BG Pulse Journal USA
ISSN:1947-2536 http://www.bgpulseusa.com/index.php/bg

PAIR PROGRAMMING AS A MEANS OF DEVELOPING ALGORITHMIC THINKING OF UNIVERSITY STUDENTS

Hamroyev Bobirjon Bahritdinovich,
Teacher of the Department of Exact Sciences and Digital Technologies, Asia International University
E-mail: khamrayevbb@gmail.com
ORCID: https://orcid.org/0009-0002-3328-1040
doi:https://doi.org/10.5281/zenodo.23385452

Abstract. The article examines pair programming as a form of collaborative learning that contributes to the development of students' algorithmic thinking in programming courses at higher education institutions. The psychological and pedagogical mechanisms through which joint work on a program influences the formation of algorithmic thinking are revealed: verbalization of reasoning, mutual control, and the division of cognitive functions between the roles of "driver" and "navigator". A protocol for organizing a pair programming session in a laboratory class is proposed, principles for forming pairs are described, and typical organizational problems and ways to solve them are analyzed.
Keywords: pair programming, algorithmic thinking, collaborative learning, driver and navigator, verbalization, programming education, laboratory classes, higher education.

INTRODUCTION

Programming courses at higher education institutions are traditionally organized as individual work: each student receives a variant of the task, writes a program alone at the computer, and submits it to the teacher. This form seems natural, since it allows the individual result of each student to be assessed. However, it has significant drawbacks. A student who encounters difficulty at the stage of understanding the problem or choosing an approach to the solution often remains alone with it, waiting for the teacher's help, which is limited in a group of 20–25 people. The student's reasoning remains hidden: neither the teacher nor the student themselves can see at which step the algorithm went wrong.

Algorithmic thinking is formed not only in the process of individual problem solving but also in communication, when a student has to explain their idea to another person, defend the chosen approach, and evaluate someone else's proposal. P. Denning emphasizes that computational thinking should be understood not as a set of isolated skills but as a way of reasoning that is formed through practice in designing and discussing computations [1]. From this point of view, forms of learning that make reasoning about an algorithm the subject of joint discussion acquire particular value.

One of these forms is pair programming — a technique that came to education from the industrial practice of software development, in which two programmers work together at one computer on one task [2]. The purpose of this article is to reveal the potential of pair programming

as a means of developing students' algorithmic thinking and to propose practical recommendations for its organization in laboratory classes in programming courses.

LITERATURE REVIEW

In pair programming, partners perform two roles. The "driver" controls the keyboard and writes the code, while the "navigator" observes the work, checks each line, thinks about the overall strategy, and looks for possible errors; the partners change roles regularly [2]. The technique was initially developed within agile software development methodologies, but in the early 2000s it began to be actively studied in the context of programming education.

A study by C. McDowell and colleagues conducted in an introductory programming course showed that students working in pairs more often completed the course, were more confident in their abilities, and created programs of higher quality than students working individually, while their results on individual exams were not lower [3]. A literature review by B. Hanks and colleagues summarized the experience of using pair programming in education and concluded that it has a positive effect on the quality of programs, students' satisfaction, and their persistence in studying computer science [4]. A systematic review by N. Salleh, E. Mendes, and J. Grundy confirmed these conclusions for higher education and at the same time drew attention to the importance of the compatibility of partners [5]. The meta-analysis by K. Umapathy and A. Ritzhaupt established that pair programming has a positive effect on assignment scores and course completion compared with individual work [6].

At the same time, most of these studies evaluate pair programming by external indicators — grades, completion rates, the quality of programs, and satisfaction. The question of how exactly joint work on a program influences the thinking processes of students and how to organize it so that it purposefully develops algorithmic thinking remains less studied. This determines the focus of the present article.

METHODOLOGY

The study is based on the theoretical analysis of psychological, pedagogical, and methodological literature on pair programming and collaborative learning, the analysis of the cultural-historical theory of the development of higher mental functions, and the generalization of the author's experience of conducting laboratory classes in programming courses at the Department of Exact Sciences and Digital Technologies of Asia International University. Based on this analysis, the mechanisms of the influence of pair programming on algorithmic thinking were identified and a protocol for organizing a pair session was developed.

RESULTS AND DISCUSSION

The analysis made it possible to identify three psychological and pedagogical mechanisms through which pair programming contributes to the development of algorithmic thinking.

The first mechanism is the verbalization of reasoning. Working in a pair, the student cannot simply type code silently: they must explain to the partner what they intend to do and why. The need to express an algorithmic idea in words makes it more precise and conscious. Research by M. Chi and colleagues showed that students who explain the studied material to themselves or others understand it more deeply and build more correct mental models than those who simply read it [7].

In pair programming, such self-explanation becomes a natural part of the work rather than an additional requirement of the teacher.

The second mechanism is the division of cognitive functions. When solving a problem individually, a novice must simultaneously keep in mind the overall plan of the solution, the syntax of the language, and the details of the current line of code, which often leads to errors. In pair programming, these functions are divided: the driver focuses on the implementation of the current step, while the navigator keeps the overall strategy in mind and controls the correctness of the algorithm. Regular role changes allow each student to practice both the strategic and the operational levels of algorithmic activity.

The third mechanism is mutual control and the zone of proximal development. According to L. Vygotsky, higher mental functions first appear in joint activity with another person and only then become the individual's own [8]. In a pair, a student can solve a problem that is still beyond their individual capabilities, relying on the partner's questions and remarks; the partner, in turn, consolidates their own knowledge by explaining it. Errors are detected earlier, at the moment the code is written, and their discussion becomes a source of learning. Social interdependence theory also shows that cooperative work with a common goal and individual responsibility leads to higher achievement than individual or competitive work [9].

In practice, however, these mechanisms do not work automatically. If the work of the pair is not structured, the stronger student often takes over all the work, and the weaker one becomes a passive observer; the navigator may be distracted, and the discussion may turn into simple dictation of code. To prevent this, we propose to conduct a pair session in a laboratory class according to a fixed protocol (Table 1).

Table 1. Protocol of a pair programming session in a laboratory class (80 minutes)

Phase	Time	Content of the work of the pair
1. Joint understanding of the problem	10 min	Partners read the problem, retell it to each other in their own words, agree on the input and output data, and write down two or three examples by hand
2. Agreement on the solution plan	10 min	Partners propose and discuss approaches, choose one, and record the plan as a short list of steps; the code is not yet written
3. Implementation with role changes	35 min	The driver writes the code, the navigator checks it against the plan; roles change every 10–15 minutes at the teacher's signal
4. Joint testing	10 min	Partners check the program on prepared examples and special cases, find and correct errors
5. Individual reflection	15 min	Each student individually writes a short explanation of the algorithm and solves a small related problem without the partner

Source: developed by the author.

The key features of the proposed protocol are the separation of the stages of understanding and planning from the stage of writing code, the mandatory regular change of roles, and the final individual phase. The latter is especially important: it allows the teacher to assess the personal contribution of each student and prevents a situation in which one partner completes the course "at the expense" of the other. To support the navigator, it is useful to give them a short card with questions to ask the driver: "What should this part of the program do?", "What happens if the input is empty or equal to zero?", "Why did you choose this loop?", "How can we check that this step is correct?". Such questions direct the dialogue of the pair toward reasoning about the algorithm rather than toward the syntax of the language.

The effectiveness of pair work largely depends on how the pairs are formed. Based on the literature [5] and our own experience, we recommend the following principles: partners should have a similar, but not necessarily identical, level of preparation, since too large a gap leads to the passivity of the weaker student; pairs should be changed every few weeks so that students learn to work with different partners and do not become dependent on one of them; and when forming pairs, the teacher should take into account students' wishes and psychological compatibility. Typical problems that arise when using pair programming and ways to solve them are summarized in Table

Table 2. Typical problems of pair programming in the classroom and ways to solve them

Problem	Way of solving
One partner dominates, the other is passive	Strict role changes at the teacher's signal; the passive partner is the first to explain the plan; changing the composition of the pair
The navigator is distracted and does not follow the code	A card with guiding questions; the navigator is responsible for the list of test examples
Difficulty in assessing individual contribution	Final individual phase of the session; individual control tasks and oral defense of the program
Conflicts and incompatibility of partners	Taking students' wishes into account when forming pairs; discussing the rules of communication at the first class
Uneven pace of pairs in the group	Additional tasks of increased complexity for pairs that finish early

Source: developed by the author.

It should be noted that pair programming does not replace individual work but complements it. It is most appropriate when studying new topics and solving problems of increased complexity, when joint discussion helps students overcome difficulty, while individual work is needed to consolidate the material and verify each student's independent mastery of it. The combination of these forms within a course, as well as within a single session, allows the advantages of both to be used.

CONCLUSION

Pair programming is an effective means of developing students' algorithmic thinking in programming courses at higher education institutions. Its developmental potential is realized through three mechanisms: the verbalization of reasoning, which makes the algorithmic idea

precise and conscious; the division of cognitive functions between the driver and the navigator, which allows students to practice both strategic and operational levels of algorithmic activity; and mutual control in the zone of proximal development, which allows students to solve problems that are still beyond their individual capabilities.

These mechanisms work only when pair work is properly organized. The proposed protocol of a pair session, which separates understanding and planning from writing code, provides for regular role changes, and ends with individual reflection, together with the principles of forming pairs and ways of solving typical problems, can be used by teachers of programming disciplines in laboratory classes. The prospect of further research is the experimental study of the influence of the proposed protocol on the level of students' algorithmic thinking.

REFERENCES

1. Denning P.J. Remaining trouble spots with computational thinking // Communications of the ACM. – 2017. – Vol. 60, № 6. – P. 33–39. – URL: https://doi.org/10.1145/2998438.
2. Williams L., Kessler R. Pair programming illuminated. – Boston : Addison-Wesley, 2002.
3. McDowell C., Werner L., Bullock H.E., Fernald J. Pair programming improves student retention, confidence, and program quality // Communications of the ACM. – 2006. – Vol. 49, № 8. – P. 90–95. – URL: https://doi.org/10.1145/1145287.1145293.
4. Hanks B., Fitzgerald S., McCauley R., Murphy L., Zander C. Pair programming in education: a literature review // Computer Science Education. – 2011. – Vol. 21, № 2. – P. 135–173. – URL: https://doi.org/10.1080/08993408.2011.579808.
5. Salleh N., Mendes E., Grundy J. Empirical studies of pair programming for CS/SE teaching in higher education: A systematic literature review // IEEE Transactions on Software Engineering. – 2011. – Vol. 37, № 4. – P. 509–525. – URL: https://doi.org/10.1109/TSE.2010.59.
6. Umapathy K., Ritzhaupt A.D. A meta-analysis of pair-programming in computer programming courses: Implications for educational practice // ACM Transactions on Computing Education. – 2017. – Vol. 17, № 4. – Article 16. – URL: https://doi.org/10.1145/2996201.
7. Chi M.T.H., de Leeuw N., Chiu M.-H., LaVancher C. Eliciting self-explanations improves understanding // Cognitive Science. – 1994. – Vol. 18, № 3. – P. 439–477. – URL: https://doi.org/10.1207/s15516709cog1803_3.
8. Vygotsky L.S. Mind in society: The development of higher psychological processes. – Cambridge, MA : Harvard University Press, 1978. – 159 p.
9. Johnson D.W., Johnson R.T. An educational psychology success story: Social interdependence theory and cooperative learning // Educational Researcher. – 2009. – Vol. 38, № 5. – P. 365–379. – URL: https://doi.org/10.3102/0013189X09339057.