ISSN:
Print - 2277 - 078X
Online - 2315 - 747X

Journal of Humanities, Social Sciences and Creative Arts

DIFFERENTIAL EFFECTIVENESS OF THINK-PAIR-SHARE PROGRAMMING STRATEGIES, CONVENTIONAL METHOD AND LEARNING STYLES ON THE PROGRAMMING ACHIEVEMENTS OF SECONDARY SCHOOL STUDENTS IN IJEBU EDUCATION DIVISION, OGUN STATE

A. O. SAKA, S. Y. ERINOSHO AND A. S. IFAMUYIWA

Department of Science and Technology Education, Faculty of Education, Olabisi Onabanjo University, Ago-Iwoye, Ogun State.
***Corresponding Author:** wale.saka@oouagoiwoye.edu.ng **Tel:** +2348055847112

ABSTRACT

This work examined the effects of the think-pair-share programming strategy on students' achievement in programming. It also determined the moderating effect of learning styles on students' achievement in programming. The study adopted a pretest-posttest-control group quasi-experimental research design. One hundred and twenty-two (122) students offering computer studies in senior secondary 2 from the two purposively selected public senior secondary schools in Ijebu Education Division of Ogun State constituted the sample. Computer Programming Achievement Test (CPAT, r = 0.760) and Learning Style Inventory (LSI, r= 0.83) were used for data collection. Data obtained were analysed through inferential statistics of analysis of covariance (ANCOVA) using IBM SPSS Statistics 23. The finding indicated that the think-pair-share programming strategy significantly improved students' achievement in the programming aspect of computer studies. It was also found that learning style is not a strong factor in the learning of programming. The findings suggest that teachers should adopt the think-pair-share programming strategy in the teaching and learning of computer programming in senior secondary schools. It is therefore recommended that teacher education programme should include the strategy as one of the methods in the computer science methods courses to enable would-be teachers to master its nitty-gritty, since it worked in the senior secondary school. Also, the government and school authority should organize series of training through workshops and conferences to enable the teachers to acquire the skills to use the strategy.

Keywords: Achievement, learning style, pair programming, think-pair-share, programming achievement

DOI:

INTRODUCTION

Programming skills are essential for navigating the digitally connected world. It enables individuals to manipulate successfully many digital devices that abound in schools, workplace, and homes whose operations depend on programs. Programming is the process of writing a set of instructions in the language that the computer understands to enhance human-computer communication. Fagerlund, Hakkinen, Vesisenaho, and Viiri (2021) observed that programming is being

weaved into the education curriculum globally to enhance students' computational thinking and problem-solving ability. Learning to write programs can develop learners' 21st-century skills such as critical thinking, computational, problem-solving, communication, collaboration, and creativity skills (Abesadze & Nozadze, 2020; Ciftci & Bildiren, 2020; Noh & Lee, 2020; Siegle, 2017).

Programming learning is a theme in the curriculum of computer studies in Nigeria from senior secondary (SS) one to three (SS 1 - SS 3). This is to provide opportunities for the students to learn the concepts continuously throughout the senior secondary education for effective utilization of the skills for competitive advantage. However, studies have indicated that students perceived programming to be difficult to learn at this tier of education (Seralidou & Douligeris, 2021; Sklirou, Andreopoulou, Georgaki, & Tselikas, 2020). Cheah (2020) further observed that the learning difficulty in programming is a global issue and more worrisome at the local level.

Furthermore, the reports of West African Examinations Council (WAEC), the West Africa regional examination body's Examiners (2014-2018) in computer studies indicated that students' achievement in the programming component of the subject in senior secondary school is poor. Cheah argued that despite many tools to teach programming, the challenges with effective learning of concept persists. The methods adopted by the teachers to deliver the concept of programming to the students have been criticized by researchers (Figueiredo & Garcia-Penalvo, 2018; Grover & Basu, 2017) for lacking the substance to bring about the desired output in students' learning. The WAEC chief examiners' reports also pointed in the direction of ineffective teaching strategy and lack of computer facilities to teach the concepts. Lack of effective teaching strategies calls for the continuous emergence of methods of teaching that can ease students' learning of programming.

Studies have found that collaborative learning strategies aid students' development of programming skills (Boudia, Benguedach, & Haffaf, 2019; Yalagi, Dixit, & Nirgude, 2020). Kanika and Chakraborty (2020) opined that collaborative programming engenders knowledge sharing which promotes the writing of efficient computer programs. Similarly, researchers have argued that collaborative methods are appropriate for the learning of language and can also be effective for learning programming because both involve language learning (Massoud, Hallman, Plaisent, & Bernard, 2018). An example of a collaborative strategy that has gained popularity in the teaching and learning of language is think-pair-share (TPS). Think-pair-share is a three-stage process where the teacher or facilitator gives time to every learner to think about a problem/task, groups them to discuss and solve the given problems after which they share their solutions with other members of the class. Yunikawati et al. (2021) described TPS as a cooperative learning method that caters for the various needs of different learners. This method was developed by Lyman (1981) at the University of Maryland.

The think-pair-share strategy has been found effective in the teaching and learning of language. Suhrowardi (2020) implemented a think-pair-share technique to teach writing skills of students of MTS AL Islahuddin Kediri, Indonesia. The results showed a significant improvement in both the writing skills of the students and the classroom atmos-

phere. The classroom was more conducive during implementation than before students' exposure to the strategy. The study also found that TPS enhanced students' critical thinking, self-confidence, classroom interaction, and active engagement in learning activities. Similarly, Mu'in, Amelia, Fadilla, and Elyani (2020) investigated the efficacy of think-pair-share on students' performance in English. The research work adopted a quasi-experimental design where the students were randomly grouped into experimental and control groups. The outcome indicated that students exposed to the think-pair-share strategy performed significantly better than those in the control group. The learners exposed to TPS outperformed their counterparts in the control group in reading comprehension. It was concluded that TPS is effective to improve students' language skills. The qualitative research work of Mualizan and Hakim (2018) also found that TPS improved students understanding of the learning materials and gave them confidence to speak fluently in the public. They were also actively engaged, focused and expressed excitement towards learning.

Another form of collaborative strategy that has been used to teach how to write programs is pair programming. Pair programming (PP) involves two learners sitting side-by-side to write computer programs and can be implemented with or without a computer (Sherrif, 2016). One learner serves as the driver while the other serves as a navigator. The driver types at the computer or writes down the program designs while the navigator observes the work of the driver for tactical and strategical defects. The roles are alternated at a regular interval between members of the groups to prevent the dominance of one member over the other. However, members of the pair programming group can be more than two depending on task complexities and class population but should not exceed six (Kafilongo, 2016).

Celepkolu and Boyer (2018) investigated the effect of pair programming on students' achievement in programming through a mixed-method design. The result revealed that pair programming significantly enhanced students' learning of the concept. It also improved their confidence and promoted their positive attitudes towards programming. The study of Papadaski (2018) also reported that pair programming was more effective than solo programming in enhancing and supporting students' learning and understanding of basic programming concepts. It was also revealed that the strategy significantly improved students' attitudes towards programming. The findings from the research works of Misra (2021); Xu, Yan, Gao, Zhang, and Yu (2020) further lend credence to the effectiveness of pair programming on students learning of programming. However, Bowman, Jarratt, Culver, and Segre (2021) reported that pair programming did not have a significant effect on students' achievement in programming and also did not affect their interest, confidence, and attitude towards programming. The conflicting results might be due to the various research designs adopted by the investigators.

It is evident from the foregoing that programming learning is difficult for students at the secondary level of education due to a lack of effective teaching methods and computer facilities among several reasons. Meanwhile, researchers have suggested the use of strategies appropriate for language learning to teach the concept. The think-pair-share strategy and pair programming strategies have been found effective for language and

programming learning respectively. However, Bowman, Jarratt, Culver, and Segre (2020) argued that research findings on the effectiveness of pair programming on students' programming learning are mixed due to inappropriate research designs and other implementation issues. This may account for why students' difficulty with the learning of programming persists. This study, therefore, fuses both think-pair-share and pair programming strategies into one strategy called the think-pair-share programming strategy (TPSPS) to examine its effects on senior secondary school students' achievement in the programming aspect of computer studies. The choice of the TPSPS was premised on the fact that the fused strategies have separately improved language and programming learning respectively. Nevertheless, TPSPS is a variation of the traditional think-think-pair-share (TPS) developed by Lyman in 1981. It has three stages: think, pair and share but the pair stage of TPSPS adopted the principles and guidelines of pair programming which is unlike the traditional TPS which does not involve pair programming.

The learning styles of the students have been identified as one of the factors that affect learning. Kolb and Kolb (2005) defined learning style as the ability of an individual to incline towards a way of learning than others. It is opined that learning as a process is meaningful when students are allowed to process the information received in their preferred ways (Pashler, McDaniel, Rohrer, & Bjork, 2009). From the viewpoint of Maia, Serey, and Figueiredo (2017), learning style determines how students collect, select, interpret, organize and store information. Also, the instructional method found effective for one learning style group may be ineffective for another group. Umar and Hui (2012) maintained that students' performance in programming is enhanced if their learning style is understood early by the teachers.

Studies have reported how learning styles affect students' achievement in programming. Yeboah and Sarpong (2012) examined the influence of learning style on students' academic achievement in programming. The research work found that learning style was a significant factor in students' learning of programming. It was also shown that divergent learners obtained the highest mean scores, followed by assimilators, accommodators and convergers. Cakiroglu (2014) who also examined the relationship among learning style, study habits and achievement in online programming learning found that learning style significantly influenced students' achievement in programming. The divergers and accommodators performed better than the assimilators and convergers. The discrepancy in the findings might be due to the difference in the learning environments. Similarly, Seyal, Mey, Matusin, Norzainah, and Abdul-Rahman (2015) examined the influence of learning style on the performance of first-year undergraduates in a programming course. The study which employed Kolb's learning style inventory reported a significant impact of students' learning style on academic achievement in programming. It was also reported that converging and assimilating learners significantly outperformed the accommodator and divergers. The discrepancy in the findings concerning which learning style subgroup obtained the highest score may be due to the environment where the learning took place. In contrast, the work of Campbell and Johnstone (2010) revealed that learning style did not significantly impact students' achievement in programming.

The preceding findings imply that the determination of the interaction between the teaching strategy and students' learning style is essential for students' purposeful learning. Hence, this research also investigated the moderating effect of learning style on students' learning of programming after exposure to the TPSPS.

There are many classifications of learning style models in the literature. Examples are the Kolb model, Honey and Mumford model, Gregorc Learning model, Herma Brain Dominance, 4mat learning model, and Felder-Silverman learning style model. Meanwhile, the study adopted the Kolb (1984) learning model due to its popularity and availability of adaptable instrument. The instrument was used to classify the learners into accommodating, assimilating, converging and diverging learning modes.

Objectives of the study

The main objective of the study was to examine the effects of treatment (Think-pair-share-programming strategy and conventional method) on students' academic achievement in programming aspect of senior secondary school computer studies. Specifically, the research examined if:

i. There is no significant difference in the effect of think-pair-share programming strategy and conventional method on senior secondary students' achievement in programming
ii. There is no significant difference in the effect of learning styles (accommodator, assimilator, converger and diverger) on senior secondary students' achievement in programming.
iii. There is no significant interaction effect of think-pair-share programming strategy and learning style on students' achievement in programming.

To achieve the objectives of this study three hypotheses were formulated as follows:

$H_0$1: There is no significant difference in the effect of think-pair-share programming strategy and conventional method on senior secondary students' achievement in programming

$H_0$2: There is no significant difference in the effect of learning style (accommodator, assimilator, converger and diverger) on senior secondary students' achievement in programming.

$H_0$3: There is no significant interaction effect of think-pair-share programming strategy and learning style on students' achievement in programming.

RESEARCH METHODS

The research design adopted was a pretest-posttest control group quasi-experimental design involving a 2X4 factorial. One hundred and twenty (122) senior secondary two (SS 2) students offering computer studies from the two purposively selected public senior secondary schools in the Ijebu Education Block of Ogun State participated in the study. The schools were selected on the criteria that they had qualified computer teachers, offered computer studies up to SS 2 since the subject is elective in senior secondary school, and the schools were far away from each other to prevent interaction among the students of the selected schools.

Computer Programming Achievement Test (CPAT) and Learning Style Inventory were the instruments used for data collection. The CPAT is a researcher-developed instrument with 40 multiple-choice items relating to programming. The 40-item tested students' ability on lower-order programming skills such as remembering and understanding as well as application of knowledge to solve programming tasks as higher-order skill. The 40-item

instrument was obtained after the initial 120 -item test had been subjected to item analysis.

The face and content validities of the instrument were obtained through the critique of two experts each in computer science and test construction. The comments and suggestions of the experts were used to further finetune the instruments. Thereafter, the instrument was administered on thirty (30) SS 2 students from schools that did not participate in the study but share similar attributes to determine its reliability. The scores obtained were analysed using split-half reliability method and it yielded a coefficient of 0.760.

The Learning Style Inventory was adapted from the work of Honey and Mumford (2006). It has 36 items in the form of a 4-point likert format questionnaire with not like me attracting 1, little like me attracting 2, like me attracting 3 and a lot like me attracting 4. The items were structured to reflect the ways individuals learn. The highest aggregate score of each respondent on the items that relate to a particular mode of learning determined the classification into learning style subgroups- accommodator, assimilator, converger and diverger. The face and content validities were ensured through the critique of experts in test construction and psychology in the Faculty of Education, Olabisi Onabanjo University, Ago-Iwoye. The reliability of the instrument was determined by administering the instrument on 20 SS 2 students of a school that did not participate in the study but had similar characteristics with the schools that participated. The ratings of the respondents were subjected to Cronbach coefficient Alpha reliability statistics which yielded a reliability coefficient of 0.83.

This experimental study lasted for six weeks. The first week was to train the three teachers and the students in the experimental group on how to implement the think-pair-share programming strategy. After the training, Computer Programming Achievement Test and Learning Style Inventory were administered on the students in both the control and experimental group as the pretests. From the second week, students in the experimental group were exposed to programming through the think-pair-share programming strategy (TPSPS).

The teacher started each class with a discussion on the selected topic. Thereafter, students were grouped such that each group had a maximum of four members using their scores in the second term examination as a proxy. This was to ensure that students of mixed abilities are grouped (Retnowati, Ayres, & Sweller, 2018). After the group constitution, tasks were given to the students to do in groups. However, each student was given 10 minutes to individually think about the problems at hand before solving the tasks in the group. During group work, the students followed the rule of pair programming. Each student in a group took turn to spend 5 minutes writing the codes and while other members served as the navigators by looking for errors in what the driver was doing. The roles were alternated among members so that no member dominated the others. The solutions to the tasks were then shared by each group with other members of the class. After each class, the teacher gave classwork and assignments to obtain feedback from the learners. Remedial classes were organized for students that did not do well in the exercises. Meanwhile, the students in the control group were taught using the conventional method of teaching which did not involve think-pair-share program-

ming strategy. At the end of the six weeks, reshuffled version of the CPAT was administered to the students as the posttest.

The collected data were analysed through inferential statistics of analysis of covariance (ANCOVA) using IBM SPSS Statistics 23.

Findings

The findings from the data analyses are presented below:

$H_0$1: There is no significant main effect of strategy (think-pair-share programming strategy and conventional method) on senior secondary students' achievement in programming.

Table 1: Summary of analysis of variance of students' achievement by teaching strategy and students' learning style

Source	Type III Sum of Squares	df	Mean Square	F	Sig.	Partial Eta Squared
Corrected Model	466.215	8	58.277	9.115	0.000	0.392
Intercept	1813.477	1	1813.477	283.653	0.000	0.715
Pretest	1.238	1	1.238	0.194	0.661	0.002
Strategy	427.929	1	427.929	66.934	0.000	0.372
Learning Style	27.833	3	9.278	1.451	0.232	0.037
Strategy * Learning Style	2.274	3	0.758	0.119	0.949	0.003
Error	722.441	113	6.393			
Total	42390.000	122				
Corrected Total	1188.656	121				

There is a significant difference in the effect of teaching strategy on the senior secondary school students' achievement in programming ($F(1,113) = 66.934$, $p = 0.000 < 0.05$). Table 1 implies that the posttest mean score of students exposed to think-pair-share programming strategy differs significantly from the posttest mean score of students exposed to the conventional method of teaching. With this outcome, the hypothesis which stipulates that there is no significant difference in the effect of strategy (think-pair-share programming (TPSPS) and conventional method) on senior secondary students' achievement in programming is rejected.

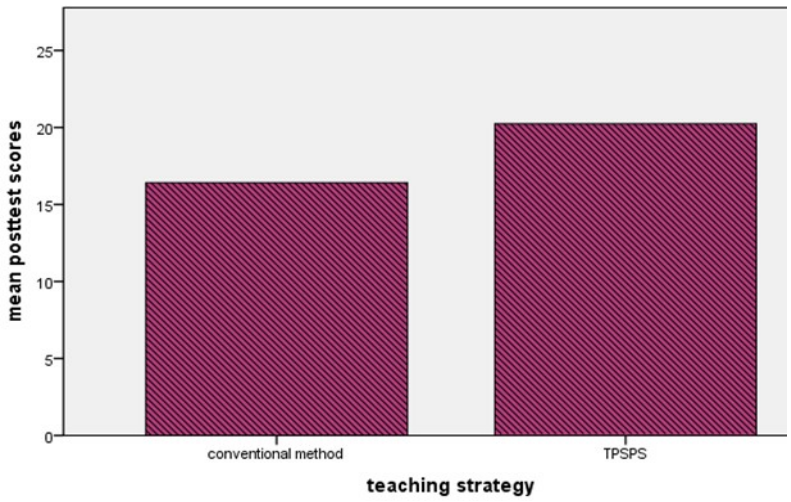

Figure 1: Effects of strategy on students' achievement in programming

After adjusting for the covariates, the think-pair-share programming strategy potently improved students' achievement in programming better than the conventional method of teaching (Figure 1). The difference between the effects of the TPSPS and the conventional method was significant according to the result in Table 1.

$H_0$2: There is no significant difference in the effect of learning style (accommodator, assimilator, converger and diverger) on senior secondary students' achievement in programming.

Evidence from table 1 reveals that there is no significant difference in the effect of learning style (accommodator, assimilator, converger and diverger) on senior secondary students' achievement in programming (F (3, 113) = 1.451, p= 0.232>0.05). This means that the posttest mean achievement scores of students who are accommodator, assimilator, converger and diverger do not differ significantly. Thus, the hypothesis that there is no significant difference in the effect of learning style (accommodator, assimilator, converger and diverger) on senior secondary students' achievement in programming is retained.

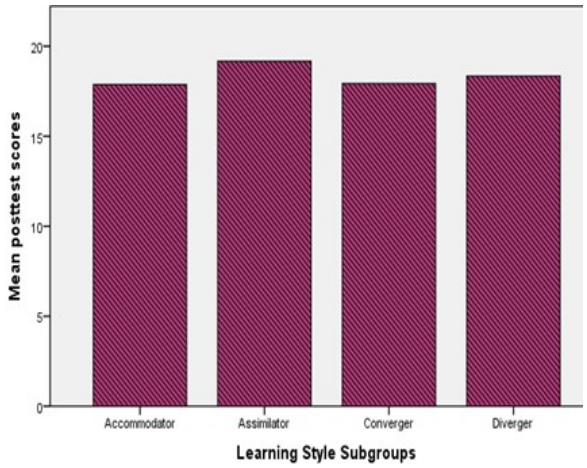

Figure 2: Effects of learning style on students' achievement in programming

After adjusting for the covariates, the assimilating learners obtained the higher posttest mean achievement score in programming whereas the posttest mean achievement scores of accommodating, converging and diverging learners seem to be the same (Figure 2).

$H_0$3: The results in Table 1 show that there is no significant interaction effect of strategy and learning style on students' achievement in programming.

There is no significant interaction effect of instructional strategy and learning style on the students' achievement in programming ($F(3, 113) = 0.119$, $p = 0.949 > 0.05$). This outcome suggests that the posttest mean achievement scores of accommodators, assimilators, convergers and divergers exposed to the conventional method and think-pair-share strategy do not differ significantly. From this outcome, the hypothesis that there is no significant interaction effect of strategy and learning style on students' achievement in programming is retained.

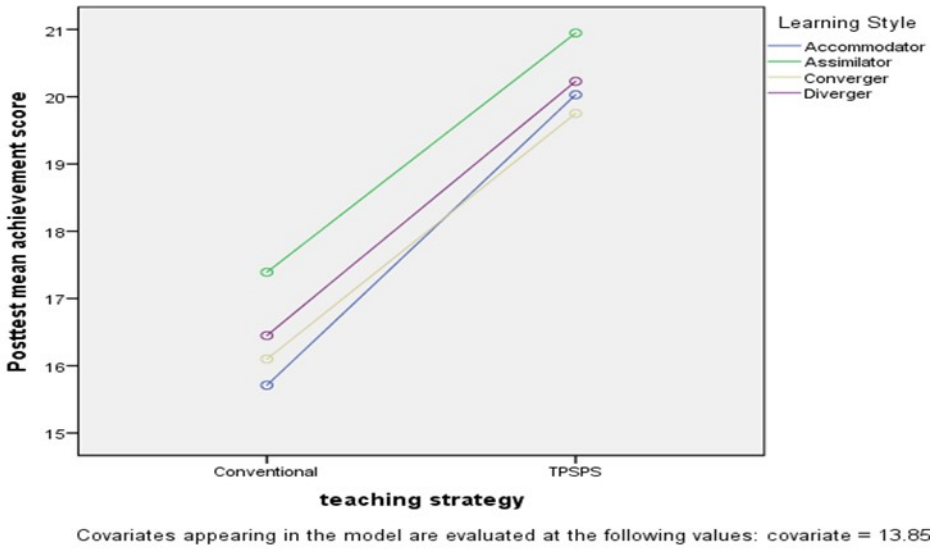

Figure 3: Graph showing interaction effect of strategy and learning style

Figure 3 discloses that the assimilating learners outperformed their counterparts from other learning style mode regardless of teaching strategy. It also reveals that students exposed to the think-pair-share programming performed better than those exposed to the conventional method irrespective of their learning style modes.

DISCUSSION

The research work hypothesized that there is no significant difference in the effect of strategy (think-pair-share programming and conventional method) on senior secondary students' achievement in programming. However, the finding revealed a significant difference in the effect of strategy in favour of the think-pair-share programming strategy (TPSPS). This finding suggests that TPSPS significantly improved students' learning of programming compared to the conventional method of teaching. The outcome supports previous findings on the effectiveness of the traditional think-pair-share strategy for learning programming (Apriyanti & Ayu, 2020; Flora et al., 2020). Also, it aligns with the

reports of the research works of Misra (2021); Xu, Yan, Gao, Zhang, and Yu (2020) that pair programming could significantly enhance students' achievement in programming. However, it conflicts the findings of Bowman, Jarratt, Culver, and Segre (2021) which indicated that pair programming did not have a significant effect on students' achievement in programming.

The effectiveness of TPSPS in this study might be due to the incorporation of pair programming at the 'pair' stage of the strategy. This afforded every learner the opportunities to participate actively in the programming learning activities. The social interaction among the learners might have also assisted the students to learn what they could not have been able to learn individually. Scotts and Palincsar (2013) argued that when learners are grouped to learn, they acquire socially shared experiences and related effects such as improved learning and also gain valuable method of solving problems. The grouping of low ability and high ability learners to exchange ideas during programming learning might have contributed to the recorded learning improvement.

Another focus of the study is to determine the effect of students' learning style on achievement in programming. The result indicates that students' learning style has no significant effect on students' achievement in programming. This may be because the think-pair-share programming meets the learning needs of the students. The finding corroborates the report of Campbell and Johnstone (2010) which revealed that learning style did not significantly impact students' achievement in programming. It is, however, incongruent with the findings of Maia et al. (2017); Saharudin, Yusoff, Haron, and Latif (2018) that learning styles significantly improved students learning in programming. Meanwhile, among the four learning subgroups, the students with an assimilating mode of learning outperformed their counterparts with accommodating, converging and diverging modes although the difference is insignificant. Assimilating learners are logical, organized, reliable, careful and thoughtful and are found majorly in Science and Information Technology-related discipline (Seyal et al., 2015). Learning programming desires thoughtfulness and carefulness in task analysis. Thus, assimilating mode of learning favours the characteristics that programmers should exhibit. From, this finding, learning style does not matter when programming is learned using the TPSPS.

On the interaction effect of strategy and learning style, the reports revealed no significant interaction effect of the strategy and learning style. This report is similar to that of Gabriel, Osuafor, Cornelius, Obinna, and Francis (2018) which revealed no significant interaction effect of strategy (cooperative learning and individualized instruction) on students' achievement in chemistry in senior secondary schools. It also aligns with the finding of Bamiro (2015) that there was no significant interaction effect of think-pair-share strategy and cognitive entry behaviours on students' achievement in chemistry.

However, students exposed to the think-pair -share programming strategy (TPSPS) performed better than their counterparts in the control group across the four modes of learning styles. This means that TPSPS is better at meeting the learning needs of learners from diverse background. It further discloses that TPSPS allowed the learners to interact with the environment without any hindrance during programming instructional deliveries.

CONCLUSION AND RECOMMENDATION

This research work investigated the effect of the think-pair-share programming strategy (TPSPS) on students' achievement in the programming aspect of computer studies in senior secondary school. It is concluded that the strategy significantly improved students' achievement in programming. This finding to teachers implies that they can use the think-pair-share programming strategy to effectively teach the programming aspect of senior secondary school computer studies. This is an aspect of the subject that has been described in the literature to be difficult to teach and learn. Similarly, students can become active producers of technology and have a global competitive advantage after exposure to programming through the TPSPS because the operations of many 21st -century digital devices depend on programming.

Another focus of this study is to examine the moderating effect of learning style on students' achievement in programming. It was established that learning style does not have a significant impact on students' achievement in programming. From this finding, it is concluded that TPSPS can assist all senior secondary school students to learn programming irrespective of their mode of learning. However, the findings on the effectiveness of TPSPS should not be generalized beyond senior secondary school where it has shown the prospect to enhance learners' performance in programming. This is because the sample used for this research work was selected from senior secondary schools.

Consequently, it is recommended that teachers should adopt the use of think-pair-share programming strategy in the teaching and learning of computer programming in senior secondary schools. This is because it enhanced students' programming learning across their learning style subgroups. The government and school authority should organize series of training through workshops and conferences to enable the teachers to acquire the skills to use the strategy.

It is also recommended that the teacher education programme in the universities which is meant to produce teachers for senior secondary schools should include the strategy as one of the methods in the computer science methods courses to enable would-be teachers to master its nitty-gritty, since it worked in the senior secondary school.

The efficacy of the strategy should be tested at other levels of education such as primary, junior secondary schools and tertiary institutions.

REFERENCES

Abesadze, S., Nozadze, D. 2020. Make 21st century education: The importance of teaching programming in schools. *International Journal of Learning and Teaching* 6(3): 6.

Apriyanti, D., Ayu, M. 2020. Think-pair-share: engaging students in speaking activities in classroom. *Journal of English Language Teaching and Learning* 1(1): 13–19. doi: 10.33365/jeltl.v1i1.246

Bamiro, A. O. 2015. Effects of guided discovery and think-pair-share strategies on secondary school students' achievement in chemistry. *SAGE Open* 5(1): 2158244014564754. doi: 10.1177/2158244014564754

Boudia, C., Bengueddach, A., Haffaf, H. (2019). Collaborative Strategy for Teaching

and Learning Object-Oriented Programming course: A Case Study at Mostafa Stambouli Mascara University, Algeria. *Informatica* 43(1). doi: 10.31449/inf.v43i1.2335.

Bowman, N. A., Jarratt, L., Culver, K. C., Segre, A. M. 2020. Pair programming in perspective: Effects on persistence, achievement, and equity in Computer Science. *Journal of Research on Educational Effectiveness* 13(4): 731–758. doi: 10.1080/19345747.2020.1799464.

Bowman, N. A., Jarratt, L., Culver, K., Segre, A. M. 2021. The impact of pair programming on college students' interest, perceptions, and achievement in Computer Science. *ACM Transactions on Computing Education* 21(3): 19:1-19:19. doi: 10.1145/3440759

Cakiroglu, U. 2014. Analyzing the effects of learning styles and study habits of distanced learners on learning performances: A case study of an introductory programming course. *The International Review of Research in Open and Distributed Learning* 15 (4): 161-185.

Campbell, V., Johnstone, M. 2010. The significance of learning style with respect to achievement in first year programming students. 165–170. doi: 10.1109/ASWEC.2010.33

Celepkolu, M., Boyer, K. E. 2018. Thematic Analysis of Students' Reflections on Pair Programming. Paper presented at the 49th ACM Technical Symposium on Computer Science Education. February 21-24, 2018. Baltimore MD, USA. ACM, NY, USA 6(pp.771-776). Retrieved from www.cise.utl.edu.https://doi.org/10.1145/3159.450.3159516.

Cheah, C. S. 2020. Factors contributing to the difficulties in teaching and learning of computer programming: A literature review. *Contemporary Educational Technology* 12(2): ep272. doi: 10.30935/cedtech/8247.

Ciftci, S., Bildiren, A. 2020. The effect of coding courses on the cognitive abilities and problem-solving skills of preschool children. *Computer Science Education* 30(1): 3–21. doi: 10.1080/08993408.2019.1696169.

Fagerlund, J., Häkkinen, P., Vesisenaho, M., Viiri, J. 2021. Computational thinking in programming with Scratch in primary schools: A systematic review. *Computer Applications in Engineering Education* 29(1): 12–28. doi: https://doi.org/10.1002/cae.22255.

Figueiredo, J., García-Peñalvo, F. J. 2018. Building skills in introductory programming. *Proceedings of the Sixth International Conference on Technological Ecosystems for Enhancing Multiculturality* 46–50. New York, NY, USA: Association for Computing Machinery. doi: 10.1145/3284179.3284190.

Flora, F., Raja, P., Mahpul, M. 2020. Discovery learning strategy: integrating think-pair-share and teacher's corrective feedback to enhance students' writing language accuracy. *International Journal of Education and Practice* 8(4): 733–745.

Gabriel, I. A., Osuafor, A. M., Cornelius, N. A., Obinna, P. P., Francis, E. 2018. Improving students' achievement in chemistry through cooperative learning and individualized instruction. *Journal of Education, Society and Behavioural Science* 1–11. doi: 10.9734/JESBS/2018/42873.

Grover, S., Basu, S. 2017. Measuring student learning in introductory block-based

programming: examining misconceptions of loops, variables, and boolean logic. *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* 267–272. New York, NY, USA: Association for Computing Machinery. doi: 10.1145/3017680.3017723.

Honey, P., Mumford, A. 2006. *Kolb's Learning Styles*. Retrieved from http://www.ycarhe.eu/uploads/Document/learning-styles-kolb-questionnaire.pdf

Kafilongo, K. W. M. 2016. The use of pair -programming to enhance the academic performance of tertiary level software development students. Retrieved February 28, 2021, from Undefined website: /paper/The-use-of-pair-programming-to-enhance-the-academic-Kafilongo/6140f3dbffd67308367a80b6e2e0b7d8005c5cb3

Kanika, S. C., Chakraborty, P. 2020. Tools and techniques for teaching computer programming: A review. *Journal of Educational Technology Systems* 49(2): 170–198. doi: 10.1177/0047239520926971.

Kolb, D. A. 1984. Experiential Learning. Englewood Cliffs, NJ: Prentice-Hall.

Kolb, A. Y., Kolb, D. A. 2005. Learning styles and learning spaces: Enhancing experiential learning in higher education. *Academy of Management Learning & Education* 4(2): 193-212.

Lyman, F. 1981. The Responsive Classroom Discussion. In A. S. Anderson (Ed.), Mainstreaming Digest. College Park, MD: University of Maryland Press, pp. 109-113. Retrieved 29 January, 2020 from https://www.scirp.org/(S(vtj3fa45qm1ean45vvffcz55))/reference/ReferencesPapers.aspx?ReferenceID=1955288

Maia, M. C. O., Serey, D., Figueiredo, J. 2017. Learning styles in programming education: A systematic mapping study. *2017 IEEE Frontiers in Education Conference (FIE)* 1 –7. doi: 10.1109/FIE.2017.8190465

Massoud, L., Hallman, S., Plaisent, M., Bernard, P. 2018. Applying and improving multidisciplinary teaching techniques to the programming classroom environment. *DACEE-18,BEHIS-18 May 11-12, 2018 Zagreb (Croatia)*. Presented at the May 11-12, 2018 Zagreb (Croatia). doi: 10.17758/HEAIG3.H0518408.

Mualizan, Z. A., Hakim, M. N. 2018. The implementation of cooperative learning think pair share strategy in teaching reading comprehension at junior high school. *IJLECR - International Journal of Language Education and Culture Review* 4(2): 155–161. doi: 10.21009/IJLECR.052.19.

Misra, S. 2021. Pair programming: An empirical investigation in an agile software development environment. In A. Przybyłek, J. Miler, A. Poth, & A. Riel (Eds.), *Lean and Agile Software Development* (pp. 195–199). Cham: Springer International Publishing. doi: 10.1007/978-3-030-67084-9_13.

Mu'in, F., Amelia, R., Fadilla, R., Elyani, E. P. 2020. Teaching reading on English for specific purposes with think-pair-share technique. *Journal of English Education and Teaching* 4(4): 583–596. doi: 10.33369/jeet.4.4.583-596.

Noh, J., Lee, J. 2020. Effects of robotics

programming on the computational thinking and creativity of elementary school students. *Educational Technology Research and Development* 68(1): 463–484. doi: 10.1007/s11423-019-09708-w

Papadakis, S. 2018. Is Pair Programming more effective than Solo Programming for Secondary Education, Novice Programmers? A case study. *International Journal of Web-based Learning and Teaching Technologies* 13 (1): 1–16. https://doi.org/10.4018/IJWLTT.2018010101/

Pashler, H., McDaniel, M., Rohrer, D., Bjork, R. 2009. Learning styles: Concepts and evidence. *Psychological Science in the Public Interest*, 9(3): 105–109. (Sage CA: Los Angeles, CA). doi: doi:10.1111/j.1539-6053.2009.01038.x

Retnowati, E., Ayres, P., Sweller, J. 2018. Collaborative learning effects when students have complete or incomplete knowledge. *Applied Cognitive Psychology* 32(6). https://doi.org/10.1002/acp.3444.

Saharudin, A. M., Yusoff, M., Haron, H., Latif, R. A. 2018. An analysis of factors influencing nstudents performance in programming assessment. *Advanced Science Letters* 24(11): 8182–8185. doi: 10.1166/asl.2018.12519.

Scotts, S., Palincsar, A. 2013. Sociocultural theory. Retrieved 29 August, 2019 from http://dr-hatfield.com/theorists/resources/sociocultural_theory.pdf

Seralidou, E., Douligeris, C. 2021. Learning programming by creating games through the use of structured activities in secondary education in Greece. *Education and Information Technologies* 26(1): 859–898. doi: 10.1007/s10639-020-10255-8

Seyal, A., Mey, Y., Matusin, M., Norzainah, H., Abdul-Rahman, A. 2015. Understanding students learning style and their performance in computer programming course: Evidence from bruneian technical institution of higher learning. *International Journal of Computer Theory and Engineering* 7(3): 241–247. doi: 10.7763/IJCTE.2015.V7.964

Sherriff, M. 2016. *Pair Programming in the Classroom*. Retrieved 6 May, 2018 from https://pdfs.semanticscholar.org/…/e196739701ddaaab93755609cc7218ad4095.pdf

Siegle, D. 2017. Technology: Encouraging creativity and problem solving through coding. *Gifted Child Today* 40(2): 117–123. doi: 10.1177/1076217517690861

Sklirou, T. S., Andreopoulou, A., Georgaki, A., Tselikas, N. D. 2020. Introducing secondary education students to programming through sound alerts. *European Journal of Engineering and Technology Research* 5(12): 130–139. doi: 10.24018/ejers.2020.5.12.2298.

Suhrowardi, S. 2020. Improving writing skills: An implementation of "think pair share" for Islamic junior schools students. *Journal of Languages and Language Teaching* 8(3): 287–296. doi: 10.33394/jollt.v8i3.2751

Umar, I. N., Hui, T. H. 2012. Learning style, metaphor and pair programming: Do they influence performance? *Procedia - Social and Behavioral Sciences*, 46: 5603–5609. doi: 10.1016/j.sbspro.2012.06.482.

West Africa Examinations Council Chief Examiners' Report. 2014. General comment, weakness/remedies and candidate's

strength. Retrieved 2 December 2018 from https://waeconline.org.ng/e-learning/Computer/Comp223mq1.htm

West Africa Examinations Council Chief Examiners' Report. 2015. General comment, weakness/remedies and candidate's strength. Retrieved 2 December 2018 from https://waeconline.org.ng/e-learning/Computer/Comp224mq1.htm

West Africa Examinations Council Chief Examiners' Report. 2016. General comment, weakness/remedies and candidate's strength. Retrieved 2 December 2018 from https://waeconline.org.ng/e-learning/Computer/Comp225mq1.htm

West Africa Examinations Council Chief Examiners' Report. 2017. General comment, weakness/remedies and strength. Retrieved 2 December 2018 from https://waeconline.org.ng/e-learning/Computer/Comp226mq1.htm

West Africa Examinations Council Chief Examiners' Report. 2018. General comment, weakness/remedies and candidate's strength. Retrieved 2 December 2018 from https://waeconline.org.ng/e-learning/Computer/Comp227mq1.htm

Xu, B., Yan, S., Gao, K., Zhang, Y., Yu, G. 2020. Influence of periodic role switching intervals on pair programming effectiveness. In G. Wang, X. Lin, J. Hendler, W. Song, Z. Xu, & G. Liu (Eds.), *Web Information Systems and Applications* (pp. 3–14). Cham: Springer International Publishing. doi: 10.1007/978-3-030-60029-7_1.

Yalagi, P. S., Dixit, R. K., Nirgude, M. A. (2020). Enhanced programming learning model (EPLM) through continuous collaborative coding (CCC) practice. *Journal of Engineering Education Transformations*, 33(Special Issue), 561–566. doi: 10.16920/jeet/2020/v33i0/150117

Yeboah, T., Sarpong, A. 2012. A study to investigate learning a style that has higher grade achievement in computer programming. *Journal of Engineering, Computers & Applied Sciences* 1(3): 33–38.

Yunikawati, N. A., Istiqomah, N., Hardinto, P., Susilo, Y. H., Prayitno, P. H., & Satrio, Y. D. 2021. Is Effective Think Pair Share (TPS) Used for Slow Learner Students (SLS)? Case Studies in Economic Development. 224–227. *Atlantis Press*. doi: 10.2991/assehr.k.210304.047

(*Manuscript received: 25th December, 2021; accepted: 6th January, 2022)*