A STUDY ON THE CREATIVE PROBLEM-SOLVING PROCESS IN COMPUTER PROGRAMMING

Ming-Chou Liu and Hsi-Feng Lu

Abstract *-- In this paper we firstly determined how creative problem solving (CPS) is incubated and developed in programming. We adopted methods that included thinking aloud, observation and interviews to compare and analyze the data collected from experts and novices while they were engaged in programming tasks to complete a sorting process using Visual BASIC. The purpose of this study was threefold: 1) to find the CPS ideas generated by expert and novice learners, 2) to find how CPS ideas related to the CPS process, and 3) to find the factors that affected the CPS process. This study found that Figural Form A of the Torrance Test of Creative Thinking (TTCT) score could predict the number of problem solving ideas and programming achievement both in strategy and the degree of program completion. Ideas might fail due to errors in thinking or syntax expression. This study also generalized different error types to help improve programming instruction.*

Index Terms – creative problem solving, programming, thinking aloud

BACKGROUND OF THIS STUDY

Being the foundation of our economic strength, it is significant to give an impetus to research and development in the Information Industry. Keeping software industry performance abreast with the hardware industry is an important goal.

Programming instruction and training are fundamental skills for the development of a software industry. It is an activity involving brainstorming that contains the CPS process, which is somewhat implicit, making it difficult to observe or measure [1]. To formalize and visualize this process and its source better, we can rebuild the relative experience and conditions in programming instruction and training to accelerate the programmer's CPS ability, which in turn will contribute to the performance and output of the software industry, laying a good foundation for our country in the 21st century.

We made a comparison between experts and novices to enhance the learning effect. During the comparison, we discussed the content, style and development of a professional. With this effective instructional method, we helped novices to access professional cultivation with this effective instructional method [2]. This instruction assisted novices with attaining a swift breakthrough in thinking or operation. By analyzing and comparing the differences between the programming experts and the programming novices, determining the specific creative skills of the experts and then handing them down to the novices, a good foundation for software development education and training for industry was laid [3].

Base on the above ideas, we must first determine how CPS is incubated in programming. These research results can be used in future research discussions to obtain essential and important contributions. The follow-up research includes: 1) how to mold relative conditions to apply the ability on problem-solving successfully; 2) how to design the relative instructional training and teaching materials for the cultivation and training of the creative problem-solving ability. This important procedure allows the creative problem-solving ability to come down in one continuous line from incubation, development to application.

Under the planning background mentioned above, this research is aimed at incubating and developing CPS in programming. We adopted methods that included thinking aloud, combining observation and interviews to compare and analyze the data collected from experts and novices. The focuses of this research included:

- Determining the relationship between the quantity of problem-solving ideas and the creativity in the creative problem-solvers (experts and novices)
- Investigate the affective process for the above problem-solving ideas and their influences on successful programming development.
- Investigate the error pattern, proportion and causes of CPS in programming, and then propose an improvement strategy in the instruction.

METHODS AND PROCEDURE

The procedure for this research included seven items as follows:

- Draft the conditions and characteristics of the research objects.
- Choose the research objects (experts and novices).
- Implement the Torrance Test of Creative Thinking.
- Design the questions and interview guidelines.
- Implement the problem-solving programming and collect data.
- Analyze the data.
- Sum up the conclusions and suggestions and write the research report.

We implemented the experiment on thirteen sophomores from the Math Education Department of NHLTC who have just learnt Visual Basic for the first time.

This research aimed at investigating the role of creativity in problem-solving procedure. In this experiment, we adopt Torrance Test of Creative Thinking (Figural Form A) as the measuring standard for creativity. This Torrance Test of Creative Thinking (in short as TTCT) organized by the America scholar E.P. Torrance includes three parts: (1) picture construction (2) incomplete figures (3) repeated figures. Each of the three parts in this test lasted for ten minutes, amounting to 30 minutes in total, plus the instruction and description that totaled 45 minutes.

The creativity involved with the procedure for programming thinking might possess both graphic and literal types. However, the reasons for choosing graphic type, rather than the verbal type in this research was: The completion of problem-solving contained two necessary actions: conceptive problem-solving strategy and the coding and programming of the conceptive problem-solving strategy, and then putting into practice for the verification of its practicability. The former involved with arranging the methods with more graphic thinking than literal thinking is the incubation and development of the CPS that this research concerns.

The three activities of TTCT led to the indexes of the four creativities: the flexibility, fluency, originality and elaboration. To compare the strength of the four abilities, we switched the original scores into standard scores (T score) with the formula as follows:

$$T=50+10x(X-X)/SD$$

After calculation, the T scores for each activity stated as Table I

TABLE I
THE T SCORES FOR EACH ACTIVITY

	Activity 1			Activity 2			Activity 3	
Problem-Solver	Originality	Elaboration	Fluency	Originality	Flexibility	Elaboration	Originality	Elaboration
M	51.5	53.3	57.9	45.1	44.7	45.9	41.7	48.9
H	57.7	55.1	57.9	50.7	50.7	68.1	43.4	59.5
E	57.7	49.7	33.7	39.6	53.2	54.8	66.5	54.2
N	51.5	53.3	57.9	47.0	59.2	45.9	49.4	57.4
I	51.5	44.3	57.9	73.0	48.3	50.3	72.0	57.4
J	57.7	73.1	33.7	54.4	50.7	50.3	52.2	47.9
K	51.5	51.5	39.8	37.7	50.7	41.5	36.2	38.4
O	20.0	35.3	57.9	48.9	48.3	50.3	45.6	42.6
G	45.2	38.9	57.9	45.1	41.1	41.5	42.8	36.3
D	57.7	46.1	39.8	45.1	50.7	45.9	48.3	36.3
L	51.5	60.5	57.9	65.6	67.6	72.5	56.6	67.9
C	45.2	49.7	45.8	43.3	50.7	41.5	48.9	44.7
F	51.5	38.9	51.9	54.4	54.4	41.5	46.1	58.4

With further calculation using this formula, the T scores for each activity (Activity 1, Activity 2 and Activity 3) stated as Table II:

TABLE II
TOTAL T SCORES FOR EACH ACTIVITY

Problem-solver	Activity 1	Activity 2	Activity3
M	53.1	46.1	43.9
H	55.4	54.7	48.2
E	50.5	44.0	62.8
N	53.1	49.1	51.8
I	45.0	64.2	67.6
J	71.5	51.7	50.9
K	51.5	40.3	36.9
O	33.8	49.7	44.7
G	39.6	44.7	40.9
D	47.3	45.7	44.7
L	59.6	66.7	60.0
C	49.3	44.1	47.7
F	40.2	51.7	49.8

The T scores for each ability (originality, elaboration, fluency, flexibility) are shown in Table III:

TABLE III
T SCORE REFERENCE FOR EACH ABILITY

Problem solver	Originality	Elaboration	Fluency	Flexibility
M	46.1	51.1	57.9	45.9
H	50.6	57.3	57.9	68.1
E	54.6	52.0	33.7	54.8
N	49.3	55.3	57.9	45.9
I	65.5	50.8	57.9	50.3
J	54.8	60.5	33.7	50.3
K	41.8	45.0	39.8	41.5
O	38.1	39.0	57.9	50.3
G	44.4	37.6	57.9	41.5
D	50.4	41.2	39.8	45.9
L	57.9	64.2	57.9	72.5
C	45.8	47.2	45.8	41.5
F	50.7	48.7	51.9	41.5

Osborn and Parnes proposed a CPS model that can be referred to the CPS process for computer programming. This model stated five stages of CPS process including fact

finding, problem finding, idea finding, solution finding, and acceptance finding [4]. Every stage repeated a cycle from emphasizing divergent thinking at the beginning to emphasizing convergent thinking at the end. In this study, we provided a pre-defined question so that the learners could start the CPS process from the 3rd stage, idea finding, within the limited time. Divergent thinking emphasizes originality, fluency, and flexibility. Convergent thinking emphasizes originality and elaboration in creative thinking. The idea finding stage emphasizes divergent thinking including originality, fluency, and flexibility. The solution finding stage emphasizes elaboration. The acceptance finding stage requires planning and implementation. This means designing, testing, debugging and interface usage in programming. In Table II and Table III we can see potential problem solvers such as L, I, and J have higher scores and ability.

For actual problem-solving we adopted thinking aloud, in which the students solved the problems with Visual Basic while expressing their ideas and methods out loud. We recorded the whole process with a video recorder for analysis. The time allotted for problem-solving was 30 minutes and the theme for the research was: list the 100 values from 0-100 generated randomly from small to large and display the results. Thinking aloud method has been proven to be an effective way to explicate implicit thinking [5]. After the problem-solving, we implemented a focus interview and then re-confirmed the obtained development data to determine the background of the problem-solvers.

For problem-solving, the sorting problem we chose was very challenging for beginners. This problem contained the following basic programming skills: (1) object-oriented programming; (2) application of variables and arrays; (3) output and input; (4) If statement; (5) the adequacy of the loop (Nested loop were commonly used).

RESULTS AND DISCUSSION

With each problem-solver's originality score in Activity 3 gained by the TTCT, we juxtaposed the quantity each problem-solvers' ideas with the problem-solvers' "Idea Quantity Index" from their realistic problem-solving performance for the convenience of comparison. "Idea Quantity Index" on the left side indicated whether all of the ideas occurred during problem-solving could be applied in the calculation. The originality scores for activity 3 on the right side stated as Table IV shows:

TABLE IV
IDEA NUMBER INDEX DURING PROBLEM-SOLVING AND PREDICTION TABLE

Problem-solver	Idea Quantity Index	Problem-solver	Creativity
E	3	I	72.0
I	3	E	66.5
D	2	L	56.6
L	2	J	52.2
C	1	N	49.4
F	1	C	48.9
G	1	D	48.3
H	1	F	46.1
J	1	O	45.6
K	1	H	43.4
M	1	G	42.8
N	1	M	41.7
O	1	K	36.2

After the detailed comparison we found significant differences appeared in the first four people: Problem-solver E, Problem-solver I, Problem-solver D and Problem-solver L produced more than two ideas. In comparison, the originality scores of the four people in the TTCT are quite identical. Although only problem-solver D got a lower originality score, the score was still at the middle upper level, within the acceptable range. As a result, the "quantity of ideas" of the programming problem-solving predicted by TTCT is very precise. As for the problem-solver C and the problem-solver F who used only one idea, we found no relation between the problem-solving and the creativity in them.

Torrance's research indicated that activity 2 had low fluency and activity 3 had high elaboration and that people with a higher originality had a tendency for successful problem-solving. Since the originality was the average of the three T scores, we predicted the "problem-solving completion" with the three T scores above and got the results as Table V shows:

TABLE V
THE THREE T SCORES AND THE PREDICTION OF THE COMPLETION

Problem solver	Fluency (Activity 2)	Elaboration (Activity 3)	Originality	Prediction of Completion
M	57.9	48.9	47.7	42.61
H	57.9	59.5	52.8	47.79
E	33.7	54.2	52.5	48.97
N	57.9	57.4	51.4	46.53
I	57.9	57.4	59.0	51.09
J	33.7	47.9	58.1	51.07
K	39.8	38.4	42.9	39.44
O	57.9	42.6	42.7	38.35
G	57.9	36.3	41.7	36.49
D	39.8	36.3	45.9	40.82
L	57.9	67.9	62.1	55.05
C	45.8	44.7	47.0	42.56
F	51.9	58.4	47.2	44.81

The calculating formula showed as:

((100-A1)+2*A2+6*A3)/10

We juxtaposed the rank for each problem-solver predicted by the TTCT with their performance during actual problem-solving. Data is displayed in Table VI. On the left

side is the rank for "Program Completion Index" with the "Prediction of the Problem-solving Completion" on the right side. The "Program Completion Index" scoring method is: 10 points for successful problem-solving, including (1) the problem-solving strategy: at least one practicable strategy (3 points); (2) components required: forms, buttons (program execution control components, 1 point) and menu box (data display components, 1 point). Components with the same functions will do. Composing the processing procedure of the events (1 point). (Totally 3 points); (3) Sorting programming: variable access, such as the assignment of the variables or array variables (=), and comparison (1 point), statement (If or Select) (1 point), loop (For, While, Do loop) (1 point), and the combination into a program able to carry out the sorting function (1 point) (Totally 4 points).

TABLE VI

COMPARISON BETWEEN THE "PROGRAM COMPLETION INDEX" AND THE COMPLETION PREDICTION IN PROBLEM-SOLVING DEVELOPMENT

Problem solver	Program Completion Index	Prediction of Completion	Problem solver
C	10	55.05	L
F	10	51.09	I
H	9	51.07	J
I	9	48.97	E
D	8	47.79	H
L	8	46.53	N
E	7	44.81	F
M	7	42.61	M
N	7	42.56	C
G	4	40.82	D
K	4	39.44	K
O	4	38.35	O
J	3	36.49	G

During the first comparison, we found some non-conformity between the problem-solving completion of each problem-solver predicted by the TTCT and their performance in realistic problem-solving. Problem-solver C and Problem-solver F were particularly different. Their ranks in the prediction of TTCT were at the medium level; however, they completed the problem-solving. For further understanding to the background of the two problem solvers, we found that problem-solver C who had always been interested in programming had already endeavored to this field since very young and had abundant experience in programming. Therefore, Problem-solver C had no difficulty in problem-solving. Problem-solver F who also had a craving for programming had taken C programming courses and learned the sorting. We classified these two people as experts with the other problem-solvers as novices. Interesting enough, after classifying Problem-solver C and Problem-solver F as experts and eliminating them from the problem-solvers, the conformity among each problem solver's "problem-solving completion" predicted by TTCT and their performance in realistic problem-solving was drawn closer.

After taking out the two experts, we found that, of the remaining nine novices, only Problem-solver D and Problem-solver J appeared unmatched. Problem-solver D with a lower predicted rank showed good performance, yet Problem-solver J with a higher predicted rank showed bad performance. The performance of the three problem-solvers in the middle of the second section completely conformed to the prediction, while the performance of the last four problem-solvers in the third section acted similarly as the first section. In conclusion, after taking out the experts, TTCT still provided a certain degree of accuracy that makes it referable.

The statistics of quantification responded to the above analysis. The correlation coefficients of the programming completion index and the completion prediction of the 13 problem-solvers in Table 6 were ($r = 0.33, p = 0.27$). Under the condition without Problem-solver C and Problem-solver F, the correlation coefficients raised as ($r = 0.47$, $p = 0.14$. Under the condition without Problem-solver D and Problem-solver J, the correlation gets even higher ($r = 0.87, p = 0.002$).

Since the knowledge involved in programming is quite complicated, it takes each of the following indispensable abilities to complete problem-solving: (1) practicable ideas; (2) application of the V.B. integrated environment; (3) the ability in V.B. programming and (4) testing and debugging ability.

Therefore, taking it from another angle, although both of Problem-solver C and Problem-solver F were experts who performed commonly in TTCT, they still solved the problems successfully. This was because the keys to problem-solving consisted of one practicable method and abundant professional knowledge. The two examples from Problem-solver C and Problem-solver F indicated that the problem-solving ability of programming was drillable; however, due to the huge and complicated professional knowledge, it took more practices and longer time to master.

During the further case analysis, we made an analysis on the representative problem-solvers: Problem-solver E, Problem-solver I and Problem-solver L who got high grades in originality with an outstanding performance within the first four ranks in the aspects of ideas and the completion during realistic problem-solving. We tried to understand the development of their ideas during the problem-solving and their effects.

Obviously, idea finding plays an important role in the CPS process. Without ideas no successful problem solving can be accomplished. Solution finding involves evaluating ideas to find the most appropriate idea as the basis of successful problem solving. Acceptance finding is to plan

and implement the most appropriate idea to check if it is valid. The process might go back to the previous stages if the result of this stage shows invalid.

At the idea finding stage, the problem solver must apply his/her divergent thinking to generate various ideas. More ideas would be better. The learners E, I, and L all generated more than two ideas within a short time and their ideas were proven valid afterwards. This means that these three problem solvers have good potential.

The analysis of the three problem-solvers indicated a more direct problem-solving method for beginners but lack of overall and detailed consideration. When they encountered results that conflicted with their expectation, they became at a loss, doubting the practicability of the original strategy, failing to analyze the result and finally gave up original practicable ideas. Secondly, the processing array data skill was also a key factor. Without full understanding of processing array data and realistic operating experience, beginners could not pass the test the first time that they processed such problems. In the end, after spending most of their time in the syntax and operation, they would run out of time.

The problem-solving result in experiment was: two of the 13 problem-solvers succeeded in solving the problem while 11 problem-solvers failed, resulting in a problem-solving success proportion of 15.4%. The two problem-solvers that completed the problem successfully spent little time thinking about the programming process because they were experts. They were also quite familiar with the scripts and made few mistakes. Even when they did make mistakes, they found them and corrected them right away. In the long run, the two problem-solvers spent little time (less then 10 minutes) completing the problem-solving. They had confidence in sorting. The only problem remained was the selection of the applicative components.

Through the classification statistics, the mistakes or difficulties the problem-solvers encountered during problem-solving are stated as Table VII shows:

TABLE VII

MISTAKES CLASSIFICATION DURING PROBLEM-SOLVING

Classification	Description	Quantity	Total	Percentage
Assignment	1. Incorrect usage of the variable assignment	3	6	15%
	2. Failed to swap the two variables correctly	3		
Data type	3. Type mismatched during data calculation	3	4	10%
	4. Overflow	1		
Array	5. Access to array	5	9	22.5%
	6. Index out of range	4		
Variables effective scope	7. Incorrect variable scope	1	1	2.5%
Syntax	8. Incorrect spelling of the variables or script	5	5	12.5%
If judgment	9. Incorrect script structure	2	5	12.5%
	10. Incorrect comparison and logic calculation	3		
Loop	11. Incorrect script structure of For loop	3	9	22.5%
	12. Incorrect script structure of While loop	1		
	13. Cooperative usage of For loop and array	1		
	14. The control variable in For loop	2		
	15. Usage of the two-layer loop	2		
Component	16. Incorrect attribute configuration of the components	1	1	2.5%

The assignment value, data type, array and variable scope all fell within the variables with an error rate up to 50%. Understanding and using the concept of variables is the most difficult part for beginners. However, the application of variables is the pith in programming, which is a basic ability learners must be equipped with. We suggest that more efforts be invested in the variable section during programming instruction, ensuring that learners completely understand this concept and be able to use the concept with skill. The emphasis lies in: (1) clearly distinguishing the differences between programming variables and mathematical variables; (2) statements, such as the syntax of the comparison in an If statement; (3) different applications, applied time and applied skills for different kinds of variables and (4) skill with complicated data structures, such as the array. Students taking the test tended to accept the object concept and put it into application with the components. A quite low error rate was found, which indicated the advantages of the object-oriented programming.

The application of arrays and loop operations are both advanced skills in programming and contribute greatly to

the solving of difficult problems. The experimental results showed that students had difficulty with these two areas with an error rate of 45% (a total rate of the two ability). Students usually failed to clarify the relationship and produce correct programs even after cudgeling their brains.

Secondly, we found that students had a direct association and application for single loops. However, the correct answers of the single loops did not come out until the application of the nested loops. In the end, the students, equipped with correct initiative concept (such as compare two by two or find out the smallest value), usually managed to use one loop for the programming initially, but failed in completing the sorting. Students gave up this method and thought of other methods or gave up solving this problem because of its complexity. In other words, most students restricted themselves to solving the problem with single loops and never thought of solving it with nested loops. Furthermore, most students did not carefully analyze the result gained from the first loop, which was a great pity as this result connoted significant meaning. By repeating the action for the first loop, the second smallest value occurred, which derived the second loop and the answer. The few students that thought of the solution with nested loops failed to clarify the relationship between the nested loops and the controlled variables or between the controlled variables and the array access. They therefore still failed to solve the problem successfully. During instruction more practical examples should be presented for the learners to have more opportunity to practice and experience the variations.

Conclusions and Suggestions

This research produced the following conclusions:

- The creativity index gained from TTCT matched the tested student's problem-solving performance in the programming development, regardless of the problem-solving strategy or the problem-solving completion.
- Although the quantity of the programming novice ideas predicted by the TTCT was quite precise, the novices were not familiar with the calculations and syntax of the ideas and still failed to solve the problems.
- Experts that already knew about the problem-solving strategy and skills produced fewer ideas, sometimes even just one idea.
- The programming job requires lots of creativity. However, due to the requirement for more professional knowledge, creativity alone was not enough. The syntax of the program language, understanding the semantics, familiarity with each basic skill of programming and the testing and debugging skills were all requisites and must be emphasized in programming instruction.
- In the syntax and meaning of the program language, understanding and application of the variables was the basic and core ability. During instruction, it is necessary to distinguish the variables from the mathematical variables carefully. Array and loop variables were another problem area, a critical and powerful tool in solving complicated problems. Nevertheless, the numerous variations in application required more detailed description and practices.

ACKNOWLEDGMENT

This study was supported by Taiwan National Science Council under contract number NSC89-2519-S-026-001

REFERENCES

[1] Kirner, K., *Implicit and explicit mental processes*, 1998, Mahwah, NJ: Lawrence Erlbaum.

[2] Johanson, R. P., "Computer, cognition and curriculum: Retrospect and prospect", *Journal of Educational Computing Research*, Vol 4, No 1, 1988, 1-29.

[3] Mayer, R. E., "Introduction to research on teaching and learning computer programming", In R. E. Mayer(ed.). *Teaching and Learning Computer Programming: Multiple Research Perspective*. 1988, 1-12. Hillsdale, NJ: Lawrence Erlbaum.

[4] Mayer, R. E., *Thinking, Problem Solving, Cognition.* (2nd ed.). 1991, NY: Freeman.

[5] van Someren, M. W., Barnard, Y. F., and Sandberg, J. A. C., *The Thinking Aloud Method: A Practical Guide to Modeling Cognitive Process*. 1994, NY: Academic Press.