Social and Management Research Journal,
Vol 16, No 1 (2019) 85-98
https://doi.org/10.24191/smrj.v16i1.6085

The Impact of Pair Programming on Students' Logical Thinking: A Case Study on Higher Academic Institution

Mahfudzah Othman, Arifah Fasha Rosmani,
Shukor Sanim Mohd Fauzi, Umi Hanim Mazlan

Faculty of Computer and Mathematical Sciences,
Universiti Teknologi MARA, Perlis Branch, Arau Campus, 02600 Arau, Perlis.

E-mail:
fudzah@uitm.edu.my

Received: 10 February 2018
Accepted: 8 April 2019
Online First: 26 June 2019

ABSTRACT

Pair Programming (PP) is a well-known agile software development technique that has been widely implemented in programming classes. Through PP, students are able to share knowledge and expertise that will contribute to better programming solutions. Nevertheless, how PP can help to improve students' cognitive abilities has yet to be explored. Therefore, this study's aim was to investigate the impacts of implementing Pair Programming (PP) on students' logical thinking. Logical thinking is part of the cognitive ability claimed to be one of the crucial factors that determine the success or failure of novice programmers. To achieve this, 60 students who enrolled in Diploma in Computer Science programme in Universiti Teknologi MARA Perlis Branch, Malaysia, were asked to take the pre-test and post-test of Group Assessment Logical Thinking (GALT) Test in the beginning and at the end of the semester. These students were divided into two main groups; Control and Test in the Test Group, students with low logical ability will be paired with their high logical thinking friends. Meanwhile, in the Control Group, no pair programming or collaborative technique took place. Five programming tasks were assigned to both groups to solve either collaboratively or individually. The results obtained via paired sample t-tests statistical analysis shows significant improvements in students' logical thinking with p-value <0.05 in the Test Group.

Keywords: *pair programming, logical thinking, introductory programming*

INTRODUCTION

Computer programming generally involves design and engineering activities that are complex, and demands high level of intellectual capabilities (Valentin *et al.*, 2013). Because of the complexity involved, failure and drop-out rates were reported to be high in many academic institutions all over the world (Umi Hanim & Mahfudzah, 2015; Watson & Li, 2014). The high failure rates were more apparent among the first year students because of various reasons such as lack of prior knowledge, interest and motivation (Nurzaid & Zulfikri, 2015).

For many years, academicians have done many researches to investigate the causes that contribute to these high failure rates in introductory programming subjects. Among the factors that have been discovered are the lack of varieties in teaching strategies, differences in learning styles, detachment towards the subject in terms of interests and motivations, and lack of cognitive abilities, a trait crucial for Computer Science students (Kalelioglu & Gulbahar, 2014; Osman & Maghribi, 2015; Wong & Wong, 2016). Prior study suggests cognitive abilities such as critical and analytical thinking skills, problem-solving skills and logical thinking skills are the important traits required to become a successful computer programmer (Iepsen *et al.*, 2013). With these skills, students should be able to analyse the given problems logically and provide the correct solutions (Iepsen *et al.*, 2013). Previous studies have also revealed that deficiency in cognitive abilities among first-year students in Computer Science will lead to problems in comprehending the fundamental notions of programming, hence will lead them to be disengaged with the course or even dropping out from the programme (Iepsen *et al.*, 2013; Umi Hanim & Mahfudzah, 2015). Therefore, cognitive ability is seen as one of the important skills that need to be moulded and mastered in order to determine the success of novice programmers.

Over the years, many efforts have been made to help the students master their programming skills in a bid to improve their performance in this subject. While many were looking to relate the use of technologies such as the e-learning systems or mobile applications as interventions to improve the skills, some were still incorporating traditional way via group collaborations or team pairings in classes. For instance, pair programming, an established agile software development practice widely implemented in programming classes (Nurzaid & Zulfikri, 2015). Nonetheless, there is still lack of studies that measure the effectiveness of pair programming on students' cognitive abilities. Therefore, this study focuses on investigating the impacts of pair programming towards students' cognitive abilities by measuring the changes in their logical thinking levels.

This paper is divided into several sections. The Introduction section discusses the background and motivation for the study. Next section discusses related works, materials and method used to perform the study. This is followed by a section on findings and discussion of the results. A concluding section ends the paper.

RELATED WORKS

Logical and Reasoning Skills in Programming

Bostro and Sandberg (2009) described cognition as the practice, which human beings use to systematise information that involves perception, memory, reasoning and coordination. Cognitive abilities can also be described as the abilities that are used to execute the simplest to more complex cognitive tasks, which require some mental processing (Bostro & Sandberg, 2009). For human beings, our cognitive abilities can be classified into attention, language, visual and spatial processing, memory, interpersonal and intrapersonal skills and logical and reasoning (Bostro & Sandberg, 2009).

In computer science studies, algorithmic thinking, critical and logical reasoning are some of the crucial skills that must be mastered by the students (Muller & Rubinstein, 2011). This is because; these skills will reflect the students' abilities to provide solutions using deductive reasoning through problem-solving strategies and techniques (Singh & Narang, 2014). The lack of logical and reasoning skills among Computer Science students will lead to other problems in their abilities to solve Mathematical calculations, computer programming or any other abstract learning (Singh & Narang, 2014).

One of the measurement tools that can be used to measure logical thinking and reasoning skills is the Group Assessment Logical Thinking (GALT) test. The GALT test developed by Roadrangka, Yeany and Padila comprises of six logical subscales; conservational reasoning, proportional reasoning, controlling variables, probabilistic reasoning, correlational reasoning and combinatorial reasoning (Roadrangka *et al.*, 1983). It has been widely implemented in various areas of teaching and learning and the Cronbach's alpha reliability coefficient of the logical thinking test has also been recorded at 0.52 which is considered moderate to be used in this study (Tuna *et al.*, 2013).

Pair programming can be defined as an agile software

Development technique used by two programmers who are working on the same task and sitting next to each other on one workstation (Beck, 2000). Each person plays important roles described as the 'driver' and 'navigator' and they will work together in designing and coding the same algorithm (Faja, 2013). The role of the 'driver' normally requires him/her to be in charge of the keyboard and mouse, while the 'navigator' monitors the 'driver' and offers suggestions, solutions or corrections to the algorithm or the programmes (Faja, 2013). While in the process of collaborating, designing, coding and reviewing the codes, each member can alternate their roles after certain duration of time (Williams & Kessler, 2002). This technique intends to enhance software productivity at a higher level of software quality (Winkler *et al.,* 2013).

Pair programming has been proven to be efficient in encouraging knowledge sharing and expertise, where the students will be more focused on detailed features when working in pairs (Wray, 2010). Besides that, pair programming has also helped to improve programming practices and students' programming skills (Wray, 2010). Through pair programming, students are more focused, have higher confidence levels when working in teams, and has helped them to develop better teamwork skills (Edwards *et al.*, 2010; Zacharis, 2011). A study also claimed that the students would learn more when working in teams and it has also helped to reduce their frustrations when their individual codes did not work out as expected (Braught, Walls & Eby, 2011).

Prior study has also proved that pair programming has helped to enhance students' learning effectiveness, efficiency and gratification in software engineering course (Akour *et al*., 2013). By implementing pair programming in classes, students could achieve higher assignment grades when working in pair compared to solo programmers and will be able to complete the course with higher passing rates (Lai & Xi, 2011).

Furthermore, empirical evidences have also proved that pair programming practice has helped to improve students' programming abilities, productivity and helped them to produce more quality codes according to Zacharis (2011). Another study found that paired students happened to be more skilled, productive and able to accomplish the task in a shorter amount of time (Salleh *et al*., 2011). Nevertheless, research is still lacking on measures of cognitive enhancements through the implementation of pair programming in the introductory programming classes. While the effectiveness towards students' performance has been proved from time to time, the enhancements of logical and reasoning skills have not yet been discussed widely. Therefore, the main objective of this study is to investigate whether pair programming has significant impacts on students' logical thinking abilities in introductory programming course.

MATERIALS AND METHOD

To achieve the objective of this study, the research method was divided into three main phases as explained below:

Phase 1: Pre-test of Group Assessment Logical Thinking Test (GALT)

The populations of this study are Computer Science students enrolled in the Diploma in Computer Science programme at UiTM Perlis Branch, Malaysia. The sample consists of 60 male and female students enrolled in two randomly selected first year computer programming classes where introductory programming course is taught to heterogeneous classrooms with no grouping or ability tracking. The students were asked to answer a controlled one-hour session of pre-test logical thinking in the beginning of the semester. The GALT test required students to answer twelve questions that measure the six subscales as illustrated in Table 1.

The results of the pre-test of logical thinking were calculated and recorded. Students who scored six marks and above were categorised as high logical thinkers (HLT), while those who scored less than six marks were categorised as low logical thinkers (LLT). These students were later grouped into two main classes, which represented the Control Group and the Test Group.

In the Test Group, the pre-test results were used to pair the students according to their levels of logical thinking where each pair consisted of one HLT and one LLT student. Meanwhile, in the Control Group class, no pairings or teams were created and students participated in the programming task sessions individually.

Table 1: Sub Scales Measurements in Galt Test

Sub scales	Item No.	Item Descriptor
Conservational easoning	1, 2	Piece of clay, metal weigh
Proportional reasoning	3, 4	Glass size, scale
Controlling variables	5, 6	Pendulum length, ball
Probabilistic reasoning	7, 8	Square and diamonds 1, square and diamonds 2
Correlational reasoning	9, 10	The mice, the fish
Combinatorial reasoning	11, 12	The dance, the shopping centre

Phase 2: Programming Task Sessions: Pair Programming vs. Individual

In this phase, both Control Group and Test Group participated in the programming task sessions that were conducted by the lecturers throughout the whole semester. The programming tasks involved five separate one-hour programming sessions. The lecturer who conducted the session in the Test Group were made aware about the principles of the pair programming that requires interchangeable roles of the driver and navigator. In the Control Group, each student work independently on the programming tasks, where discussions among classmates were not encouraged.

Meanwhile, for the construction of the programming questions, there were five structured questions developed by lecturers with more than seven years' experience in teaching introductory programming course. The programming questions were carefully constructed based on topics found in the Fundamentals of Computer Problem-Solving course, and each question was constructed according to Bloom's Taxonomy Cognitive domain as depicted in Table 2.

Table 2: Constructions of the Programming Questions based on Bloom's Taxonomy Cognitive Domain

No	Name of Question	Topic Covered	Cognitive Level
1	Calculate the discount	Sequential Control Structure	C3 - Application
2	Calculate the profit based on the sales	Selection Control Structure	C4 - Analysis
3	Find and display the quotient and/ or remainder	Selection Control Structure	C4 - Analysis
4	Divisible numbers	Repetition Control Structure	C4 - Analysis C5 - Synthesis
5	The tallest students	Functions	C5 - Synthesis C6 - Evaluation

Phase 3: Post-test of Group Assessment Logical Thinking Test (GALT)

At the end of the semester, students in both Control and Test Groups were asked to take the same GALT test again, where the results were recorded as post-test results. Both, pre-test and post-test results were compared and analysed using the paired samples *t*-test analysis to investigate whether there were significant enhancements of students' logical thinking.

RESULTS AND FINDINGS

Pre-test vs. Post-test GALT results

Table 3 shows the overall pre-test and post-test logical thinking results for both the Control Group and Test Group. For the Control Group, the pre-test results show 53.33% of the students have scored less than six marks. This results also indicates that about 16 students were identified as LLT and the other 14 students (46.67%) were HLT students. The mean score for the pre-test Control Group is 5.33, which indicates a low level of logical thinking ability. Table 3 also depicts the results of the post-test for the Control Group where there were only slight improvements on the students' logical thinking abilities with an overall mean score of 5.40, which is still in the low logical thinking zone.

Meanwhile, for the Test Group, based on the pre-test results, it shows that about 50% (15 students) have scored less than six marks in the pre-test logical thinking. None of the students could answer all 12 questions and the overall mean score for the pre-test logical thinking was recorded at 5.50, which also indicates a low level of logical thinking ability. After the pair programming sessions, post-test results have showed significant improvements in students' logical thinking abilities where about 63.33% (19 students) have scored more than six marks and the mean score for the post-test was recorded at 6.50, that represents a higher logical thinking score.

Table 3: Overall Pre-Test vs. Post-Test Logical Thinking Results

No. of questions answered	Control Group		Test Group	
	Pre-test (Frequency)	Post-test (Frequency)	Pre-test (Frequency)	Post-test (Frequency)
1	0		0	0
2	0		1	0
3	6	6	5	0
4	4	3	5	6
5	6	7	4	5
6	8	7	7	5
7	3	4	4	6
8	1	1	1	4
9	1	1	1	1
10	1	1	1	1
11	0	0	1	1
12	0	0	0	1
Total	30	30	30	30

For further investigation, a paired samples *t*-test was also conducted to compare the Test Group's pre-test and post-test logical thinking scores as shown in Table 4. There was a significant difference in the scores for the pre-test (Mean=0.458, Standard Deviation=0.180) and post-test (Mean=0.542, Standard Deviation=0.176) logical thinking levels; with t (29) = 8.523 and p-value < 0.05. These results suggest that the pair programming sessions does have significant impacts on students' logical thinking abilities.

Table 4: Paired Samples *t*-Test Analysis for Individuals' Pre-Test and Post-Test Logical Thinking

Paired Samples Test								
	Paired Differences					*t*	df	Sig. (2-tailed)
	Mean	Std. Dev	Std Mean Err	95% Confidence Interval of the Difference				
				Low.	Upp.			
Pair Post-Pre Mean	0.083	0.054	0.009	0.063	0.103	8.523	29	0.000

CONCLUSION

As a conclusion, this study provides another piece of evidence that Pair Programming (PP) can adopted as effective method to enhance students' logical thinking abilities, particularly among first year Diploma in Computer Science students. However, the success of this study relies on two major factors. First factor was the formation of the pairs that took into consideration the pairings of HLT and LLT or high achievers and low achievers. This is to provide a way for the LLT to discuss and learn more from their HLT partner in solving the programming tasks given. The second factor was the structured programming questions that were carefully designed by experienced lecturers, based on the Bloom's Taxonomy Cognitive domains. Further investigations are required to investigate whether a least controlled environment or minimising the controlling factors can also boost the students' cognitive abilities in programming, such as using the online social networking platform such as the Facebook or Massive Open Online Course (MOOC). Furthermore, it has also been suggested that to ensure the successful implementation of the PP, peer or pair evaluations were also needed. The peer evaluation can become one of the influential factors that ensures the suitability and vulnerability of the pairings. Future in-depth investigation on the students' logical thinking enhancements in introductory programming will also need to include different types of technology-based interventions and assessments such as using gamifications elements and media visualisations.

ACKNOWLEDGEMENT

This work is partially supported by Academic and Research Assimilation Grant, IRMI, UiTM. The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

REFERENCES

Akour, M., Al-Radaideh, K., Alazzam, I. & Alsmadi, I. M. (2013). Effective pair programming practice: Toward improving student learning in software engineering class. *International Journal of Teaching and Case Studies, 4(*4), 336-345. DOI: 10.1504/IJTCS.2013.060635

Beck, K. (2000). *Extreme Programming Explained: Embrace Change*. Upper-Saddle River: Pearson Education, Inc.

Bostro, N. & Sandberg, A. (2009). Cognitive enhancement: Methods, ethics, and regulatory challenges. *Science & Engineering Ethics, 15*(3), 311–341. https://doi.org/10.1007/s11948-009-9142-5

Braught, G., Wahls, T. & Eby, L. M. (2011). The case for pair programming in the computer science classroom. *ACM Transactions on Computing Education, 11*(1), Article 2. Doi: 10.1145/1921607.1921609

Edwards, R. L., Stewart, J. K. & Ferati, M. (2010). Assessing the effectiveness of distributed pair programming for an online informatics curriculum. *ACM Inroads, 1*(1), 48-54. DOI: 10.1145/1721933.1721951

Faja, S. (2013). Evaluating effectiveness of pair programming as a teaching tool in programming courses. *Information Systems Education Journal, 12*(6), 36-45.

Iepsen E. F., Bercht, M. & Reategui, E. (2013, October). Detection and assistance to students who show frustration in learning of algorithms. Paper presented at *2013 IEEE Frontiers in Education Conference (FIE),* Oklahoma, USA.

Kalelioglu, F. & Gulbahar, Y. (2014). The effects of teaching programming via Scratch on problem solving skills: A discussion from learners' perspective. *Informatics in Education, 13*(1), 33–50.

Lai, H. & Xi, W. J. (2011, October). An experimental research of the pair programming in Java programming course. Paper presented at *e-Education, Entertainment and e-Management*, Bali.

Muller O. & Rubinstein A. (2011, August). Work in progress – Courses dedicated to the development of logical and algorithmic thinking. Paper presented at *Frontiers in Education Conference (FIE)*, USA.

Nurzaid M. Z. & Zulfikri P. (2015, August). Pair programming: An overview. Paper presented at *Colloquium in Computer & Mathematical Sciences Education (CCMSE 2015),* Perlis.

Osman, M. N. & Maghribi, M. (2015, August). Capturing best practice in teaching introductory programming course: A case study for non-computer science students. Paper presented at *Colloquium in Computer & Mathematical Sciences Education (CCMSE 2015)*, Perlis.

Roadrangka V, Yeany, R. H. & Padila, M. J., (1983, April). The construction of a group assessment of logical thinking (GALT). Paper presented at *the meeting of the National Association for Research in Science Teaching, Dallas, Texas.*

Salleh, N., Mendes, E. & Grundy, J. (2011). Empirical studies of pair programming for CS/SE teaching in higher education: A systematic literature review. *IEEE Transactions on Software Engineering, 37*(4), 509-525. DOI: 10.1109/TSE.2010.59

Singh, M. & Narang, M. (2014). *Cognitive Enhancement Techniques. IJITKM, 7*(2), 49-6.

Tuna, A., Biber, A. C. & Incikapi, L. (2013). An analysis of mathematics teacher candidates' logical thinking levels: Case of Turkey. *Journal of Educational and Instructional Studies in the World, 3*(1), 2013, pp. 83-91.

Umi Hanim M. & Mahfudzah O. (2015, August). Fundamentals of algorithm design course: Issues, challenges & proposed teaching-learning approaches. Paper presented at *Colloquium in Computer & Mathematical Sciences Education (CCMSE 2015),* Perlis.

Watson, C. & Li, F. W. B. (2014, June). Failure rates in introductory programming revisited. Paper presented at *Proceeding of the 2014 Conference on Innovation & Technology in Computer Science Education (ITiCSE '14), Upsalla, Sweden.*

Williams, L. & Kessler, R. (2002). *Pair Programming Illuminated*. USA: Addision-Wesley Longman Publishing Co., Inc.

Winkler, D., Kitzler, M., Steindl, C. & Biffl, S. (2013). Investigating the impact of experience and solo/pair programming on coding efficiency: Results and experiences from coding contests, In H. Baumeister and B. Weber (Eds.), A*gile Processes in Software Engineering and Extreme Programming, Lecture Notes in Business Information Processing, Vol. 149,* Springer-Verlag Berlin Heidelberg, 106-120.

Wong, S. Y. & Wong, L. W. (2016). An exploratory qualitative and quantitative study into the effectiveness of digital games as a tool to enhance the learning introductory programming. *International Journal of Digital Society (IJDS), 7*(1), 1113-1122.

Wray, S. (2010). How Pair Programming Really Works. *IEEE Software, 27*(1), 50-55.

Valentin, L. F., Pardo, A. & Kloos, C., S. (2013) Addressing drop-out and sustained effort issues with large practical groups using an automated delivery and assessment system. *Computers & Education, 61*, 33-42. https://doi.org/10.1016/j.compedu.2012.09.004

Zacharis, N. Z. (2011). Measuring the effects of virtual pair programming in an introductory programming java course. *IEEE Transactions on Education, 54*(1), 68-170. DOI: 10.1109/TE.2010.2048328