Proceeding of the 5th International Conference on Education, Vol. 5, 2019, pp. 67-74

ISSN 2424 - 6700 online
DOI: https://doi.org/10.17501/24246700.2019.5108

Exploring The Relationship Between Self-Regulated Learning Strategies And Computer Programming Achievement In Higher Education

Gary Cheng[1]*, Leonard K. M. Poon[1], Wilfred W. F. Lau[2], Rachel C. Zhou[1]

[1]*The Education University of Hong Kong, Hong Kong, SAR*

[2]*The Chinese University of Hong Kong, Hong Kong, SAR*

Abstract: This paper aims to report and discuss the findings of a research project exploring the relationship between self-regulated learning (SRL) strategies and computer programming achievement in higher education. In the project, data were collected from 66 undergraduate students who enrolled in a 13-week introductory computer programming course offered by a Hong Kong university during the academic year 2018/19. Participants were asked to complete the Motivated Strategies for Learning Questionnaire (MSLQ) to measure their use of SRL strategies. Their computer programming achievement was assessed by continuous exercises and the end-of-course examination as part of the course assessment. A quantitative correlational analysis on the questionnaire and assessment scores was conducted to explore the relationship between use of SRL strategies and computer programming achievement. The results of the project indicate that higher-order cognitive strategies, metacognitive control strategies, time and study environment strategies as well as help-seeking strategies were positively associated with computer programming achievement. The findings suggest that students could be trained to use certain SRL strategies in order to effectively participate in computer programming activities.

Keywords: self-regulated learning strategies; computer programming; learning achievement

Introduction

To fill the gap in the literature, the purpose of this paper is to evaluate the relationship between students' use of learning strategies and their computer programming achievement in higher education. In particular, this paper focuses on self-regulated learning (SRL) strategies which can be described as self-initiated and goal-oriented actions taken by a student to control, regulate and improve his or her learning. Since a considerable body of research has reported positive associations between academic performance and use of appropriate SRL strategies in a variety of disciplinary areas but not specifically computer programming (Akyol, Sungur & Tekkaya, 2010; Cheng & Chau, 2013; Trevors, Feyzi-Behnagh, Azevedo & Bouchet, 2016), it would be well worth conducting research into this understudied area.

Computer programming can be regarded as a process of creating a sequence of step-by-step instructions executed by a computer to accomplish tasks and meet goals. Papert (1993) argued that computer programming can engage students in decomposing a problem into sub-problems, planning and executing a solution, identifying errors and debugging. Aho (2012) added that computer programming involves the thought processes in formulating problems so that the problems can be solved by computational steps and algorithms. Computer programming has been considered as closely related to 21st century skills that are particularly important in this highly competitive world (Grover & Pea, 2013; Lye & Koh, 2014). For this reason, there has been a growing recognition of the need to equip students with the ability to design and develop computer programs (Resnick et al., 2009).

Nevertheless, computer programming has often been cited as one of the most difficult and challenging areas in computing education (Mcgettrick et al., 2005). Nearly 33% failure rates on introductory programming courses at university level have been reported, indicating that almost one in every three university students failed in

Corresponding Author's Email: * chengks@eduhk.hk

computer programming (Bennedsen & Caspersen, 2007; Watson & Li, 2014). Some research efforts have advanced the knowledge base about predictors of computer programming performance (Watson & Li, 2014). However, there has been very limited research into students' use of learning strategies for computer programming. Moreover, little is known about which strategies are more effective for students to acquire knowledge and skills in computer programming.

Research Questions

To improve understanding of the association between SRL strategies and computer programming in higher education, the present study aimed (1) to identify students' use of SRL strategies and (2) to examine the relationship between students' use of SRL strategies and their achievement in computer programming. The following key research question guided this study: which SRL strategies correlate positively with computer programming achievement?

This research question hypothesized that students' learning of computer programming would benefit more from some SRL strategies than others. One possible way to approach the question is to explore and compare the differences in usage patterns of SRL strategies between high- and low-achieving students in computer programming. The approach was applied to this study.

Research Method

Participants

A total of 66 first year undergraduate students (38 males and 28 females) at a Hong Kong university gave their written consent to participate in this study. The participants were enrolled on an introductory course in Java programming in the first semester of 2018/19. They all majored in information and communication technology and were in their late teens or early twenties 17 to 22 years (M = 20.1 and SD = 1.86). They were assured that their performance in this study would not affect their academic results at university and could opt for withdrawal if they so wished. At the start of the study, all the participants reported that they had little or no experience of computer programming.

Measures

The Motivated Strategies for Learning Questionnaire (MSLQ) was used to assess students' use of SRL strategies (Pintrich, Smith, Garcia, & McKeechie, 1991). The MSLQ is an 81-item instrument comprising six motivational subscales and nine learning strategies scales. Respondents can rate themselves on each item with a 7-point Likert scale, ranging from "not at all true of me" (1) to "very true of me" (7).

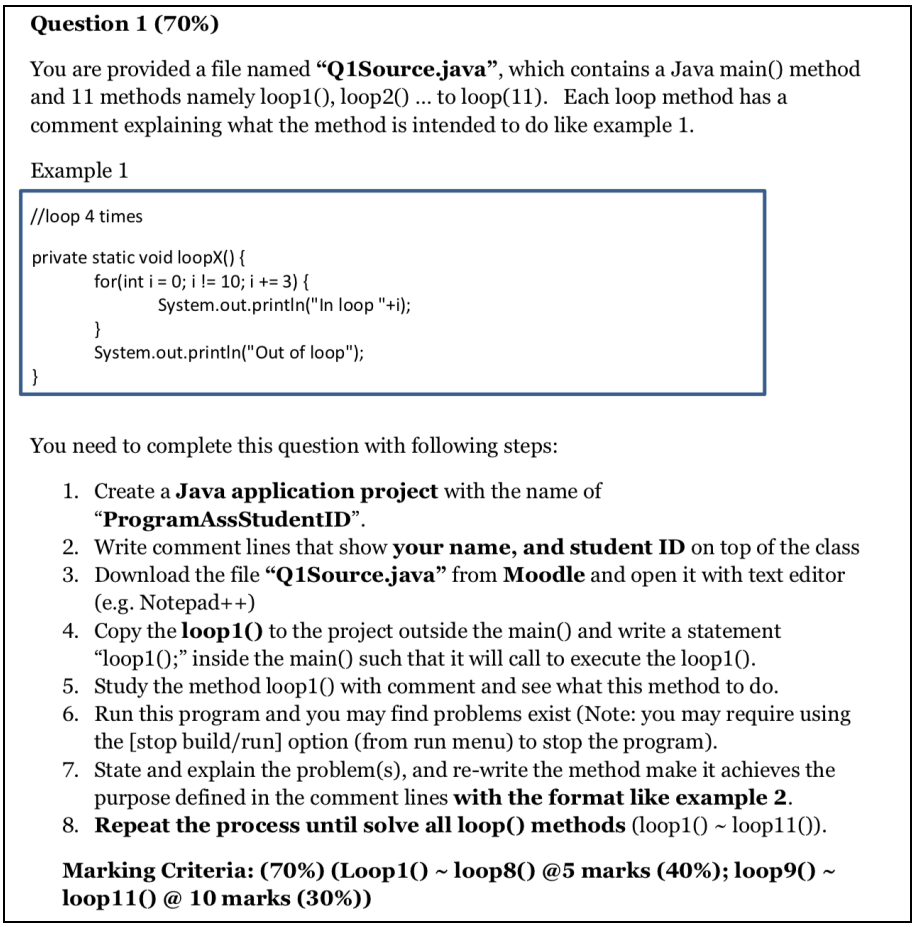
The MSLQ has been widely used to evaluate students' SRL strategies for different research purposes (Chang, 2010; Kilic-Cakmak, 2010; Cheng & Chau, 2013). Since the full MSLQ was too long and not totally relevant to this research purpose, only 50 items from the learning strategies section were administered to the participants. These items were concerned with cognitive strategies (rehearsal, elaboration, organization and critical thinking), metacognitive control strategies, as well as resource management strategies (time and study environment management, effort regulation, peer learning, and help seeking). Table 1 details the subscales and number of items used in this study.

To assess students' achievement in computer programming, two types of individual assessment were taken into account: continuous exercises and the end-of-course examination. Prior to this study, an experienced teacher in

computer programming developed the assessments and rubrics. The teacher was also responsible for marking students' exercises and examination scripts. A sample computer programming exercise is shown in Figure 1.

Table 1 Details of the subscales and number of items used in this study

Scales	Subscales	No. of items
Cognitive strategies	Rehearsal	4
	Elaboration	6
	Organization	4
	Critical thinking	5
Metacognitive control strategies	Metacognitive self-regulation	12
Resource management strategies	Time and study environment management	8
	Effort regulation	4
	Peer learning	3
	Help seeking	4



Question 1 (70%)

You are provided a file named **"Q1Source.java"**, which contains a Java main() method and 11 methods namely loop1(), loop2() ... to loop(11). Each loop method has a comment explaining what the method is intended to do like example 1.

Example 1

```
//loop 4 times

private static void loopX() {
        for(int i = 0; i != 10; i += 3) {
                System.out.println("In loop "+i);
        }
        System.out.println("Out of loop");
}
```

You need to complete this question with following steps:

1. Create a **Java application project** with the name of "**ProgramAssStudentID**".
2. Write comment lines that show **your name, and student ID** on top of the class
3. Download the file **"Q1Source.java"** from **Moodle** and open it with text editor (e.g. Notepad++)
4. Copy the **loop1()** to the project outside the main() and write a statement "loop1();" inside the main() such that it will call to execute the loop1().
5. Study the method loop1() with comment and see what this method to do.
6. Run this program and you may find problems exist (Note: you may require using the [stop build/run] option (from run menu) to stop the program).
7. State and explain the problem(s), and re-write the method make it achieves the purpose defined in the comment lines **with the format like example 2**.
8. **Repeat the process until solve all loop() methods** (loop1() ~ loop11()).

Marking Criteria: (70%) (Loop1() ~ loop8() @5 marks (40%); loop9() ~ loop11() @ 10 marks (30%))

Figure 1. A sample computer programming exercise

Procedure

At the outset of the study, the learning strategies section of MSLQ was administered to participating students to evaluate their use of SRL strategies. All students then started to learn Java programming

language in the introductory programming course and were required to complete course assessments (e.g. continuous exercises and the end-of-course examination). The course teacher marked the assessments and gave each a score to evaluate the programming performance of individual students.

At the end of the course, the participants were divided into two achievement groups according to the median split of their assessment score. Those with a score higher than the median were assigned to the high-achieving group while the remaining to the low-achieving group. The median-split method resulted in 33 participants being classified into the high-achieving group and 33 into the low-achieving group. For continuous exercises, the mean score attained by the high-achieving group was 86.5 out of 100 (SD = 5.1) and that by the low-achieving group was 68.0 (SD = 10.9). For the end-of-course examination, the mean score attained by the high-achieving group was 72.0 out of 100 (SD = 9.3) and that by the low-achieving group was 40.6 (SD = 9.9). The research procedure of this study is illustrated in Figure 2.

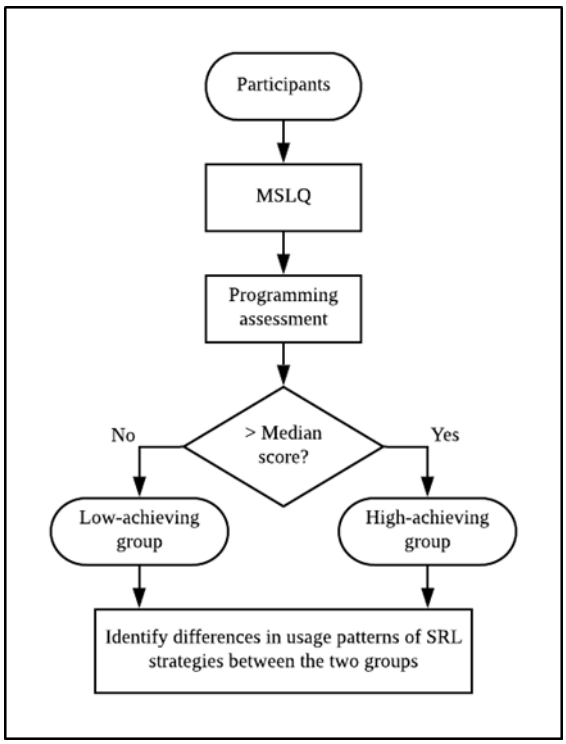


Figure 2. The research procedure

Results and Discussion

Internal Reliability

Internal reliability for each subscale of SRL strategy was measured using Cronbach's alpha (α). As shown in Table 2, the values of Cronbach's alpha ranged from 0.80 to 0.93. They were all well above the acceptance level (0.7), suggesting that all items in each subscale measured the same underlying SRL strategy (Nunnally & Bernstein, 1994).

Continuous Exercises

An independent samples t-test was conducted to identify differences in the use of SRL strategies for continuous exercises between the high- and low-achieving groups (see Table 3). Results in Table 3 show statistically significant differences between the two groups in terms of the four types of cognitive strategies (i.e. rehearsal, elaboration, organization and critical thinking), but indicate no statistically significant differences in terms of metacognitive control strategies and resource management strategies. The results suggest that students in different groups used cognitive strategies at different levels for continuous exercises. The high-achieving group tended to make more frequent use of cognitive strategies than its counterpart.

Table 2 Internal consistency reliability

SRL strategies	No. of items	Cronbach's alpha (α)
Cognitive strategies		
Rehearsal	4	0.82
Elaboration	6	0.93
Organization	4	0.81
Critical thinking	5	0.91
Metacognitive control strategies		
Metacognitive self-regulation	12	0.92
Resource management strategies		
Time and study environment management	8	0.85
Effort regulation	4	0.89
Peer learning	3	0.89
Help seeking	4	0.80

Table 3 Means, standard deviations and t-test values for continuous exercises

SRL strategies	Low-achieving group (N = 33)		High-achieving group (N = 33)		*t*-value
	M	***SD***	***M***	***SD***	
Cognitive strategies					
Rehearsal	4.25	0.83	4.88	0.88	*-2.98***
Elaboration	4.41	0.78	5.52	0.70	*-6.08****
Organization	4.42	0.70	5.23	0.78	*-4.42****
Critical thinking	4.65	0.84	5.30	0.81	*-3.20***
Metacognitive control strategies					
Metacognitive self-regulation	4.63	0.80	4.82	0.77	-1.00
Resource management strategies					
Time and study environment management	4.80	0.86	4.92	0.60	-0.65
Effort regulation	4.33	0.65	4.46	0.94	-0.65
Peer learning	4.25	0.88	4.33	1.05	-0.34
Help seeking	4.86	0.82	4.93	0.78	-0.38

*** $p < 0.01$, **** $p < 0.001$*

From the literature (Pintrich et al., 1991), it is noted that rehearsal strategies like reviewing class notes and examples can help students to familiarize themselves with the subject knowledge. Elaboration strategies like note-taking and summarizing can help students to make connections between various ideas of central topics.

Organization strategies like outlining and grouping can help students to organize their knowledge to develop integrated understanding, while critical thinking strategies can help students to apply their knowledge to new situations. All these types of cognitive strategies could potentially lead to enhanced conceptual understanding of computer programming and higher achievement on programming exercises. As a result, the extent to which the cognitive strategies were used could differentiate the high- and low-achieving students in continuous exercises.

End-of-course Examination

An independent samples t-test was used to evaluate differences in the use of SRL strategies for the end-of-course examination between the high- and low-achieving groups. Table 4 shows the statistical results. As can be seen from the table, there were statistically significant differences in higher-order cognitive strategies (i.e. elaboration, organization and critical thinking), metacognitive control strategies (i.e. metacognitive self-regulation) and resource management strategies (i.e. time and study environment management and help seeking). The results indicate that the high-achieving group tended to make more frequent use of higher-order cognitive strategies, metacognitive control strategies and resource management strategies than its counterpart for the examination.

Table 4 Means, standard deviations and t-test values for the end-of-course examination

SRL strategies	**Low-achieving group (*N* = 33)**		**High-achieving group (*N* = 33)**		***t*-value**
	M	***SD***	***M***	***SD***	
Cognitive strategies					
Rehearsal	4.44	0.82	4.69	0.98	-1.12
Elaboration	4.53	0.82	5.41	0.80	*-4.42****
Organization	4.39	0.66	5.26	0.78	*-4.87****
Critical thinking	4.42	0.69	5.53	0.68	*-6.57****
Metacognitive control strategies					
Metacognitive self-regulation	4.31	0.53	5.14	0.78	*-5.11****
Resource management strategies					
Time and study environment management	4.46	0.71	5.26	0.53	*-5.19****
Effort regulation	4.36	0.67	4.44	0.92	-0.42
Peer learning	4.27	0.68	4.31	1.19	-0.18
Help seeking	4.40	0.67	5.30	0.71	*-4.71****

$^{**}p < 0.01$, $^{***}p < 0.001$

From the literature, it is clear that higher-order cognitive strategies like elaboration, organization and critical thinking can help students to construct deep meaning from their own learning experiences. This is in contrast to rudimentary cognitive strategies (e.g. rehearsal) that are often associated with surface learning (Aukrust, 2011). It is also evident that metacognitive self-regulation strategies can help students to control and regulate their own cognition (Zimmerman & Schunk, 2001). With regard to external influences, time and study environment management strategies can help students to choose an appropriate time and place to study and practice subject knowledge and skills, while help-seeking strategies can help students to seek assistance for their problems from peers or teachers when needed (Pintrich et al., 1991). These strategies were adopted more frequently by the high-achieving group than by the low-achieving group, as shown in Table 4. This suggest that students' use of higher-order cognitive strategies, metacognitive control strategies and resource management strategies may be positive predictors of their achievement in computer programming examination at the course end.

Correlational Analysis

Table 5 presents the correlation coefficients between SRL strategies use and computer programming achievement. The table shows that all cognitive strategies (i.e. rehearsal, elaboration, organization and critical thinking) were significantly positively correlated with scores for continuous exercises ($r > 0.38$, $p < 0.01$). It also shows that higher-order cognitive strategies (i.e. elaboration, organization and critical thinking), metacognitive control strategies (i.e. metacognitive self-regulation) and resource management strategies (i.e. time and study environment management and help seeking) were significantly positively correlated with scores for the end-of-course examination. Overall, the results reveal the importance of cognitive strategies use in continuous assessment (e.g. exercises or assignments). However, for summative assessment (e.g. the end-of-course examination), the results highlight that students who could adopt higher-order learning strategies, regulate their own learning and manage external learning resources more frequently would attain higher scores.

Table 5 Correlations between SRL strategies use and computer programming achievement

SRL strategies	Continuous exercises	End-of course examination
Cognitive strategies		
Rehearsal	*0.38***	0.07
Elaboration	*0.59***	*0.50***
Organization	*0.53***	*0.53***
Critical thinking	*0.41***	*0.62***
Metacognitive control strategies		
Metacognitive self-regulation	0.24	*0.61***
Resource management strategies		
Time and study environment management	0.11	0.*44***
Effort regulation	0.17	0.10
Peer learning	0.02	-0.03
Help seeking	0.22	0.*50***

*** $p < 0.01$*

Conclusion and Future Work

This study aimed to explore the relationship between students' use of SRL strategies and their computer programming achievement. The learning strategies section of the Motivated Strategies for Learning Questionnaire (MSLQ) was used to measure students' use of SRL strategies, while computer programming achievement was evaluated by continuous exercises and the end-of-course examination. The results of this study show that cognitive strategies were significantly positively correlated with the exercise scores of the participants. They also show that higher-order cognitive strategies, metacognitive control strategies and resource management strategies were significantly positively correlated with the examination scores of the participants.

The overall findings suggest that learning computer programming is a challenging task that could not simply be accomplished by using rudimentary cognitive strategies. Students should make more use of higher-order cognitive strategies and other SRL strategies (e.g. metacognitive self-regulation, time and study environment management as well as help seeking) in order to demonstrate higher learning achievement in computer programming. To this end, students could be trained to use appropriate SRL strategies for effective computer programming.

In future research, eye-tracking technology would be used to investigate how students solve computer programming problems and what SRL strategies they use during the problem-solving process. Furthermore, students would be asked to write reflective learning journals on computer programming for analysis of their learning experiences. Both research directions could help triangulate the findings of this study and could further contribute to ways of helping students learn computer programming more effectively.

Acknowledgements

This research is financially supported by General Research Fund (GRF no. 18601118) of the Hong Kong SAR government.

References

Aho, A. V., 2012, Computation and computational thinking. Computer Journal, 55, 832-835.

Akyol, G., Sungur, S., and Tekkaya, G., 2010, The contribution of cognitive and metacognitive strategy use to students' science achievement. Educational Research and Evaluation, 16(1), 1-21.

Aukrust, V. G., 2011, Learning and cognition (Oxford: Elsevier).

Bennedsen, J., and Caspersen, M. E., 2006, Failure rates in introductory programming. SIGCSE Bulletin, 39(2), 32-36.

Chang, M. -M., 2010, Effects of self-monitoring on web-based language learner's performance and motivation. CALICO Journal, 27(2), 298-310.

Cheng, G., and Chau, J., 2013, Exploring the relationship between students' self-regulated learning ability and their ePortfolio achievement. The Internet and Higher Education, 17, 9-15.

Grover, S., and Pea, R., 2013, Computational thinking in K-12: A review of the state of the field. Educational Researcher, 42(1), 38-43.

Kilic-Cakmak, E., 2010, Learning strategies and motivational factors predicting information literacy self-efficacy of e-learners. Australasian Journal of Educational Technology, 26(2), 192-208.

Lye, S. Y., and Koh, J. H. L., 2014, Review on teaching and learning of computational thinking through programming: What is next for K-12? Computers in Human Behavior, 41, 51-61.

Mcgettrick, A., Boyle, R., Ibbet, R., Lloyd, J., Lovegrove, G., and Mander, K., 2005, Grand challenges in computing education - A summary. The Computing Journal, 48(1), 42-48.

Nunnally, J. C., and Bernstein, I. H., 1994, Psychometric theory (3rd ed.) (New York, NY: McGraw-Hill).

Papert, S., 1993, The children's machine: Rethinking school in the age of the computer (New York, NY: Basic Books).

Pintrich, P. R., Smith, D. A. F., Garcia, T., and McKeechie, W. J., 1991, A manual for the use of the motivated strategies for learning questionnaire (MSLQ) (Ann Arbor, MI: National Center for Research to Improve Postsecondary Teaching and Learning, University of Michigan).

Resnick, M., Maloney, J., Monroy-Hernandez, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., and Kafai, Y., 2009, Scratch: Programming for all. Communications of the ACM, 52(11), 60-67.

Trevors, G., Feyzi-Behnagh, R., Azevedo, R., and Bouchet, F., 2016, Self-regulated learning processes vary as a function of epistemic beliefs and contexts: Mixed method evidence from eye tracking and concurrent and retrospective reports. Learning and Instruction, 42, 31-46.

Watson, C., and Li, F. W. B., 2014. Failure rates in introductory programming revisited. Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education, Uppsala, Sweden, June, pp. 39-44.

Zimmerman, B., and Schunk, B., 2001, Self-regulated learning and academic achievement: Theoretical perspectives (2nd ed.) (Mahwah, NJ: Erlbaum).