Study on The Development of Computational Thinking Decomposition Strategies for Senior Primary Students

Mengtao LI, Yaxin GUAN & Guang CHEN*
Faculty of Education, Beijing Normal University, China
**guang@bnu.edu.cn*

Abstract: As an important component of Computational Thinking, the decomposition strategy has attracted the attention of researchers, but few studies have explored how the decomposition strategy itself is formed and developed. In this study, we carry out thematic-based programming teaching for senior primary school students. The research is divided into four stages: pre-test, exercise, post-test, and migration. We use Microgenetic Design to repeatedly and deeply observe students' programming learning activities in four stages, and conduct interviews and records, so as to analyze the development path and migration of students' decomposition strategies. The research results show that the decomposition ability of senior primary school students can be improved in programming learning, and the overall performance is that advanced strategies gradually replace low-level strategies. However, this process varies from person to person and is not a linear development, and there may be other influencing factors that lead to the return of low-level strategies.

Keywords: Computational Thinking, decomposition strategies, programming, primary students

1. Introduction

In the 21st century, Computational Thinking (CT) has become a prerequisite skill for individuals to cope with a wide range of new challenges and is influencing various fields related to the natural and social sciences. CT has gained worldwide attention as a key literacy for sustaining national competitiveness and has been adopted as a generic skill to be learned by all students in many national initiatives, curricula, standards and other policy documents. Since the introduction of CT, there has been a lot of focus on how to develop this skill (Hsu et al., 2018). In terms of developmental approaches, Hromkovič (2006)and Wing (2006) suggest programming as the main strategy for developing CT in schools. Scherer et al.(2019) also found through meta-analysis that programming is a primary method for developing students' CT. Nevertheless, a large amount of research has focused on the strategies and effects of developing CT, and there is a lack of research on students' strategy development and behaviour during programming.

Based on the above background, this study explored the developmental patterns of cognitive strategies during programming in upper primary school students. Focusing on decomposition strategies, which are the most difficult to be mastered (Selby, 2015) and the most fundamental (Rijke et al., 2018) in CT, we used a Microgeneration Design to describe the developmental pathways and degree of transfer of students' decomposition strategies during programming activities. On the one hand, we focus on the developmental process of decomposition strategies in primary school children's CT; on the other hand, we provide support for future insights into aspects of computational thinking and its internal structure.

2. Literature Review

2. 1 Computational Thinking

CT is a term first used by Papert (1980) to describe the thought process of formulating problems and their solutions in a way that can be executed by a computer. He found that learning with a programming language helped students to simulate the algorithmic logic of a computer, which in turn influenced the way they thought (Papert, 1982).CT is considered to be the psychological basis for problem-solving, designing systems, and understanding human behaviour, and can help people simplify a difficult problem into multiple problems that we know how to solve (Wing, 2006). This process not only prepares students for study in the field of computer science but also provides them with the tools and skills to approach and solve problems in different areas of knowledge (Werner et al., 2012).

2.2 Decomposition strategy

As one of the foundations of CT, decomposition strategies can deconstruct complex problems into multiple easily solvable subproblems(J. Wing, 2008). Therefore, decomposition strategies are crucial for solving complex and large real-world problems(Rijke et al., 2018).

Tsai et al. (2022) argue that decomposition and abstraction are two basic and critical factors required for the development of CT, and suggested that decomposition and abstraction are the starting point for teaching based on the developmental sequence of CT skills. Decomposition is particularly important for the development of CT skills for beginning programmers. Decomposition has always been emphasized as a part of CT as a whole, but few studies have revealed specific details about decomposition itself(Rich et al., 2019). Although researchers have defined four levels of strategies from 0-3 in terms of assessment criteria for their development, with different conceptual and descriptive definitions(Avila et al., 2019), it does not explain how students' Disaggregation strategies develop in the process of CT development and lacks depth and guidance for practice.

2.3 Visual programming

Visualisation means expressing abstract content in an intuitive way(Rijke et al., 2018). Unlike textual programming such as Python and Java, visual programming tools provide students with the opportunity to learn computer programming concepts through visual feedback-based learning, reducing the cognitive load placed on students by complex programming grammar (Lye & Koh, 2014), and allowing students to focus on the logic and structure involved in programming rather than worrying about the mechanics of writing programs(J. M. Wing, 2006). Numerous studies have shown that it makes sense to develop decomposition strategies from the elementary school level: on the one hand, programming develops CT(Buitrago Flórez et al., 2017). On the other hand, children already have the cognitive level to understand basic computer programming concepts by the age of four (Bers et al., 2014).

During programming, students are exposed to CT, using computer science concepts such as abstraction, decomposition, and generalization to solve problems. Programming education should focus on how students can break down and categorize complex problems to solve them efficiently (Sáez-López et al., 2016). The Block-coding programming tool used in this study is the Discovery Education Coding Curriculum (DECC), which supports students in allocating limited cognitive resources to the mobilization of problem-solving strategies for students who are new to programming.

2.4 Research questions

Piaget believes that children's cognitive strategy development pattern has four stages from low to high. In CT development, there is variability in the development of different elements in the four stages. Even if they have been trained by teachers in advanced strategies for each element, they will be in the no-strategy stage or the low-strategy stage and will not be able to

use advanced strategies comfortably due to the lack of cognitive resources. Focusing on CT can help students be able to select and apply appropriate strategies and tools to better conceptualize, analyze, and solve complex problems. However, most research has focused on issues related to learning skills, concepts and practices (Ezeamuzie & Leung, 2022; Grover & Pea, 2013), and the specific processes involved in the development of students' CT need to be further explored. This study focuses on one of the most fundamental and critical starting points in the development of CT: decomposition strategies. By implementing instructional activities that facilitate the development of CT decomposition strategies in elementary classrooms and tracking students' behaviors and characteristics in the process, this study proposes to address the following questions:

RQ1: How well do senior primary students use decomposition strategies at different stages?

RQ2: What are the developmental pathways of decomposition strategies for students in senior primary grades at different stages of the same programming activity?

RQ3: To what extent do senior primary school students transfer different strategies within the same programming activity?

3. Research Methodology

We explored the development of CT decomposition strategies and pathways in primary school students: a micro-incidence design was used to collect and code relevant data through classroom behavioural observations and post-classroom interviews.

3.1 Subjects of study

The subjects of the study were 5th-grade students in a primary school in Beijing. Through pre-experimental observations, interviews and data analyses, 18 subjects were finally identified. During the implementation of the activity, 4 students did not participate in the complete experimental process for personal reasons, so 14 valid data were received after the teaching activity. The sample of 14 consisted of 4 boys and 10 girls, all aged 9-11 years (M = 10.10, SD = 0.59).

3.2 Study design

Microgenetic Design is a research method to study the cognitive and behavioral development process of children, which is widely used in the research of cognitive strategy development of young children to high school students (Siegler & Crowley, 1991). Through iterative and in-depth observation and analysis of changes in behavior throughout the process, Microgenetic Design can provide detailed information about the sources, pathways, velocity, and extent of cognitive changes, as well as the diversity of change patterns. There are several reasons:(1) observations cover the entire interval from the onset of change to relative stability; (2) the frequency of observations is highly consistent with the rate of change of the phenomenon; and (3) the observed behavior is finely iteratively analyzed in iterative experiments to facilitate speculation on the processes that produce qualitative and quantitative changes. In short, the high density of fine-grained data provides a guarantee of the validity of this approach.

Traditional pre- and post-test experiments have a low density of observation of the process of behavioural change throughout the study and do not allow for a sophisticated analysis of strategy development behaviour during this process. This can be well avoided by using a Microgenetic Design to observe students' behaviour and thinking in the classroom during the course of the lesson. When episodic behaviours were more ambiguous or absent, a categorical basis for strategy assessment was obtained by promptly asking students how they designed the procedure.

3.3 Research process

The experiment consisted of two parts, a pre-experiment and a formal experiment. All six sessions were taught by the same researcher and lasted two and a half weeks, with each session lasting 40 minutes. The six class periods were centred around six thematic activities that were taught as CT activities: Movement, Simple Input, Different Kinds of Input, Buttons & Commands, Sequence & Animation and Conditional Events. Each theme has three different programming activities under it, and students need to complete these three programming activities and perform a programming challenge in each lesson, which together constitute the pre-test, practice, post-test, and migration phases of the experiment.

In the pre-experiment, Topic Activity 1 was used to screen students with zero foundation. After completing each task, students were tested on their decomposition skills.

All 5 thematic activities in the formal experiment consist of 3 learning tasks, which are used to assess students' CT strategy development in the pre-test, practice, and post-test phases. In addition to this, students were required to undertake the completion of a programming project as a migration test.

3.4 Data collection

Data were collected through classroom observations and interviews. Classroom observations were conducted to track and record student behaviour and development in the classroom. Interviews were conducted in the form of questions asked by the teacher and shared by the students in the classroom and were used for further analysis and refinement. Interview outlines were prepared by the researcher based on the literature and what was focused on in this study, and the interviews included students' thoughts on programming as well as their feelings about learning during the four phases.

Data were coded according to the decomposition strategy level coding table. Based on a definition of decomposition strategy development levels proposed by Avila et al. (2019), we took into account specific programming situations as well as theories related to cognitive strategies and invited two researchers in the field of CT to classify students' decomposition strategies into four levels. The Decomposition Strategy Hierarchy scale is shown in the appendix.

4. Research findings

The results of the study will depict the change routes and development of decomposition strategies in senior primary school students during the completion of programming tasks in the following three ways.

4.1 Students' overall use of decomposition strategies at different stages

As shown in Figure 1, the Roma numerals from small to large represent the four stages of pre-test, practice, post-test, and migration. In the pre-test stage, students' main decomposition strategy is the level 0 strategy, and they hardly use the advanced strategy; in the practice stage, level 1 strategies are the students' main strategies, and level 3 strategies begin to appear; in the post-test stage, level 2 and level 3 strategies become the main strategies, and level 0 strategies have been completely replaced; in the migration stage, the use of level 3 strategies reaches 50%, and the use of level 2 strategies is as high as 45%.

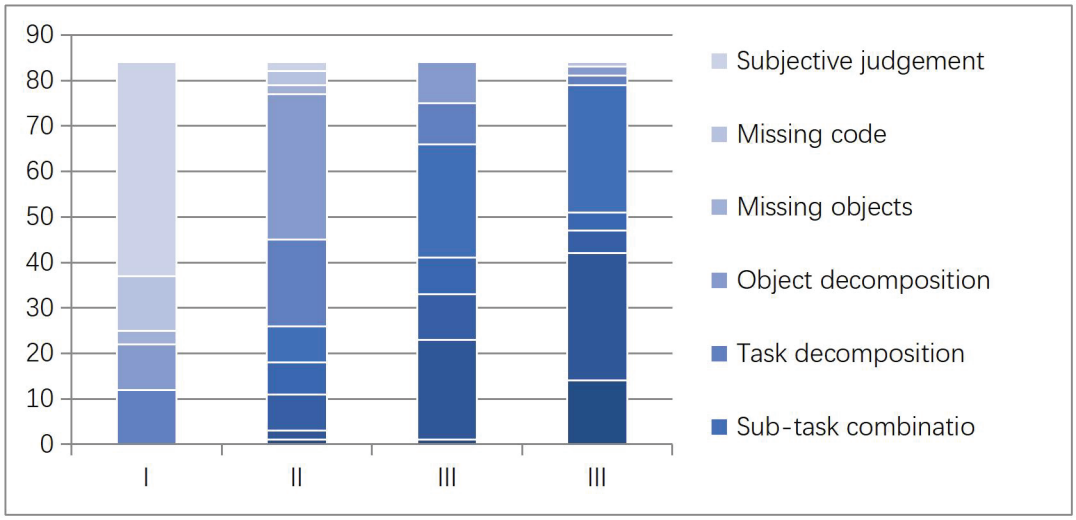

Figure 1. The use of decomposition strategies at different stages.

4.2 *Development of students' decomposition strategies in programming activities Pathways*

As shown in Figure 2, the line from light blue to dark blue represents the four-level decomposition strategy from low to advanced (e.g. the lightest blue line represents the 0-level decomposition strategy). The route analysis of the strategies adopted in four different stages of the same programming activity shows that the development trend of different levels of decomposition strategies in different thematic activities is basically the same, which is manifested as the gradual replacement of low-level strategies by high-level strategies.

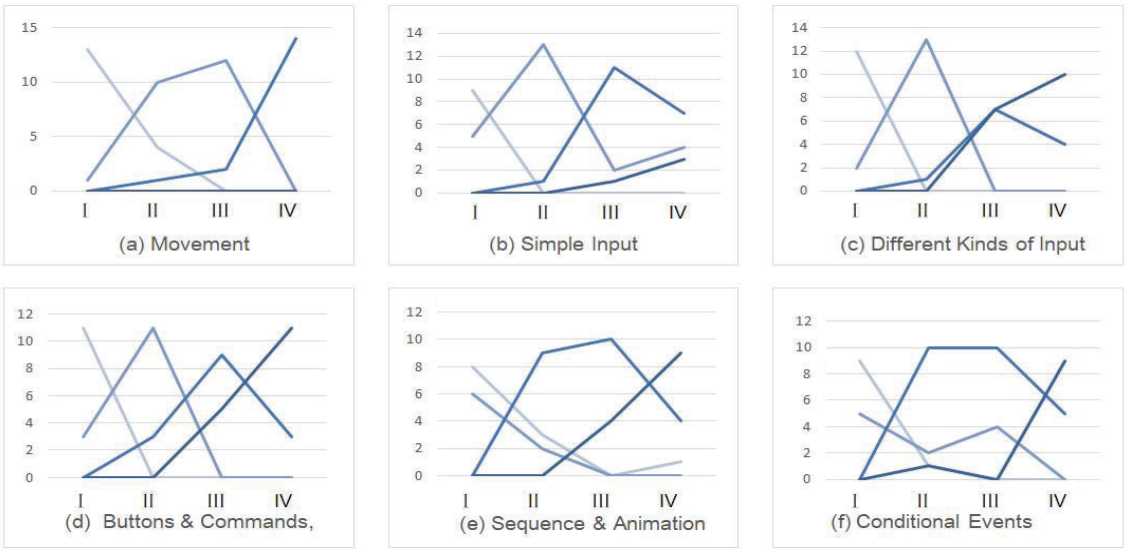

Figure 2. The overall line of development of the decomposition strategy under different thematic activities.

4.3 *Transfer of students' decomposition strategies in programming activities extent*

Figure 3 shows a roadmap for individuals to adopt strategies at different stages of programming activity. ① represents the level, and the number on the horizontal line between the two levels represents the number of people whose route changes between the two strategy levels. The

results show that before being taught programming, all students initially used only low-level decomposition strategies, and the majority of students used zero-level strategies. In most cases, the decomposition strategies used by students follow a continuous development from low-level to high-level strategies, but there will be a brief regression from high-level strategies to low-level strategies.

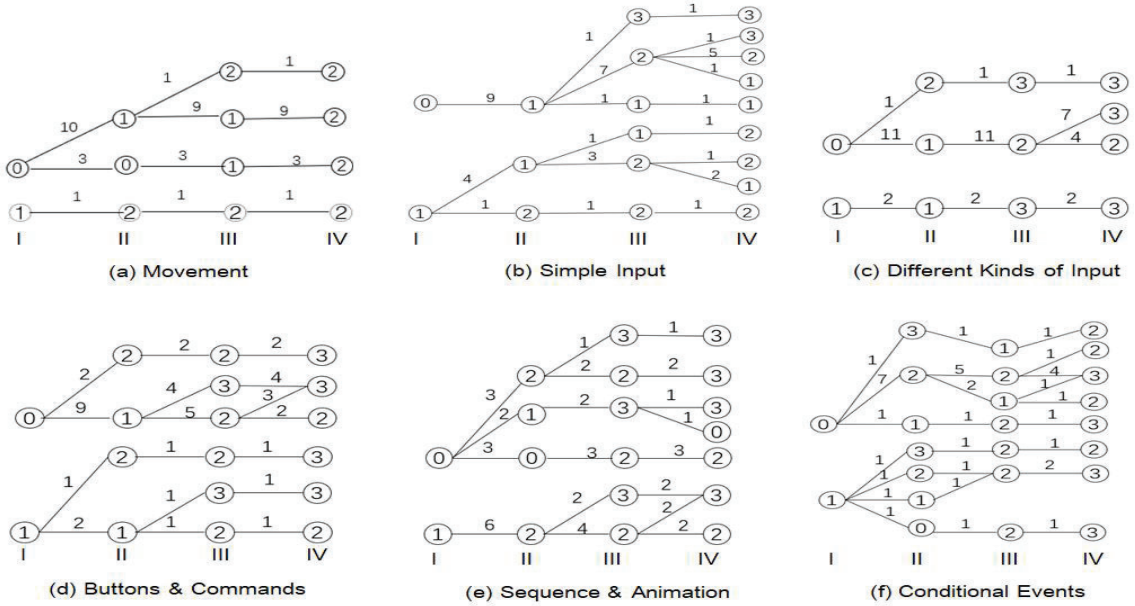

Figure 3. The degree of individual migration of strategies in different activities.

5. Discussion and conclusions

The purpose of this study is to investigate the developmental pathways and strategy transfer of CT decomposition strategies of students in the upper primary grades, and we can summarise three main findings. Firstly, students' decomposition strategies can be developed through topic-based programming learning, and programming instruction in the upper primary grades can be an effective way to develop students' decomposition skills. This result reaffirms Bers et al.(2014)and Rijke et al.(2018) that it is feasible and meaningful to develop students' decomposition skills by teaching programming from the primary level. Secondly, students' decomposition strategies are not linear but a dynamic process of development, and there may be other factors that lead to the recurrence of low-level strategies. This result is similar to the findings of Siegler & Stern(1998) on the development of mathematical strategies in second-graders. In addition, students' decomposition strategy development is a complex process, and students' own experiences, teachers' interventions, and differences in instructional content may affect the developmental trends and the degree of transfer of students' decomposition strategies. In the process of guiding students to learn programming, teachers need to provide effective guidance to help students decompose problems and tasks. At the same time, through the creation of specific problem situations and practice opportunities, teachers can transfer the decomposition strategy teaching content to other disciplines and make effective adjustments to the curriculum design, so as to improve students' decomposition strategies in multiple ways.

Preference

Bers, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers & Education*, *72*, 145–157. https://doi.org/10.1016/j.compedu.2013.10.020

Buitrago Flórez, F., Casallas, R., Hernández, M., Reyes, A., Restrepo, S., & Danies, G. (2017). Changing a Generation's Way of Thinking: Teaching Computational Thinking Through Programming. *Review of Educational Research*, *87*(4), 834–860. https://doi.org/10.3102/0034654317710096

Ezeamuzie, N. O., & Leung, J. S. C. (2022). Computational Thinking Through an Empirical Lens: A Systematic Review of Literature. *Journal of Educational Computing Research*, *60*(2), 481–511. https://doi.org/10.1177/07356331211033158

Grover, S., & Pea, R. (2013). Computational Thinking in K–12: A Review of the State of the Field. *Educational Researcher*, *42*(1), 38–43. https://doi.org/10.3102/0013189X12463051

Hromkovič, J. (2006). Contributing to General Education by Teaching Informatics. In R. T. Mittermeir (Ed.), *Informatics Education – The Bridge between Using and Understanding Computers* (Vol. 4226, pp. 25–37). Springer Berlin Heidelberg. https://doi.org/10.1007/11915355_3

Hsu, T.-C., Chang, S.-C., & Hung, Y.-T. (2018). How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Computers & Education*, *126*, 296–310. https://doi.org/10.1016/j.compedu.2018.07.004

Kafai, Y. B., & Proctor, C. (2022). A Revaluation of Computational Thinking in K–12 Education: Moving Toward Computational Literacies. *Educational Researcher*, *51*(2), 146–151. https://doi.org/10.3102/0013189X211057904

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, *41*, 51–61. https://doi.org/10.1016/j.chb.2014.09.012

Otero Avila, C., Foss, L., Bordini, A., Simone Debacco, M., & Da Costa Cavalheiro, S. A. (2019). Evaluation Rubric for Computational Thinking Concepts. *2019 IEEE 19th International Conference on Advanced Learning Technologies (ICALT)*, 279–281. https://doi.org/10.1109/ICALT.2019.00089

Papert, S. (1982). *Mindstorms: Children, computers, and powerful ideas* (Reprint). Harvester Press.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. https://cn.bing.com/academic/profile?id=d041c558229147ac04a3b561314ae782&encoded=0&v=paper_preview&mkt=zh-cn

Rich, P. J., Egan, G., & Ellsworth, J. (2019). A Framework for Decomposition in Computational Thinking. *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*, 416–421. https://doi.org/10.1145/3304221.3319793

Rijke, W. J., Bollen, L., Eysink, T. H. S., & Tolboom, J. L. J. (2018). Computational Thinking in Primary School: An Examination of Abstraction and Decomposition in Different Age Groups. *Informatics in Education*, *17*(1), 77–92. https://doi.org/10.15388/infedu.2018.05

Sáez-López, J.-M., Román-González, M., & Vázquez-Cano, E. (2016). Visual programming languages integrated across the curriculum in elementary school: A two year case study using "Scratch" in five schools. *Computers & Education*, *97*, 129–141. https://doi.org/10.1016/j.compedu.2016.03.003

Scherer, R., Siddiq, F., & Sánchez Viveros, B. (2019). The cognitive benefits of learning computer programming: A meta-analysis of transfer effects. *Journal of Educational Psychology*, *111*(5), 764–792. https://doi.org/10.1037/edu0000314

Selby, C. C. (2015). Relationships: Computational thinking, pedagogy of programming, and Bloom's Taxonomy. *Proceedings of the Workshop in Primary and Secondary Computing Education*, 80–87. https://doi.org/10.1145/2818314.2818315

Siegler, R. S., & Stern, E. (1998). Conscious and unconscious strategy discoveries: A microgenetic analysis. *Journal of Experimental Psychology: General*, *127*(4), 377–397. https://doi.org/10.1037/0096-3445.127.4.377

Tsai, M.-J., Liang, J.-C., Lee, S. W.-Y., & Hsu, C.-Y. (2022). Structural Validation for the Developmental Model of Computational Thinking. *Journal of Educational Computing Research*, *60*(1), 56–73. https://doi.org/10.1177/07356331211017794

Werner, L., Denner, J., Campe, S., & Kawamoto, D. C. (2012). The fairy performance assessment: Measuring computational thinking in middle school. *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, 215–220. https://doi.org/10.1145/2157136.2157200

Wing, J. (2008). Computational thinking and thinking about computing. *2008 IEEE International Symposium on Parallel and Distributed Processing*, 1–1. https://doi.org/10.1109/IPDPS.2008.4536091

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33–35. https://doi.org/10.1145/1118178.1118215

Appendix

Table 1. Decomposition Strategy Hierarchy

Grade	Definition	Specific description
Level 0 Strategy	During the analysis of the programming task, programming relies on subjective judgement and may not achieve the programming objectives.	**Subjective judgement:** the process of identifying the task problem and thus completing the programming by virtue of an autonomous understanding of the programming task, in which errors may occur. **Missing objects:** Objects needed in the programming are not added to the scene. Resulting in missing code. **Missing code:** Missing code in the programming process due to a lack of systematic decomposition of programming tasks.
Level 1 Strategies	Objective decomposition of programming tasks based on subjective judgement, using combinations of existing code to complete programming tasks.	**Task decomposition:** the ability to decompose the main subtasks of a programming task. **Object decomposition:** the ability to program action codes for each object based on the objects that appear in the programming.
Level 2 Strategies	Be able to completely analyse the objectives of a task in programming, analyse the relationships between sub-problems and use combinations of previously learned code blocks to complete programming tasks	**Sub-task combination: t**he ability to break down the overall goal of a programming task into several task blocks and combine them using existing code to complete the programming. **Sequential decomposition:** problem decomposition in the order of the problems that can be programmed for a programming task. **Conditional decomposition: t**he ability to decompose programming tasks according to the different situations of the programming problem.
Level 3 Strategies	The process of decomposing task objectives enables a systematic breakdown of programming objectives into smaller problems in a combination of process and content, which are effectively combined to complete the programming task.	**Autonomous decomposition:** the ability to autonomously identify the main problems in a programming task of their own design and the relationships between the various sub-problems. **Optimal sub-problem combination:** The ability to form an optimal combination of sub-problems in programming into an efficient solution to the entire programming task.