Effect of Collaborative Programming on Students Achievement Learning Object-Oriented Programming Course

Krismadinata, Efan*, Chérifa Boudia, Jalius Jama, and Arie Yandi Saputra

***Abstract*—Employers need qualified human resources with high competitiveness and employability skills to compete in the age of technological disruption especially collaborative work. IT Students could fill the profile. Nevertheless, teachers still face challenges in teaching object-oriented programming (OOP) to students who struggle with complexity that involves a level of abstraction necessary to understand the concepts. The objective of this paper is to discern and study the overall effects non-real-time collaborative programming using Media Wiki in OOP courses. According to the quasi-experimental method conducted among 56 students majoring in Informatics, and utilizing Media Wiki, a collaborative programming learning model has been applied. The study was divided into an experimental class of 30 students and a control class of 26 students. The survey results show that students get better programming achievement. There is a decrease in the level of OOP abstraction by students. Students' collaborative experience increases and indirectly improves their collaboration skills. After the experiment, students were asked to complete an online questionnaire mainly occurred to assess the effectiveness of the teaching and learning method using of Media Wiki for collaborative programming. As results, the students report that they are able to assimilate complex concepts and code more easily in collaborative learning using Media Wiki.**

***Index Terms*—Collaborative programming, collaborative learning, object-oriented programming (OOP), teaching, learning, media Wiki**

I. Introduction

Employers need qualified human resources with high competitiveness and employability skills to compete in the age of technological disruption. Especially, the current IT industry not only requests prospective programmers to have professionalism and personal skills, but also requires the ability to develop collaborative software [1]. Basically, programming can be done individually, but in the 21st century, industry involves collaborative programming skills so the main goals of programming can be achieved [2]. In addition, students are often charged with programming tasks on applications or software in teams consisting of several students [3], with effective collaboration between programmers [4].

However, students as the employee suffer from insufficient profile for job application due to many skills lacks that is directly related to educational quality. For IT students, these skills could be learned efficiently through improved teaching and learning strategies of computer sciences courses such as oriented-object programming (OOP) course where OOP is a programming paradigm designed to represent objects into procedure blocks by taking into account the concepts (objects, classes, properties, methods, etc.) [5]. One of the challenges faced by students is that they do not understand the OOP concept [6]. OOP requires a high grasping complex concepts and applying these notions to sophisticated programming methods [7]. OOP is difficult to learn for students, especially novice students, because both the basic concepts (objects and classes) and fundamental concepts (encapsulation, inheritance, and polymorphism) are abstract [8–11], hard to describe [12–14] and more than just understanding the definition [15]. Some of these difficulties include the fact that OOP translates real-world objects into object-oriented code [16]; sometimes the lecturers do not have enough experience with the OOP concept on how to teach the abstraction of the concept to students [17]; many applications are designed based on object interactions which is different from the procedural programming style [18]; of course, the transition from a procedural programming paradigm to an object-oriented paradigm is a challenge in itself [19]. Due to complexity and abstraction, the OOP concept needs innovative solutions, which can serve as a means to learn these concepts [20]. There is no unique and specific method to teach OOP, therefore, lecturers must experiment extensively in order to identify better and more successful ways to introduce OOP [21]. In other words, teaching OOP notions and other difficult subjects is always challenging, and consequently, many students fail this course [22].

Deploying collaborative programming in teaching and learning experience can alleviate the issues mentioned above. Collaborative programming allows students to connect with one another and benefit from one other's resources and experiences, resulting in improved programming learning [23, 24]. Collaborative programming, besides being able to generate a lot of student ideas by observing other students' program codes, also indirectly adds to the student experience by seeing other students' mistakes when coding [25]. Collaborative programming can also improve problem solving abilities as a result of inherited metacognitive activities [26]. In addition, students can also create better programs when using collaborative programming [27].

It is possible to distinguish two distinct types of collaborative programming real-time and non-real-time

Manuscript received Nov 10, 2022, revised December 8, 2022, accepted January 28, 2023.

Krismadinata and Jalius Jama are with the Universitas Negeri Padang, Padang, West Sumatra, Indonesia. E-mail: krisma@ft.unp.ac.id (K.), jamajalius@yahoo.com (J.J.)

Efan is with Universitas Negeri Padang, Padang, West Sumatra, Indonesia and with the Institut Teknologi Pagar Alam, Pagar Alam, South Sumatra, Indonesia.

Chérifa Boudia is with the University Mustafa Stambouli of Mascara, Mascara, Oran, Algeria. E-mail: cherifa.boudia@univ-mascara.dz (C.B.)

Arie Yandi Saputra is with STMIK Bina Nusantara Jaya, Lubuklinggau, South Sumatra, Indonesia. (mail: arielahat@gmail.com (A.Y.S.)

*Correspondence: efan@itpa.ac.id (E.)

collaborative programming. The first allows students to edit a code together and deploy update together at the same time, the second is adopted for version control systems use [4]. The objective of this paper is to discern and study the overall effects of non-real-time collaborative programming using Media Wiki for teaching OOP courses. This quasi-experimental research was conducted by involving 56 undergraduate students (S1) majoring in Informatics, which was divided into experimental class and control class.

The following research questions are stated specifically to maintain the focus of research:

RQ1: Can students reach better achievement in collaborative programming than in traditional programming?

RQ2: Will there be a decrease in the level of abstraction of OOP by students when using collaborative programming using Media Wiki compared to conventional programming?

RQ3: Does implementing collaborative programming using Media Wiki enhance student collaboration more effectively than conventional approaches?

The hypotheses for the three research questions above are:

Ha1: Students reach better achievement in collaborative programming than in traditional programming Students get better programming achievement with collaborative programming compared to conventional programming.

Ha2: There was a decrease in the level of abstraction of OOP by students when using collaborative programming using Media Wiki compared to conventional programming.

Ha3: The application of collaborative programming using Media Wiki enhance student collaboration more effectively than conventional approaches.

The following structure describes the research method used in this article: Section II describes the quasi-experimental study, Section III presents the results of the study and discusses the answers to the research questions, and Section IV presents the study's conclusions.

II. Related Work

There is a need for a complete redesign of teaching strategies and methods in computer science, especially in programming courses. Mostly this is due to the mistaking the abstract concepts learnt in lectures with their implementation for computer coding. In lab sessions, students appear disconnected from the practice of programming. Additionally, they do not use well-known tools instead of current technologies. Hence, a set of instruments that go beyond pedagogical tools are required. These components will offer an excellent programming learning experience in order to increase students understanding of concepts, build up skills, and enhance their reasoning process. Researchers argue the necessity of developing a strong environment for programming education with the ICT support. The use of some educational tools has contributed to a better understanding of concepts and the ability to develop programming skills such as games. There is still an overwhelming percentage of education that is teacher-centered, where the instructor serves as a knowledge source, a role model for participation, and a catalyst for student engagement and teamwork. Internet use, particularly Facebook, has made a huge contribution to student-centered learning.

For most students, programming is a very difficult task [28, 29], since it requires strong levels of metacognitive ability, including abstraction and comprehension, tenacity, and competences to complete the stages of problem-solving and programming [30]. In teaching, students' progress from beginners to experts by acquiring skills, developing intelligence, and learning how to communicate with teachers and classmates, recognize their own convictions and attitude, and continue their assignment to the next higher stage. Additionally, becoming a computer programmer is thought to be a highly challenging path, especially for beginners. It necessitates continuing education in programming. If students do not improve, they will regress. Learning to code demands a variety of techniques, including using open-source codes, which are freely available in books and websites that host and distribute millions of lines of software, be aware of the function of each code block, execute experiments using test data, and reading carefully program's outputs to learn more about compilers.

A few of these coding strategies are useful in certain situations, while others concentrate on the environments and tools used in programming. Therefore, it is crucial to understand what obstacles make this form of learning difficult and how learners could learn effectively and simply.

Various methods and strategies are used to teach OOP courses: Conventional face-to-face [31], distance learning [32], and hybrid (blended) learning [33].

Collaborative learning is a situation where students engage in activities such as exchanging and discussing thoughts and resources, dividing roles [34] to meet learning objectives [35]. When students collaborate, they pool their resources together to create a complementary combination that produces exceptional results. These students initiate a discussion and work by considering several roles to find the solution together for the problem. At least two students participate in collaborative learning to solve the same assignment together. Andriessen [36] is arguing about encouraging debate using illustrations can independently teach students critical thinking.

Pair programming is the most popular method, where two students use one computer to work on a coding challenge, one serving as the "pilot" who directly code and the other as the "navigator" to detect and fix problems [37]. The collaborative learning activities involve smaller or bigger groups formed of more than two individuals must follow scenarios or scripts based on various tasks [38]. In conventional learning contexts, the assignment of those tasks could be managed automatically [39]. TPS (Think-Pair-Share) is an alternative strategy adopted in collaborative learning. Students proceed in three phases: first, everyone in the group thinks about the assignment and function independently for a determinated moment, then they communicate with their mates to write the appropriate code with the assistance of the teacher. Students, in the last stage, debate about the best solutions [40]. Programming involves skills like critical thinking, information analysis and synthesis, as well as coordinating and planning in groups, are just a few of the numerous talents needed for programming.

These capacities are reliant on other abilities, such as

social competences. It increases group members' motivation and communication [41]. Collaboration and ICT technologies strengthen student interaction [42]. Through online discussion, students can debate subject-related issues, exchange concepts, and/or share code fragments [43, 44]. Collaboration on a programming assignment activity involves creating several pedagogical scenarios according to the group setting. As a result, collaborative learning may be a suitable practice for learning programs.

III. Method

A quasi-experimental method was conducted among 56 students majoring in Informatics, by deploying a collaborative programming learning model using Media Wiki.

A. Participants

56 informatics engineering majors joined this study. The experiment included two classes: an experimental class with 30 students and a control class with 26 students. The trial was carried out only in the laboratory session according to the planned time. Students are divided into five groups consisting of five members for each group. The learning experience was completed successively by each group.

B. Learning Topics and Procedures

1) Learning topics

The learning topics for the OOP course is carried out as shown in Table I using the OO-PHP programming language. This study only applies the first four topics to the test sample. The study continued for four weeks during the normal semester of the academic year 2021.

TABLE I: Learning Topics

NO.	TOPICS
1	Classes, Objects, Properties & Methods
2	Constructor
3	Inheritance
4	Overriding
5	Visibility
6	Accessor Method
7	Static Keyword
8	Constant
9	Abstract Class
10	Interface
11	Autoloading
12	Namespace

2) Learning procedure

Fig. 1 depicts the learning procedure of collaborative learning inspired form [25] where each learning topic goes through the procedure cycle as follow:

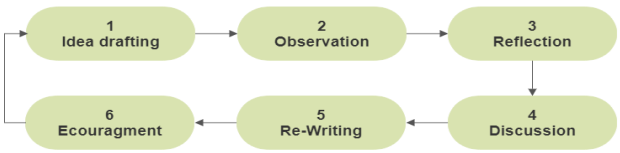


Fig. 1. Stages of collaborative programming using *Media Wiki* — Adopted from [25].

1) Stage 1 (Idea drafting): in the first step, every student formulates individually the solution in a code editor such as Visual Studio Code, and then makes any necessary to the initial edit. Writing the paper first draft is usually difficult for students.
2) Stage 2 (Observation): Afterwards, students are asked to carefully examine the code written by their group members and those from other groups using Media Wiki's history log and editing section.
3) Stage 3 (Reflection): in this step, students are expected to recognize their mistakes in their programs then suggest solutions.
4) Stage 4 (Discussion): after that, they should use Media Wiki's discussion board to explore numerous different ideas with their mates.
5) Stage 5 (Re-writing): Finally, with new ideas, students improve and update their program.
6) Stage 6 (Encouragement): Students are also urged to proofread the code and informed that their excellent work will be publicized.

In adequate time, teachers intervene in each stage, monitor the learning process, and handle situations corresponding to unique programming issues and student learning progress.

In this study's quasi-experimental methodology, in each class, students in each class were assigned to a collaborative programming group randomly. Teaching the experimental group was conducted using a collaborative programming technique, and the students use remote interactivity to co-edit the software with their peers without leaving their places. To execute and test the program in the browser, they must copy it from Media Wiki and paste it into the Visual Code Studio editor.

The students in the control group were guided using conventional teaching methods, sitting next to their partners and collaborating face-to-face while editing their code in the Visual Code Studio editor on the same computer.

C. Data Collection

In this part, three distinct types of data were gathered and analyzed: post-test grades, survey responses, and Media Wiki Platform history records for each group. A post-test evaluating declarative syntactic and declarative conceptual programming knowledge was used to assess the students' programming thinking. After the experiment, students were asked to complete an online questionnaire mainly occurred to assess the effectiveness of the teaching and learning method using of Media Wiki for collaborative programming, consisted of 16 Likert scale questions and a column to accommodate student comments and responses regarding the use of Media Wiki in the application of collaborative programming.

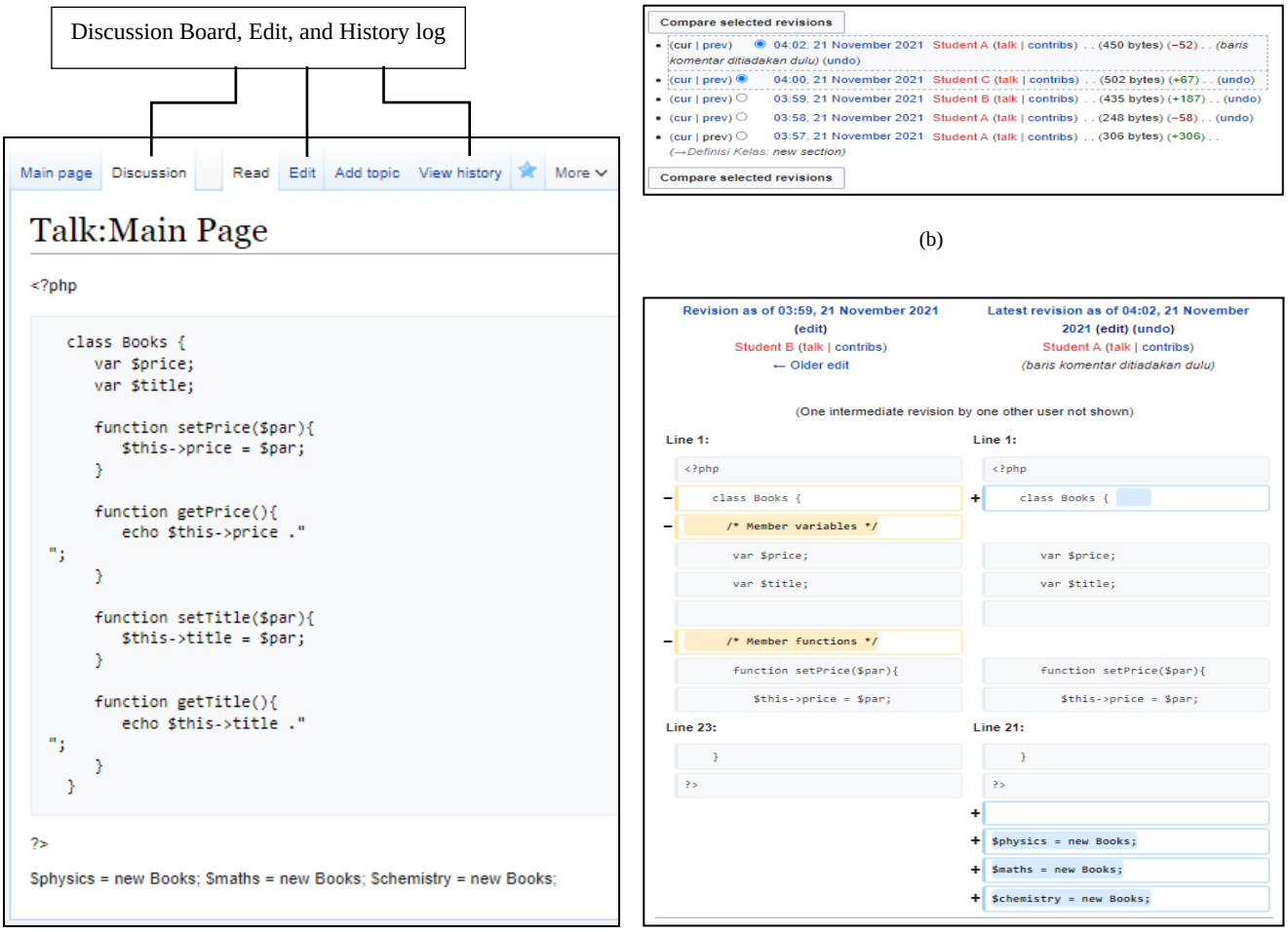
1) Post-test instrument

TABLE II: Categories of Questions

Cognitive Domain	No. of Question	Portion
C1	1, 2, 3, 7, 16	25%
C2	6, 9, 10, 11, 17	25%
C3	4, 5, 8, 12	20%
C4	13, 14, 15, 18, 19, 20	30%

This test consists of 20 multiple choice questions about basic programming concepts and program tracing. The

post-test questions given consist of elements C1 (remembering), C2 (understanding), C3 (applying) and C4 (evaluating) based on Bloom's taxonomy [28]. The categories of questions are classified into four cognitive domains as shown in Table II.



(a) (b) (c)

Fig. 2. (a) Main page, (b) History log, (c) Version modification.

2) *Questionnaire*

A link to the online questionnaire was made, http://bit.ly/3Z71ACP that consists of 16 Likert scale questions related to the abstraction of OOP material, laboratory sessions, and ICT technology support during the implementation of collaborative programming. The link was shared via students' WhatsApp group to both experiment and control class. In addition, the questionnaire provides a column to accommodate student comments and responses regarding the use of Media Wiki in the application of collaborative programming. Table III shows the categories of survey's questions.

TABLE III: CATEGORIES OF SURVEY'S QUESTIONS

Category	Question
Ch1. Abstraction Theory	Q1. OOP material is not too abstract
	Q2. OOP material is not too complex
	Q3. I had no trouble switching from the procedural paradigm to OOP
	Q4. In my opinion, OOP is very suitable to be studied by collaborating
	Q5. I think Object-Oriented Programming can accommodate large programs
Ch2. Laboratory Session	Q6. I easily found an idea for the code
	Q7. I can easily see the friend code
	Q8. I easily find and revise errors in the code
	Q9. I get convenience when discussing with friends
	Q10. I am motivated to improve program code
Ch3. ICT Technology Support	Q11. I think ICT technology that supports collaborative programming is needed
	Q12. The technology used must be able to support the collaboration process, especially
	Q13. the technology used must be able to accommodate the process of various source code (programming code)
	Q14. the technology used to facilitate the function to manage shared documents
	Q15. the technology used must have discussion facilities
	Q16. the technology used must support offline and online networks at the same time

3) *Media Wiki platform*

Fig. 2 depicts the Media Wiki interface. Discussion rooms, updates, and history logs are the most common functions utilized in learning activities. This allows students to talk about programming challenges they've been given, update the code, run it and see the results. During the process, students can compare the revisions between different

versions of the code. All programs written are documented in the following Media Wiki log, which includes documentation of the specific time, contributors, and contribution type. The version comparison tool compares two versions of a program based on the highlighted label provided by the Media Wiki system, with red text indicating the difference between the two. Different colored fields display the differences (yellow represents the previous version and green represents the new version). Students may then use the history log to compare their code between versions or with other students' work. This feature may also be used by researchers and lecturers to obtain insight into students' thought processes.

IV. Results and Discussion

A. Programming Learning Outcomes

Programming scores obtained from the post-test were analyzed using the ANCOVA method involving class variables (experimental and control), acquisition scores and the last Grade Point Average (GPA). Cronbach's Alpha value for the post-test instrument is 0.808 which indicates that the instrument used is reliable. As a covariable, GPA significantly made changes in programming learning achievement (F (1.53) = 718.675, $p < 0.05$, 2 = 0.931). There is a significant difference in programming learning achievement based on the teaching model by controlling the GPA statistically (F (1.53) = 6.101, $p < 0.05$, 2 = 0.103) (see Table IV).

The objective of this study's quantitative analysis was to determine if the experimental class's usage of collaborative learning was more effective than the control class's. The following hypothesis was evaluated by the researchers during the analysis:

Mathematical hypothesis

Ha1: $\mu A > \mu B$

A: Experimental Class

B: Control Class

The results of the descriptive analysis (see Table V) found that the average programming learning achievement of the experimental class (M = 67.00, SD = 21.600) was higher than the control class (M = 61.73, SD = 22.889). Post hoc using Bonferonni (see Table VI) found that there was a significant difference in learning achievement between the experimental class and the control class (Mean Difference = 3.890, SE = 1.577, $p < 0.05$).

The hypothesis tested using Table *t* in Table V is Ha1: $\mu A > \mu B$ that is Students get better programming achievement with collaborative programming compared to conventional programming. From the Table VI, it obtained a significant value of *t* of 0.017, using the criteria of a Significant Level of Error that is $\alpha = 0.05$, then it can be concluded that the Significant Value t is smaller than the limit value of error $\alpha = 0.05$. The hypothesis acceptance criteria in this study are H_0 accepted if sig.t>$\alpha = 0.05$. From the results of the study, it was found that sig.t = 0.017 <$\alpha = 0.05$, it was concluded that $H_0$1 was turned down and eventually accepted Ha1: $\mu A > \mu B$. This indicates that students who used Media Wiki for collaborative programming had greater learning results on average than those who employed traditional teaching methods.

TABLE IV: Results of ANCOVA Analysis

Source	Type III Sum of Squares	df	Mean Square	F	Sig.	Partial Eta Squared
Corrected Model	25185.042*	2	12592.521	364.941	0.000	0.932
Intercept	14304204	1	14304204	414.547	0.000	0.887
GPA	24798.318	1	24798.318	718.675	0.000	0.931
Class	210.518	1	210.518	6.101	0.017	0.103
Error	1828.798	53	34.506			
Total	260375.000	56				
Corrected Total	27013.839	55				

Note: * R Squared = 0.926 (Adjusted R Squared = 0.923)

TABLE V: Descriptive Statistics of Post Test Scores

Class	mean	SD	N
Experiment	67.00	21.600	30
Control	61.73	22.889	26

The research findings show a positive impact on increasing cognitive achievement after the application of the collaborative programming learning model. Although the difference in the mean value of the experimental class and the control class is not too large, the significance value shows a significant difference between the two treatments. Regarding the cognitive domain, it can be explained that the experimental class and the control class have relatively the same level of achievement in the domains C1 (difference 1%), C2 (difference 5%), and C3 (difference 2%). However, a contrasting difference is seen in the achievement of the C4 level with a difference of 15% higher for the experimental class (See Fig. 3). This proves that the application of a collaborative programming model using the Media Wiki platform can not only improve understanding and programming skills, but also improve high-level thinking skills in OOP courses. This is in line with prior research, which claims that program implementation necessitates higher-order thinking abilities like application and synthesis, as well as the ability to learn procedural programming by witnessing other people's coding techniques [25].

TABLE VI: MEAN VALUE OF THE EXPERIMENTAL CLASS AND THE CONTROL CLASS

(I) Class	(J) Class	Mean Difference (IJ)	Std. Error	Sig.a	95% Confidence Interval for Differences	
					Lower Bound	Upper Bound
Experiment	Control	3.890*	1.575	0.017	0.731	7.048
Control	Experiment	−3.890*	1.575	0.017	−7.048	−0.731
Based on estimated marginal means						

Note: * Adjustment for multiple comparisons: Bonferroni.

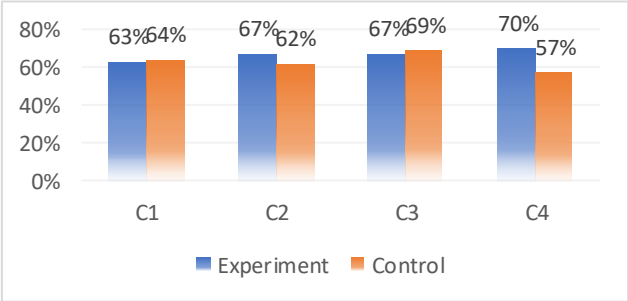


Fig. 3. Number of answers to post test questions that are answered correctly.

B. Questionnaire Summary

A questionnaire consisting of 16 questions on a Likert scale with a range (1–5) was given to students in the experimental class and control class after the post-test (see Table VII). The t-test was carried out on the results of the questionnaire to see the significance of the difference between the experimental class and the control class. The significance of the difference occurred in Q1–Q11 ($p < 0.05$) while in Q12–Q16 there was no significant difference ($p > 0.05$).

TABLE VII: RESULTS OF T-TEST CATEGORIES CH1, CH2, AND CH3 BETWEEN THE EXPERIMENTAL CLASS AND THE CONTROL CLASS

Category	Question	N=30		N=26		*t*	*p*
		mean	SD	mean	SD		
Ch 1	Q1	43.67	0.72	29.62	0.66	7.568	0.000
	Q2	44.33	0.68	27.31	0.83	8.457	0.000
	Q3	44.67	0.78	30.00	0.49	8.303	0.000
	Q4	44.00	0.77	36.92	0.88	3.202	0.002
	Q5	43.33	0.66	36.54	0.85	3.371	0.001
Ch 2	Q6	45.00	0.68	31.54	0.46	8.496	0.000
	Q7	45.67	0.57	32.31	0.51	9.164	0.000
	Q8	43.67	0.62	33.08	0.55	6.751	0.000
	Q9	44.00	0.72	35.00	0.81	4.384	0.000
	Q10	46.00	0.62	34.23	0.70	6.652	0.000
Ch 3	Q11	43.33	0.66	47.31	0.45	−2.585	0.012
	Q12	46.33	0.49	47.69	0.43	−1.095	0.278
	Q13	44.67	0.57	46.54	0.49	−1.310	0.196
	Q14	44.33	0.63	44.62	0.65	−0.166	0.869
	Q15	44.67	0.68	45.00	0.65	−0.187	0.853
	Q16	44.00	0.72	44.62	0.71	−0.321	0.750

1) Abstraction of OOP

The result of the Ch1 (abstraction) of questionnaire on this study was designed to test whether there decreasing in the level of abstraction of OOP by students when using collaborative programming using Media Wiki compared to conventional programming? To do the analysis, the researchers used the hypothesis as follows:

Mathematical hypothesis

Ha2: $\mu A > \mu B$

A: Experimental Class

B: Control Class

The first 3 questions have a fairly high mean difference between the experimental class and the control class (Q1.t = 7,568, Q2.t = 8,457, and Q3.t = 8,303) where the mean value of the experimental class is greater than control class mean. Detail of each mean comparison as follows:

Q1: 43.67 > 29.62

Q2: 44.33 > 27.31

Q3: 44.67 > 30.00

Q4: 44.00 > 36.92

Q5: 43.33 > 36.54

Overall, mean of experiment class is greater than control class. It can be explained that the control class that does not receive the collaborative programming model indirectly proves the complexity and abstraction of the OOP concept. Meanwhile, the experimental class that received the treatment also indirectly proved that the collaborative programming model was quite effective in reducing abstraction problems in the OOP learning process. This condition was concluded that Ha2: $\mu A > \mu B$ was accepted. It means there was a decrease in the level of abstraction of OOP by students when using collaborative programming using Media Wiki compared to conventional programming.

2) Laboratory session

Category Ch2 (laboratory session) was designed to measure whether the implementation of collaborative programming using Media Wiki improve student collaborative experiences more than conventional programming? To do the analysis, the researchers used the hypothesis as follows:

Mathematical hypothesis

Ha3: $\mu A > \mu B$

A: Experimental Class

B: Control Class

Overall, mean of experiment class is greater than control class. It can be explained that students of the experiment class that received the collaborative programming model get new experiences that help the learning process when collaborating on the Media Wiki platform compared to the control class that did not receive the treatment. This condition was concluded that Ha3: $\mu A > \mu B$ was accepted. It means the application of collaborative programming using Media Wiki is more able to improve the collaborative student experience compared to conventional programming.

Category Ch2 was also designed to measure the practicality of the steps of collaborative programming procedures. This measurement is focused on the experimental class where the steps of the procedure are

applied. Question Q6 which represents the measurement of the idea-drafting stage is dominated by a very positive response (strongly agree: 60%). Likewise, with Q7 which represents observation (strongly agree: 60%), Q9 for discussion (strongly agree: 53%) and Q10 for encouragement (strongly agree: 67%). Only the reflection stage and the re-writing stage were represented by Q8, dominated by the lower response (agree: 50%).

In general, these stages are included in the very practical category. This is evidenced by the responses given to questions Q6–Q10 by experimental class students compared to the responses of control class students (see Table VIII). The positive response from this experimental class proves that students get new experiences that help the learning process when collaborating on the Media Wiki platform. In other words, Ha3 is declared to be accepted.

TABLE VIII: STUDENTS' RESPONSE FOR EACH STAGES OF COLLABORATIVE PROGRAMMING USING MEDIA WIKI

Stage	Strongly agree	Agree	Just Agree	Disagree	**Do not agree**
Idea drafting (Q6)	60%	30%	10%	0%	0%
Observation (Q7)	60%	37%	3%	0%	0%
Reflection & Re-writing (Q8)	43%	50%	7%	0%	0%
Discussion (Q9)	53%	33%	13%	0%	0%
Encouragement (Q10)	67%	27%	7%	0%	0%

3) *ICT technology support*

Category Ch3 (ICT technology support) was designed to see the level of need for ICT technology support for teaching OOP courses. Based on statistical data, it can be explained that there is no difference between the experimental class and the control class. The six questions in this category (Q12–Q16), the mean difference is not more than 1.5 only (0.321–1.310), but the difference is slightly higher in Q11 (2.585). This creates two conditions. The first condition states that some respondents from the experimental class are satisfied with the Media Wiki platform, but need some features to be added. While the second condition states that most of the respondents from the experimental class and of course all respondents from the control class propose the use of more sophisticated technology.

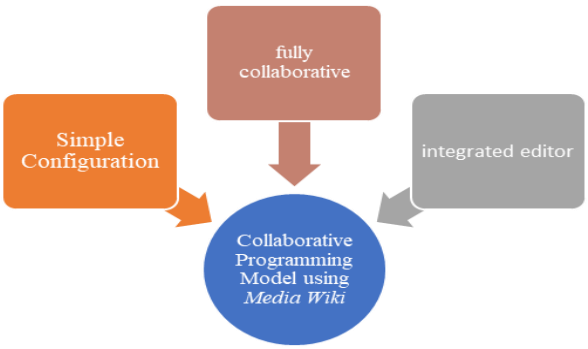


Fig. 4. Invention criteria to collaborative programming model using Media Wiki.

Some criteria of the ICT technology are needed to support teaching and learning OOP course collaboratively. The criteria are: fully supports collaborative programming, minimizes the configuration process, and is able to run program execution commands without having to move the code to a separate editor. Fig. 4 depicts the invention the criteria to Collaborative Programming Model using Media Wiki.

V. CONCLUSION

This study has successfully piloted a collaborative programming model using the support of the Media Wiki platform in learning OOP courses as a response to two major problems faced by lecturers and students: abstraction of material and challenging the industrial world. As a result, this research states that students get better programming achievement compared to conventional programming. In addition, this study found that there was a decrease in the level of OOP abstraction by students when using collaborative programming using Media Wiki compared to conventional programming. Regarding the collaboration process, the application of the model on the Media Wiki platform succeeded in improving the collaborative experience of students and indirectly improving their collaboration skills. The questionnaire results show that students found that, among other things, the benefits of this method made it easier for them to code programs. Some touch of more sophisticated technology from Media Wiki is needed so that the above achievements can be improved. Among the technology criteria are: fully supports collaborative programming, minimizes the configuration process, and is able to run program execution commands without having to move the code to a separate editor.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

Krismadinata conducted the research and verified the manuscript before it was be submitted. Efan conducted the research and wrote the manuscript; Chérifa Boudia enhanced the writing quality and the manuscript language before it was be submitted; Chérifa Boudia also contributed some ideas and a section on the manuscript; Jalius Jama verified the manuscript before it was be submitted; Arie Yandi Saputra was involved in collecting and managing the empirical data; all authors had approved the final version.

FUNDINGS

This work was supported in part by the Institut Teknologi Pagar Alam, Indonesia and in part by Universitas Negeri Padang, Indonesia.

REFERENCES

[1] O. G. Glazunova, O. V. Parhomenko, V. I. Korolchuk, and T. V. Voloshyna, "The effectiveness of GitHub cloud services for implementing a programming training project: Students' point of view," *Journal of Physics: Conference Series*, vol. 1840, no. 1, 2021.

[1] X. Lai and G. K. W. Wong, "Collaborative versus individual problem solving in computational thinking through programming: A meta-analysis," *British Journal of Educational Technology*, 2021.

[2] M. Borowski, J. Zagermann, C. N. Klokmose, H. Reiterer, and R. Radle, "Exploring the benefits and barriers of using computational notebooks for collaborative programming assignments," in *Proc. the Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE*, Feb. 2020, pp. 468–474.

[3] H. Fan, J. Gao, H. Zhu, Q. Liu, Y. Shi, and C. Sun, "Balancing conflict prevention and concurrent work in real-time collaborative programming," in *Proc. the 12th Chinese Conference on Computer Supported Cooperative Work and Social Computing*, Sep. 2017, pp. 217–220.

[4] E. Lotfi and B. Mohammed, "Teaching object-oriented programming concepts through a mobile serious game," in *Proc. the 3rd International Conference on Smart City Applications*, Oct. 2018.

[5] Y. S. Wong and M. H. M. Yatim, "A propriety multiplatform game-based learning game to learn object-oriented programming," in *Proc. the 2018 7th International Congress on Advanced Applied Informatics, IIAI-AAI 2018*, Jul. 2018, pp. 278–283.

[6] G. A. S. Oliveira and R. Bonacin, "A method for teaching object-oriented programming with digital modeling," in *Proc. the IEEE 18th International Conference on Advanced Learning Technologies, ICALT 2018*, Aug. 2018, pp. 233–237.

[7] Z. Z. Abidin and M. A. A. Zawawi, "OOP-AR: Learn object-oriented programming using augmented reality," *International Journal of Multimedia and Recent Innovation*, vol. 2, no. 1, 2020.

[8] A. J. Olier, A. A. Gomez, and M. F. Caro, "Design and implementation of a teaching tool for introduction to object-oriented programming," *IEEE Latin America Transactions*, vol. 15, issue 1, pp. 97–102, 2017.

[9] T. Tanielu, R. A. Ola, E. Varoy, and N. Giacaman, "Combining analogies and virtual reality for active and visual object-oriented programming," in *Proc. the ACM Conference on Global Computing Education, CompEd 2019*, May 2019, pp. 92–98.

[10] K. K. Zaw, N. Funabiki, E. E. Mon, and W. C. Kao, "An informative test code approach for studying three object-oriented programming concepts by code writing problem in Java programming learning assistant system," in *Proc. the 2018 IEEE 7th Global Conference on Consumer Electronics, GCCE 2018*, 2018.

[11] Y. Asano and K. Kagawa, "Development of a web-based support system for object-oriented programming exercises with graphics programming," in *Proc. the 2019 18th International Conference on Information Technology Based Higher Education and Training (ITHET)*, 2019.

[12] E. Lotfi, B. Y. Othman, and B. Mohammed, "Towards a mobile serious game for learning object-oriented programming paradigms," in *Proc. the Third International Conference on Smart City Applications*, 2019, pp. 450–462.

[13] J. M. Su and F. Y. Hsu, "Building a visualized learning tool to facilitate the concept learning of object-oriented programming," in *Proc. the 2017 6th IIAI International Congress on Advanced Applied Informatics, IIAI-AAI 2017*, Nov. 2017, pp. 516–520.

[14] V. Thurner, "Fostering the comprehension of the object-oriented programming paradigm by a virtual lab exercise," in *Proc. the 2019 5th Experiment at International Conference*, 2019.

[15] K. Kanaki and M. Kalogiannakis, "Introducing fundamental object-oriented programming concepts in preschool education within the context of physical science courses," *Educ Inf Technol (Dordr)*, vol. 23, no. 6, 2018.

[16] V. Holmstedt and S. A. Mengiste, "Zirr—Why and how practice impacts confidence in introductory object-oriented programming Courses," in *Proc. the Fifth International Conference Modelling and Development of Intelligent Systems*, 2017, pp. 29–42.

[17] P. Sedlacek, M. Kvet, and M. Václavková, "Development of FRIMAN—Supporting tool for object-oriented programming teaching," *Open Computer Science*, vol. 11, no. 1, 2021.

[18] S. Abbasi, H. Kazi, A. W. Kazi, K. Khowaja, and A. Baloch, "Gauge object-oriented programming in student's learning performance, normalized learning gains and perceived motivation with serious games," *Information*, vol. 12, no. 3, p. 101, Feb. 2021.

[19] E. Kucera, O. Haffner, and R. Leskovsky, "Multimedia application for object-oriented programming education developed by unity engine," in *Proc. the 30th International Conference on Cybernetics and Informatics*, Jan. 2020.

[20] J. Gabaruk, D. Logofatu, D. Großkreutz, and C. Andersson, "On teaching Java and object-oriented programming by using children board games," in *Proc. the IEEE Global Engineering Education Conference, EDUCON*, 2019, vol. April-2019.

[21] M. A. López, E. V. Duarte, and E. C. Gutiérrez, "Experience in teaching object-oriented programming and advanced topics of programming through the development of a video game project," in *Proc. the CEUR Workshop*, 2020, vol. 2747, pp. 109–118.

[22] B. Alorda, K. Suenaga, and P. Pons, "Design and evaluation of a microprocessor course combining three cooperative methods: SDLA, PjBL and CnBL," *Comput. Educ.*, vol. 57, no. 3, 2011.

[23] W. Y. Hwang, R. Shadiev, C. Y. Wang, and Z. H. Huang, "A pilot study of cooperative programming learning behavior and its relationship with students' learning performance," *Comput. Educ.*, vol. 58, no. 4, 2012.

[24] Y. T. Lin, C. C. Wu, and C. F. Chiu, "The use of wiki in teaching programming: Effects upon achievement, attitudes, and collaborative programming behaviors," *International Journal of Distance Education Technologies*, vol. 16, no. 3, pp. 18–45, Jul. 2018.

[25] H. J. Lee, "A learning process mechanism in CSCL," in *Proc. ICCE 2008: 16th International Conference on Computers in Education*, 2008.

[26] G. Braught, L. M. Eby, and T. Wahls, "The effects of pair-programming on individual programming skill," *ACM SIGCSE Bulletin*, vol. 40, no. 1, 2008.

[27] S. Ahmed and M. Amarif, "An interactive animation tool for java object-oriented programming understanding," *International Journal of Programming Languages and Applications*, vol. 9, no. 3, pp. 1–17, 2019.

[28] N. Passerini and C. Lombardi, "Postponing the concept of class when introducing OOP," in *Proc. the 2020 ACM Conference on Innovation and Technology in Computer Science Education*, 2020, pp. 152–158. doi: 10.1145/3341525.3387369.

[29] C. Boudia, A. Bengueddach, and H. Haffaf, "Collaborative strategy for teaching and learning object-oriented programming course: A case study at Mostafa Stambouli Mascara University, Algeria," *Informatica (Slovenia)*, vol. 43, no. 1, pp. 129–144, 2019.

[30] M. Husain, N. Tarannum, and N. Patil, "Teaching programming course elective: A new teaching and learning experience," in *Proc. the 2013 IEEE International Conference in MOOC, Innovation and Technology in Education, MITE 2013*, 2013.

[31] B. Estácio *et al.*, "Evaluating collaborative practices in acquiring programming skills: Findings of a controlled experiment," in *Proc. the 29th Brazilian Symposium on Software Engineering, SBES 2015*, 2015.

[32] F. Silvestre, P. Vidal, and J. Broisin, "Tsaap-notes—An open micro-blogging tool for collaborative notetaking during face-to-face lectures," in *Proc. the IEEE 14th International Conference on Advanced Learning Technologies, ICALT 2014*, 2014.

[33] A. V. P. Arboleda, L. R. R. Mazuera, and G. S. Montemiranda, "Competences that facilitate the achievement of the objectives of an introductory programming course," in *Proc. the 2015 International Conference on Interactive Collaborative Learning, ICL 2015*, 2015.

[34] J. Olivier, "Blended learning in a first-year language class: Evaluating the acceptance of an interactive learning environment," *Literator*, vol. 37, no. 1, 2016.

[35] M. I. Dascalu, C. N. Bodea, A. Moldoveanu, A. Mohora, M. Lytras, and P. O. de Pablos, "A recommender agent based on learning styles for better virtual collaborative learning experiences," *Comput. Human Behav.*, vol. 45, 2015.

[36] G. Temperman, B. De Lièvre, P. Bossaert, and J. De Stercke, "Effects of role distribution by learning style in a collaborative distance environment," in *Colloque TICE 2010—Nancy*, 2010.

[37] J. Andriessen, M. Baker, and D. Suthers, "Arguing to Learn: confronting cognitions in computer-supported collaborative learning environments," in *Computer-Supported Collaborative Learning Series (CULS, volume 1)*, J. Andriessen, M. Baker, and D. Suthers, Eds. 2013.

[38] A. Zagalsky, J. Feliciano, M. A. Storey, Y. Zhao, and W. Wang, "The emergence of GitHub as a collaborative platform for education," in *Proc. the 2015 ACM International Conference on Computer-Supported Cooperative Work and Social Computing*, 2015.

[39] F. J. Rodríguez, K. M. Price, and K. E. Boyer, "Exploring the pair programming process: Characteristics of effective collaboration," in *Proc. the Conference on Integrating Technology into Computer Science Education, ITiCSE*, 2017.

[40] P. Dillenbourg, "Over-scripting CSCL: The risks of blending collaborative learning with instructional design. In Three worlds of CSCL: Can we support CSCL," *Three Worlds of CSCL. Can we support CSCL?* 2002.

[41] M. Maleko, D. Nandi, M. Hamilton, D. D'Souza, and J. Harland, "Facebook versus blackboard for supporting the learning of programming in a fully online course: the changing face of computing

education," in *Proc. the 2013 Learning and Teaching in Computing and Engineering, LaTiCE 2013*, 2013.

[42] C. Armstrong, "Catalyzing collaborative learning: How automated task distribution may prompt students to collaborate," *E-Learning and Digital Media*, vol. 7, no. 4, 2010.

[2] S. M. Alyami and A. M. Alagab, "The difference in learning strategies in virtual learning environment and their effect on academic achievement and learning satisfaction for distance teaching and training program students," in *Proc. the 2013 4th International Conference on e-Learning Best Practices in Management, Design and Development of e-Courses: Standards of Excellence and Creativity, ECONF 2013*, 2013.

Copyright © 2023 by the authors. This is an open access article distributed under the Creative Commons Attribution License which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited (CC BY 4.0).