both semantical and syntactical errors.
4 The program is semantically correct but includes syntactical errors.
5 The program is semantically as well as syntactically correct.

Authors

AU Sai-kit, SKH Leung Kwai Yee Secondary School, Kwun Tong, Kowloon.
CHUNG Choi-man, Lecturer, Department of Curriculum and Instruction, The Chinese University of Hong Kong, Shatin, New Territories, Hong Kong.