# Programming Plagiarism and Collusion: Student Perceptions and Mitigating Strategies in Indonesia


Oscar Karnalim
Faculty of Information Technology
Maranatha Christian University
Indonesia
oscar.karnalim@it.maranatha.edu

Irwan Alnarus Kautsar
Faculty of Science and Technology
Universitas Muhammadiyah Sidoarjo
Indonesia
irwan@umsida.ac.id

Bayu Rima Aditya
School of Applied Science
Telkom University
Indonesia
bayu@tass.telkomuniversity.ac.id

Yogi Udjaja
Computer Science Department
School of Computer Science
Bina Nusantara University
Indonesia
yogi.udjaja@binus.ac.id

Matahari Bhakti Nendya
Faculty of Information Technology
Duta Wacana Christian University
Indonesia
didanendya@ti.ukdw.ac.id

I Nyoman Darma Kotama
Post Graduate Electrical Engineering
Udayana University
Indonesia
kotama@student.unud.ac.id



***Abstract*— Many strategies have been proposed to mitigate programming plagiarism and collusion. However, the effectiveness can vary across learning environments due to different cultural and/or geographical backgrounds. A few specific studies have been conducted to provide a better picture about the matter in particular countries, but none of them focus on a developing country. This paper proposes a country-specific study for Indonesia, a developing country. Specifically, it summarizes student perceptions of programming plagiarism and collusion on 345 students from 16 universities. This paper also presents a list of strategies for mitigating programming plagiarism and collusion in six universities. Finally, it highlights remaining issues and recommends some practical solutions.**



***Keywords—academic integrity, programming, questionnaire survey, instructor experience, computing education***


## I. INTRODUCTION

In academia, computing students are sometimes unable to complete a programming assessment due to various reasons [1] such as poor time management and lack of appropriate resources. When the students are unable to get sufficient help, they might become desperate and decide to cheat [2], [3]. Typically, they would reuse another student's work. If the act is allowed by the student of the copied work, it is called programming collusion [4]. Otherwise, it is programming plagiarism.

To mitigate programming plagiarism and collusion, students should have neither opportunity, rationalization, nor pressure to cheat [5]. Opportunity can be reduced by introducing a similarity detection tool like JPlag [6] or STRANGE [7]. Rationalization can be dealt by properly informing students about the instructors' expectation of academic integrity in their courses [8]. Pressure can be mitigated by offering many small assessments instead of a few large ones [9].

Before applying any mitigating strategies, it is important to understand student perceptions about programming plagiarism and collusion, what strategies that have been applied, and what are the remaining issues. These can vary across learning environments, especially if they come from different cultural and/or geographical background [10].

A few relevant studies have been conducted to provide a better picture of such phenomena in particular countries. However, they are focused on developed countries (UK [11], China [12], Slovenia [13], and Cyprus [10]). None of them address developing countries, Indonesia in particular. A few studies have focused on the country, but they are about text plagiarism. Two of them are a study reviewing general policies [14] and a study about the use of bibliographic management software [15].

In response to the aforementioned gap, this study has three contributions. First, we report student perceptions about programming plagiarism and collusion in Indonesia. The perceptions were collected via a questionnaire survey, responded by 345 computing undergraduates from 16 Indonesian universities. Second, we summarize some strategies that have been applied in Indonesia to mitigate programming plagiarism and collusion. It is based on the authors' experiences as instructors at six different universities and shared stories from the authors' colleagues about their strategies. Third, based on the first two contributions, we list any remaining issues and suggest some recommendations.

To the best of our knowledge, this is the first study of its type. The study is expected to foster research about programming plagiarism and collusion in Indonesia by summarizing student perceptions and mitigating strategies. Although it is dedicated to a particular country, it might still be relevant and useful for other developing countries, especially those in Asia.

## II. RELATED WORK

Students do plagiarism or collusion since they have the opportunity to do so, they are not able to deal with some pressure, and they can justify their acts via misleading rationalization [5]. In dealing with plagiarism and collusion, at least one of those reasons should not be applicable.

Many studies are focused on reducing the opportunity to cheat. Typical approaches are introducing additional confirmation of the authorship of the work [16], issuing personalized assessments [17], and using a similarity detection tool [18].

Some studies are focused on educating students about programming plagiarism and collusion so that they cannot justify the misbehavior. The instructors are in charge of informing the students about that matter [8]. A few tools have been developed to partly support the process such as an educational mobile application [19] and a Moodle plug-in showing the futility of doing academic misconduct [20].

To mitigate the pressure, it is important to ensure that students are not overwhelmed by the assessments. Issuing smaller assessments [9] or allowing the students to work in pairs [21] are two examples of it.


978-1-6654-3687-8/21/$31.00 ©2021 IEEE

In addition to studies about the mitigating strategies, a number of supporting studies have been conducted. They aim to understand the phenomena of programming plagiarism and collusion in a particular learning environment, summarizing successful strategies, and/or providing practical recommendations. Most of the studies are general such as in [4], [22], and [23]. While these studies are still crucial to foster research in programming plagiarism and collusion, it is important to conduct some country-specific studies given that each country has its own regulations, culture, and geographical background.

We are aware of six country-specific studies, covering four countries. These studies capture general perceptions of students, instructors, and industrial employees regarding programming plagiarism and collusion via questionnaire surveys.

The UK is covered in two studies; one of them focuses on student perspective [11] while another focuses on instructor perspective [24]. The former found that both reusing program from previous assessments and translating program to another language without acknowledgment are wrongly considered as acceptable practices by students. The latter found that there is a grey area about reusing programs as it is encouraged in object-oriented paradigm, a common programming paradigm.

China is also covered in two studies. The first study [12] focuses on students, instructors, and industrial employees. They found that standardized policies about how to treat plagiarism and collusion cases are needed. The second study [25] focuses only on students. Some students were unsure that collusion and reusing code from textbook without acknowledgment are not acceptable.

Slovenia and Cyprus are covered in one study each. The Slovenian study [13] shows that about three fourths of the respondents had been involved in programming plagiarism and collusion at least once. Further, most of the disguises are about identifier renaming. The Cypriot study [10] found that students were not used to acknowledge reused code and they were not aware about self-plagiarism.

## III. METHOD

The study summarizes Indonesian student perceptions of programming plagiarism and collusion. It also lists applied strategies to mitigate programming plagiarism and collusion in Indonesia. Last but not least, it identifies any remaining issues and provides some practical recommendations.

Student perceptions of programming plagiarism and collusion were collected via a questionnaire survey, containing eleven questions from an evaluation instrument of [26]. The questions are originally from [27] but slightly modified to specifically cover plagiarism and collusion. Each question asked student agreement whether a particular scenario is not acceptable since it is considered as programming plagiarism or collusion. All questions should be responded with either “this is an acceptable practice”, “this is NOT an acceptable practice”, or “do not know”. The questions and their expected responses can be seen in Table 1. It is worth noting that the expected responses are defined by the authors based on their teaching experiences; the correct answers to these questions are not universal and they depend on local policies.

For deeper analysis, the survey also asks the students to provide information about their university and their enrollment year. The former lets the student to choose one of seven options. The first six are the universities of the authors while the last one is open-ended where the students can freely write the name of other universities if not listed. The latter lets the students to choose one of four options: year one, year two, year three, and year four or higher. The survey is fully anonymized; we do not ask their personal details like name, gender, and email. Further, we do not provide any incentives for those who fill the survey.

TABLE I. SURVEY QUESTIONS [26] ADAPTED FROM [27]

| ID | Question | Expected Answer |
|---|---|---|
| Q01 | Purchasing code written by others to incorporate into one’s own work | Unacceptable |
| Q02 | Paying another person to write the code and submitting it as one’s own work | Unacceptable |
| Q03 | Basing an assessment largely on work that one wrote and submitted for a previous course, without acknowledging this | Unacceptable |
| Q04 | Incorporating the work of another student without their permission | Unacceptable |
| Q05 | Borrowing another student's code and changing it so that it looks quite different | Unacceptable |
| Q06 | Borrowing an early draft of another student's work and developing it into your own | Unacceptable |
| Q07 | Discussing with another student how to approach a task and what resources to use, then developing the solution independently | Acceptable |
| Q08 | Discussing the detail of one’s code with another student while working on it | Acceptable |
| Q09 | Showing troublesome code to another student and asking them for advice on how to fix it | Acceptable |
| Q10 | Asking another student to take troublesome code and get it working | Unacceptable |
| Q11 | Completing an assessment and then adding features that one noticed when looking at another student's work | Acceptable |

The survey was mainly distributed to students in six universities of the authors. To gain broader perspective, we also asked our students to re-distribute the survey to their colleagues from other universities. Further, we informally contacted some students from other universities as well.

The responses were analyzed by summarizing the proportion of correct responses, counting how many correct responses per question, and observing possible reasons behind each phenomenon. We also checked whether student perceptions of programming plagiarism and collusion are not affected by their universities and their enrollment years via one way ANOVA with 95% confidence rate. If the perceptions are not affected by those two factors, it is more likely that our findings are generalizable across Indonesian universities regardless of enrollment years.

Strategies to mitigate programming plagiarism and collusion in Indonesia were qualitatively summarized mainly based on the authors’ experiences as instructors at six different universities. Some shared stories from the authors’ colleagues about their strategies are also included. For convenience, the strategies would be mapped based on three reasons to cheat: opportunity, rationalization, and pressure [5].

Based on the student perceptions and the mitigating strategies, remaining issues were identified, and some possible recommendations are listed.

## IV. Student Perceptions of Programming Plagiarism and Collusion

Student perceptions was collected via a questionnaire survey, and it was responded by 345 undergraduate students from 16 Indonesian universities. Fig. 1 depicts that in most scenarios, only a small portion of the responses show uncertainty (i.e., responded with "do not know"). Students are somewhat familiar with the scenarios and feel the relevancy. Among the scenarios, Q09 has the largest portion of "acceptable" responses while Q04 has the largest portion of "unacceptable" responses.

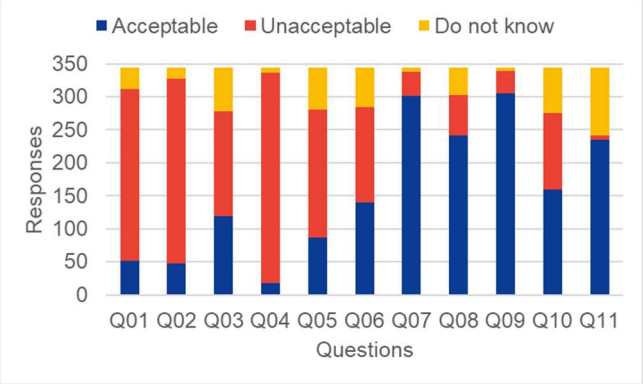


Fig. 1. Student responses distribution regarding the survey of programming plagiarism and collusion

On average, students were able to correctly answer 7 of 11 questions (67% correct response rate). It can be stated that Indonesian students are quite knowledgeable in this matter. However, the score does not necessarily represent unmotivated students since the survey was entirely voluntary and they were unlikely to take part in that.

Table II shows the questions sorted based on their proportion of correct responses. Q04 is correctly answered by most respondents (92%). It is about incorporating another student's work without their permission. Students are aware that asking permission is deemed appropriate before "borrowing" a particular work. Based on the authors' experiences as instructors, the permission is asked even for illegitimate collaboration so that if they are suspected of academic dishonesty, the colleague will help the student in mitigating the penalty.

TABLE II. Survey Questions Sorted Based on Proportion of Correct Responses

| ID | Correct Responses |
|---|---|
| Q04 | 92% |
| Q09 | 88% |
| Q07 | 88% |
| Q02 | 81% |
| Q01 | 76% |
| Q08 | 70% |
| Q11 | 68% |
| Q05 | 56% |
| Q03 | 46% |
| Q06 | 42% |
| Q10 | 34% |

While incorporating another student's work, the student should also ensure that the assessment allows some collaboration and mention the collaboration either in the program (as a comment) or in the documentation. Guideline for citing external resources in a program can be seen in [8].

Q07 and Q09 are two other scenarios that are correctly answered by most respondents (88% for both). Q07 is about discussing with another student how to approach a task and what resources to use, then developing the solution independently. Q09 is about showing troublesome program to another student and asking them for advice on how to fix it. Both are considered as acceptable practices since all of the programs are still written individually and sharing the idea or getting advice is acceptable unless mentioned otherwise. They are common practices among Indonesian students.

Q02 is correctly answered by 81% respondents, followed by Q01 with 76% respondents. Q02 is about paying another student to complete the student's work while Q01 is about purchasing another student's work and incorporating it to the student's work. Both scenarios seldom occur in Indonesia; students are reluctant to allocate some amount of money to complete their work. Further, they are still able to get the copied program without paying any fees as their colleagues sometimes are willing to illegitimately help them. Given the involvement of money, students are aware that these are academic misconduct and can be easily distinguished from legal collaboration.

Q08, which is about discussing the detail of the program with another student while working on it, is only correctly answered by 70% of the respondents. The scenario is somehow similar to Q07 except that the shared information is more detailed. Again, students perceive it as acceptable practice given that all of the programs are still written individually. Further, sharing idea for solving a task is acceptable since the implementation might be different. In Indonesia, this is quite common during lab sessions where students complete a particular assessment together in one place.

Q11, which is about adding features noticed when looking at another student's work, is a scenario with the proportion of correct answers comparable with that of Q08, two percent lower to be precise. It is an acceptable practice since knowing a particular feature does not necessarily entail an ability to implement it. The student still needs to think about how to translate the feature to code and thus how to incorporate the code in their own work. This kind of act is more likely to occur on large assessments (like final projects) where students are somewhat free to choose features for their own programs.

Q05 is correctly responded by half of the respondents (56%). It is about copying another student's work and changing it so that it looks different. Typically, this scenario happens after the student has got permission from the author of the copied program (Q04). Although students are aware that copying another student's work is academic misconduct, the proportion of correct responses is not too high. Some students believe that changing the program is a complex task, justifying their ownership of the changed program. Further, they are not aware that program similarity is often measured at semantic level, ignoring superficial variation like changing comments and variable names. This is quite different with text similarity, where paraphrasing can somehow reduce the degree of similarity.

The remaining scenarios result in less than half of correct responses and thus are in need to be explicitly informed to students. Q03 is about basing an assessment largely on a work that has been previously submitted for a previous course without acknowledging this. Some students think that the scenario is acceptable given that the reused work is their own and they do not harm anyone. Other students accurately perceive it as illegal since the work is used to complete more than one assessment. Without acknowledgment of reuse, the work will be valued as it is written solely for the assessment and that can be unfair for students who do actually write their work only for that purpose. The act is often called self-plagiarism in academia and proper acknowledgment of reuse can be written either in the program or the documentation.

In Indonesia, the scenario is quite common on advanced courses where the assessments are somewhat expanded from those of earlier courses. The reuse can also happen across assessments in a particular course if that course has repetitive topics (e.g., data structure course).

Q06 is about copying an early draft of another student's work and developing it into their own. Some students believe that this is not an academic misconduct since the students still need to put a lot of effort to complete the work prior to submission. However, the issue actually lies on the use of that early draft, not the student's contribution to the work. It should be at least properly acknowledged. In Indonesia, the copied early draft is often obtained from smart and/or motivated students as they are likely to progress faster than others in completing the assessment.

Q10 is about asking another student to take troublesome program and get it working. This is a variation of Q09 except that the student is not in charge of fixing the errors. Two thirds of students misinterpret the scenario as legal collaboration since the program is still mainly written by the student and the scenario seldom changes the program substantially. Moreover, in Indonesia, similar help is sometimes provided by the instructors for students who desperately needs it during computer lab sessions. Regardless, the scenario is only acceptable if the help is accessible for all students or it is acknowledged in the program or the documentation. This is for fairness purpose as some students might write their programs without any help from their colleagues.

As shown in Table III, proportion of correct responses is not much affected by enrollment year. One way ANOVA with 95% confidence rate shows that there are no statistically significant differences among the groups (p-value = 0.15). Hence, it can be stated that student perceptions are not affected by enrollment year.

TABLE III. PROPORTION OF CORRECT RESPONSES PER ENROLLMENT YEAR

| Enrollment Year | Correct Responses |
|---|---|
| Year one | 68% |
| Year two | 68% |
| Year three | 66% |
| Year four or higher | 62% |

Table IV depicts proportion of correct responses across authors' universities (but anonymized as our main goal is not to compare universities). There are no statistically significant differences among the groups according to one way ANOVA with 95% confidence rate (p-value = 0.08). Similar to the enrollment year, student's university does not also affect their perceptions of programming plagiarism and collusion. In other words, it can be stated that our findings reported here are more likely to be generalizable across enrollment years and Indonesian universities.

TABLE IV. PROPORTION OF CORRECT RESPONSES PER AUTHOR UNIVERSITY

| University ID | Correct Responses |
|---|---|
| U1 | 70% |
| U2 | 68% |
| U3 | 68% |
| U4 | 66% |
| U5 | 65% |
| U6 | 63% |

## V. STRATEGIES TO MITIGATE PROGRAMMING PLAGIARISM AND COLLUSION

This section summarizes strategies to mitigate programming plagiarism and collusion in Indonesia based on the authors' experiences as instructors and their colleagues' shared stories.

Many of the strategies are focused on reducing the opportunity to do programming plagiarism and collusion. The most common strategy is to require students to complete the assessment while being monitored by the instructors in a physical classroom (applicable prior to the pandemic). Sometimes, the students are not allowed to connect to the internet, use their data drive, discuss the assessment with other students, and/or choose their seating position. In some universities, students' screens are also monitored via an application like NetSupport.

Another common strategy is to make unique assessments per course offering. Some students are close with their seniors and if the assessments are the same as earlier offerings, they can reuse their seniors' solutions. Given that developing new assessments are labor intensive and time consuming, some instructors modify existing assessments and/or ask students to freely choose their own case studies.

To confirm the authorship of the work, some instructors apply a post-submission test about the program. Students should be able to explain the program otherwise their marks will be deducted. The test can be designed as an interview, an oral presentation, or a written test. On some occasions, the test is only issued for some students, selected at random or based on their likeliness to cheat.

Many instructors manually check for similar programs and if the similarity is a result of programming plagiarism and collusion, the students will be penalized. A few instructors use a code similarity detection tool [7], [28] to expedite the process [18]. Instructors are only required to check programs filtered by the similarity detection tool.

To prevent students rationalizing programming plagiarism and collusion, Indonesian instructors inform the students about the matter and elaborate what kinds of penalties that they would get if caught. Typically, it is delivered verbally at the beginning of the course. Despite its simplicity, it is relatively effective if combined with other strategies.

To mitigate pressure to do plagiarism and collusion, some courses adapt "many small programs approach" [9] where a number of small assessments are issued in replacement of a few large ones. Students are less likely to be overwhelmed by the assessments. Another strategy is to let students complete

the assessments in pairs or groups, which is particularly helpful for slow-paced students [21].

Some instructors allow late submissions to deal with time pressure. Students who cannot complete the assessment in time are still able to get some marks. A few instructors encourage students who have completed their assessment early to help other students. However, this is only applicable on a monitored lab session since otherwise, it can result in illegitimate collaboration.

## VI. Issues and Recommendations

According to student perceptions of programming plagiarism and collusion, students are not particularly aware about three unacceptable practices: basing an assessment largely on a work that has been previously submitted for a previous course without proper acknowledgment (Q03); copying an early draft of another student's work and developing it into their own (Q06); and asking another student to take troublesome program and get it working (Q10).

Apart from explicitly informing students that those scenarios are unacceptable, there is a need to introduce a way to acknowledge that some parts of the code are from external resources, like the one proposed in [8].

Regarding current strategies to mitigate programming plagiarism and collusion, a few of them have some limitations and can be improved.

In terms of reducing the opportunity to cheat, current ways of monitoring students in completing their assessments are less applicable to online learning (which became much common during the pandemic) since that kind of learning involves no physical meetings. We are aware that instructors can require the students to always open their camera during the meeting and/or install a monitoring application. However, these are not really effective as Indonesian internet connection is quite slow and we cannot monitor the whole activities (e.g., students might do something outside the camera view). Instructors are expected to focus more on developing unique assessments and/or applying post-submission tests. Both are relatively effective, and they do not rely on physical meetings.

Manual check for similar programs is another strategy with drawbacks. It is both labor intensive and time consuming. More instructors should use a code similarity detection tool. JPlag [6], MOSS [29], and Sherlock [30] are three commonly mentioned similarity detection tools [31] and they can be alternatives for use. MOSS is less preferred for Indonesian universities as it requires student programs to be uploaded to the server in the US while Indonesian internet connection is quite slow. JPlag and Sherlock do the comparison without internet connection and might be more preferred. JPlag offers more interactive similarity reports whereas some modes in Sherlock can focus on shallower level of similarity.

In case instructors need more comprehensive explanation regarding the reported similarity, they can use STRANGE [7], a dedicated similarity detection tool for that matter with both English and Indonesian as the languages of explanation. A recent study [18] shows that STRANGE is more effective on assessments that are not open to many semantically distinct solutions, like the ones that are typically offered in courses with "many small programs" approach [9] (issuing many small assessments instead of a few large ones).

Although "many small programs" approach reduces pressure to cheat, it has a drawback. Given that each assessment only contributes a little to the final grade, some students might think that it is okay to cheat, especially if the penalty is not severe (e.g., getting the assessment score reduced). It is important to only give students a few chances. For example, if they are caught cheating twice, they would get zero as the final grade.

Last but not least, informing students about programming plagiarism and collusion should be featured with a written document summarizing the matter. If the information is only given verbally, students might forget or mis-remember the information. The document should also include some examples and explain how to cite external code.

## VII. Conclusion and Future Work

This paper summarizes student perceptions of programming plagiarism and collusion in Indonesia. It also lists strategies for mitigating programming plagiarism and collusion. Based on these two, it observes any remaining issues and provide practical recommendations.

In general, students are aware of programming plagiarism and collusion except those that involve self-plagiarism, copying early draft of the work, and asking someone to fix the code. Many strategies have been proposed including monitoring student behavior, informing students about programming plagiarism and collusion, and implementing "many small programs" approach. In response to the findings, it is recommended to introduce a way to cite external code, encourage instructors to use a similarity detection tool, apply severe penalty while implementing "many small programs" approach, and develop a written document informing students about programming plagiarism and collusion.

For future work, we plan to reconduct this study on postgraduate students to observe the consistency of our findings on higher university degrees. We are also interested in performing a comprehensive study comparing our findings with those of similar studies in different countries. It is expected to provide a brief picture of learning environments regarding programming plagiarism and collusion.

## References


[1] J. Sheard, A. Carbone, and M. Dick, "Determination of factors which impact on IT students' propensity to cheat," in *Fifth Australasian conference on Computing education*, 2003, pp. 119–126.

[2] A. Hellas, J. Leinonen, and P. Ihantola, "Plagiarism in take-home exams: help-seeking, collaboration, and systematic cheating," in *22nd Conference on Innovation and Technology in Computer Science Education*, 2017, pp. 238–243, doi: 10.1145/3059009.3059065.

[3] D. Vogts, "Plagiarising of source code by novice programmers a 'cry for help'?," in *Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists*, 2009, pp. 141–149, doi: 10.1145/1632149.1632168.

[4] R. Fraser, "Collaboration, collusion and plagiarism in computer science coursework," *Informatics in Education*, vol. 13, no. 2, pp. 179–195, 2014, doi: 10.15388/infedu.2014.01.

[5] I. Albluwi, "Plagiarism in programming assessments: a systematic review," *ACM Transactions on Computing Education*, vol. 20, no. 1, 2019, doi: 10.1145/3371156.

[6] L. Prechelt, G. Malpohl, and M. Philippsen, "Finding plagiarisms among a set of programs with JPlag," *Journal of Universal Computer Science*, vol. 8, no. 11, pp. 1016–1038, 2002.

[7] O. Karnalim and Simon, "Explanation in code similarity investigation," *IEEE Access*, vol. 9, pp. 59935–59948, 2021, doi: 10.1109/ACCESS.2021.3073703.

[8] Simon, J. Sheard, M. Morgan, A. Petersen, A. Settle, and J. Sinclair, "Informing students about academic integrity in programming," in *20th Australasian Computing Education Conference*, 2018, pp. 113–

122, doi: 10.1145/3160489.3160502.

[9] J. M. Allen, F. Vahid, K. Downey, and A. D. Edgcomb, "Weekly programs in a CS1 class: experiences with auto-graded many-small programs (MSP)," 2018.

[10] G. Cosma *et al.*, "Perceptual comparison of source-code plagiarism within students from UK, China, and South Cyprus higher education institutions," *ACM Transactions on Computing Education*, vol. 17, no. 2, 2017, doi: 10.1145/3059871.

[11] M. Joy, G. Cosma, J. Y.-K. Yau, and J. Sinclair, "Source code plagiarism—a student perspective," *IEEE Transactions on Education*, vol. 54, no. 1, pp. 125–132, 2011, doi: 10.1109/TE.2010.2046664.

[12] L. Yu, H. Jiang, H. Zhu, Q. Zhao, and J. Chen, "Investigating the understanding of plagiarism: a case study of code plagiarism in China," in *15th International Conference on Computer Science Education*, 2020, pp. 176–181, doi: 10.1109/ICCSE49874.2020.9201827.

[13] D. Sraka and B. Kau\uci\uc, "Source code plagiarism," in *31st International Conference on Information Technology Interfaces*, 2009, pp. 461–466, doi: 10.1109/ITI.2009.5196127.

[14] A. Akbar and M. Picard, "Understanding plagiarism in Indonesia from the lens of plagiarism policy: lessons for universities," *International Journal for Educational Integrity*, vol. 15, no. 1, p. 7, 2019, doi: 10.1007/s40979-019-0044-2.

[15] N. Setiani, B. R. Aditya, I. Wijayanto, and A. Wijaya, "Acceptance and usage of bibliographic management software in higher education: the student and teacher point of view," in *IEEE Conference on e-Learning, e-Management and e-Services*, 2020, pp. 55–60, doi: 10.1109/IC3e50159.2020.9288437.

[16] B. Halak and M. El-Hajjar, "Plagiarism detection and prevention techniques in engineering education," in *11th European Workshop on Microelectronics Education*, 2016, pp. 1–3, doi: 10.1109/EWME.2016.7496465.

[17] S. Bradley, "Managing plagiarism in programming assignments with blended assessment and randomisation," in *16th Koli Calling International Conference on Computing Education Research*, 2016, pp. 21–30, doi: 10.1145/2999541.2999560.

[18] O. Karnalim, Simon, M. Ayub, G. Kurniawati, R. A. Nathasya, and M. C. Wijanto, "Work-in-progress: syntactic code similarity detection in strongly directed assessments," in *IEEE Global Engineering Education Conference (EDUCON)*, 2021, pp. 1162–1166, doi: 10.1109/EDUCON46332.2021.9454152.

[19] H. H. Tsang, A. S. Hanbidge, and T. Tin, "Experiential learning through inter-university collaboration research project in academic integrity," in *23rd Western Canadian Conference on Computing Education*, 2018, p. 5, doi: 10.1145/3209635.3209645.

[20] T. Le, A. Carbone, J. Sheard, M. Schuhmacher, M. De Raath, and C. Johnson, "Educating computer programming students about plagiarism through use of a code similarity detection tool," in *Learning and Teaching in Computing and Engineering*, 2013, pp. 98–105, doi: 10.1109/LaTiCE.2013.37.

[21] M. Ayub, O. Karnalim, R. Risal, W. F. Senjaya, and M. C. Wijanto, "Utilising pair programming to enhance the performance of slow-paced students on Introductory Programming," *Journal of Technology and Science Education*, vol. 9, no. 3, pp. 357–367, 2019, doi: 10.3926/JOTSE.638.

[22] Simon *et al.*, "Negotiating the maze of academic integrity in computing education," in *ITiCSE Working Group Reports*, 2016, pp. 57–80, doi: 10.1145/3024906.3024910.

[23] D. Weber-Wulff, "Plagiarism detection software: promises, pitfalls, and practices," in *Handbook of Academic Integrity*, Springer Singapore, 2016, pp. 625–638.

[24] G. Cosma and M. Joy, "Towards a definition of source-code plagiarism," *IEEE Transactions on Education*, vol. 51, no. 2, pp. 195–200, 2008, doi: 10.1109/TE.2007.906776.

[25] D. Zhang, M. Joy, G. Cosma, R. Boyatt, J. Sinclair, and J. Yau, "Source-code plagiarism in universities: a comparative study of student perspectives in China and the UK," *Assessment & Evaluation in Higher Education*, vol. 39, no. 6, pp. 743–758, 2014, doi: 10.1080/02602938.2013.870122.

[26] O. Karnalim, "Automated, Personalised, and Timely Feedback for Awareness of Programming Plagiarism and Collusion," in *17th ACM Conference on International Computing Education Research*, Aug. 2021, pp. 393–394, doi: 10.1145/3446871.3469768.

[27] Simon, B. Cook, J. Sheard, A. Carbone, and C. Johnson, "Academic integrity: differences between computing assessments and essays," in *13th Koli Calling International Conference on Computing Education Research*, 2013, pp. 23–32, doi: 10.1145/2526968.2526971.

[28] C. Kustanto and I. Liem, "Automatic source code plagiarism detection," in *10th ACIS International Conference on Software Engineering, Artificial Intelligences, Networking and Parallel/Distributed Computing*, 2009, pp. 481–486, doi: 10.1109/SNPD.2009.62.

[29] S. Schleimer, D. S. Wilkerson, and A. Aiken, "Winnowing: local algorithms for document fingerprinting," in *ACM International Conference on Management of Data*, 2003, pp. 76–85, doi: 10.1145/872757.872770.

[30] M. Joy and M. Luck, "Plagiarism in programming assignments," *IEEE Transactions on Education*, vol. 42, no. 2, pp. 129–133, 1999, doi: 10.1109/13.762946.

[31] M. Novak, M. Joy, and D. Kermek, "Source-code similarity detection and detection tools used in academia: a systematic review," *ACM Transactions on Computing Education*, vol. 19, no. 3, pp. 27:1--27:37, 2019, doi: 10.1145/3313290.