Computational Thinking

Subjects: Computer Science, Artificial Intelligence

Contributor: Changqin Huang

Computational Thinking (CT) has been widely regarded as an essential ability to solve problems by applying basic knowledge of computer science in technological societies. Initially, CT was defined as using the fundamental concepts of computer science to solve problems, design systems, and understand human behaviors.

Keywords: computational thinking ; critical reflection ; secondary school students ; interactions

1. Introduction

Meanwhile, some operational definitions of CT have been proposed. For example, the CSTA & ISTE put forward that CT is a particular set of problem-solving skills that includes identifying the problem, organizing and analyzing data, representing the data, developing automated solutions, implementing and evaluating the optimal solution, and generalizing and transforming solutions [1]. Adding to this, Brennan and Resnick proposed the three-dimensional framework of CT. This framework argues that students' CT should be developed and evaluated in three dimensions: (1) computational concepts (the concepts that students often use in programming, such as sequences, loops, and events); (2) computational practices (the practices that students develop when they engage with computational concepts, such as iterating, debugging, and abstracting); and (3) computational perspectives (the perspectives students form about themselves and about the world, such as recognizing that computation is a medium of creation, realizing the power of working with others, and feeling empowered to ask questions) [2]. This integrated framework has been used frequently to cultivate K–12 students' CT, because it not only emphasizes the conceptual knowledge and practical skills of CT but also attaches importance to the social attribute of CT [3][4].

To sum up, we draw three conclusions. First, CT is an important ability to solve problems. Second, generalizing solutions is an integral part of learning CT. Third, students need not only to learn computational concepts and computational practices but also to develop computational perspectives.

Integrating CT into Secondary School Education

In the past years, CT has been integrated into secondary school education to build a solid foundation for students' future success and to support the sustainable development of our computer science-driven world [5][6][7]. Consequently, various learning strategies have been used to develop students' CT. In general, there are two types of learning strategies for developing CT with secondary school students.

The first and most popular is student-centered learning. Motivated by constructionism [8], researchers have used the design-based learning strategy to develop CT in eighth-grade students [9]. Students self-explored the applications of CT by engaging in multiple cycles of design, evaluation, and redesign to make computer games about science topics iteratively in a programming environment. The results showed that design-based learning activities enable students to master computational concepts and computational practices. Along similar lines, a pedagogical strategy based on agile software engineering methods has been adopted to develop secondary school students' CT [10]. This strategy allows students to explore, iterate, and experience computational practices in different settings. The results showed that students' CT was enhanced by exploring these multidisciplinary activities. Moreover, collaborative learning strategies have been integrated into student-centered learning activities. A three-year research project claimed that co-designing mobile games about social change could advance secondary school students' understanding of computational concepts and computational practices [11].

The second is teacher-directed learning. For instance, Saritepeci conducted a study that explored the effect of completing programming tasks on CT development with ninth-grade students [12]. The teacher introduced computational concepts and sample problem-solving activities to the students. Then, the teacher directed the students to collaborate on computational practices. The results showed that the students' CT had been significantly improved, while interacting with others had a positive effect on developing these students' computational perspectives about themselves and their relationships with others. Moreover, teacher-directed learning has also been used to develop interdisciplinary CT in ninth-grade students [13]. In this pedagogical process, the teacher demonstrated examples, guided the students to discuss, and provided the students with a series of pre-designed problems. As a result, the students' CT and programming skills were improved.

Overall, gaining computational experiences through self-practice or observation has been a main learning activity in cultivating CT with secondary school students. These cognitive experiences not only help students understand computational concepts and computational practices, but also shape students' computational perspectives [2].

Meanwhile, many studies have been conducted to explore the possible means of assessing CT with students. In general, there are four ways of assessing CT. The first popular way is works analysis. For example, a visualization tool named Scrape (http://happyanalyzing.com/ accessed on 9 October 2021) has been developed and used to automatically present the computational concepts used within Scratch projects [2]. Moreover, there are researchers who analyze works manually from the four aspects of content, creativity, artistry, and technology to evaluate students' learning performance in computational concepts and computational practices [4]. The second way is using traditional quizzes, such as multiple-choice items, to evaluate students' learning performance in computational concepts and computational practices [14]. The third way is conducting artifact-based interviews or self-reports to assess computational concepts, computational practices, and computational perspectives [2]. The fourth way is using scales. For instance, a five-point Likert scale was developed by Korkmaz et al. [15]. Computational thinkers use this scale to evaluate their creativity, algorithmic thinking, cooperation, critical thinking, and problem solving.

However, generalizing and transferring CT to other contexts remains a challenge for secondary school students [2][11]. In addition, more attention should be paid to the ways of developing K–12 students' computational perspectives [1][16].

2. Learning CT through Critical Reflection

In the context of thinking, critical reflection is defined as a process of thinking about the conditions and the effects of what a person is doing or has done [17]. It reveals the influence and function of thought and action on people [18]. To achieve critical reflection, students need to complete the following steps. First, students need to examine noticeable details of the problem-solving process. Then, they are required to judge the reasons for their decisions from different perspectives. Following this, they re-cast prior experiences and knowledge into other contexts. Finally, they should develop new plans for solving similar problems in the future [19][20][21]. Therefore, researchers believe that a statement can be measured as critical reflection only when it shifts from a description of events to a critical report that analyzes, integrates and reconstructs experiences, and ultimately produces a new perspective [22][23].

In the domain of education, the theory of experiential learning recognizes critical reflection as an essential metacognitive strategy for acquiring meaningful learning outcomes from specific experiences [24][25]. Researchers believe that metacognition refers to thinking about one's own thinking, which is a crucial component in controlling and regulating one's thinking, especially problem-solving [24][26][27]. If problem-solvers can be aware of their cognition and can use this awareness to control and regulate their problem-solving process, they will have a better chance of success [28].

In recent years, some researchers have discussed the relationship between CT and critical reflection. They argued that CT is exactly a particular set of problem-solving abilities, while critical reflection could improve students' problem-solving abilities [29][30][31][26]. Moreover, generalizing solutions and forming computational perspectives are essential parts of CT, while critical reflection features prominently in promoting the generalization of solutions and the formation of new perspectives [1][32].

In view of these arguments, critical reflection has been employed as a metacognitive strategy for developing CT-related knowledge in higher education. For instance, university students working in programming and multimedia systems development were required to blog about their critical reflections on their learning process. The students were prompted to examine what problems they have encountered, how they solved the problem, why the solution worked or not, and what they should do next time. The results showed that these activities could help the participants develop computational practices, such as finding problems and making revisions [33]. Moreover, Kyungbin and Jonassen's study showed that critical reflection had positive effects on helping university students understand computational concepts and complete computational practices in the domain of programming [31]. In their study, participants were asked to explain and critically judge their decisions. Similarly, Miller et al. integrated critical reflection into university students' CT learning activities [34]. The students were provided with critical reflection prompts to think about their CT learning activities at a more abstract level. The results confirmed that critical reflection enabled the students to transfer CT to more general problem-solving contexts.

However, whether and how critical reflection can improve CT with secondary school students remains to be explored. At present, only noncritical reflection has been integrated in the CT learning activities of secondary school students. These noncritical reflections stated the noticeable details of the problem-solving process but did not evaluate the solutions. For instance, Zhong et al. stated that reflective card, which was designed for students to report their noticeable errors after they finished computational practices, was a useful tool for helping teachers discover students' learning barriers [4]. Similarly, reflective journals have been adopted to reveal secondary school students' perspectives on CT training activities and problems encountered [11]. In particular, the students were assigned to individually report the successes and difficulties they faced, as well as the activities they enjoyed and disliked. Although these noncritical reflections benefit teachers in detecting students' immediate and present learning difficulties, they have a trivial effect on activating and enhancing metacognition, which is essential for controlling and regulating learners' thinking [26][35].

References

1. CSTA & ISTE. Operational Definition of Computational Thinking for K-12 Education. Available online: https://c.ymcdn.com/sites/www.csteachers.org/resource/resmgr/CompThinkingFlyer.pdf (accessed on 15 August 2020).
2. Brennan, K.; Resnick, M. New frameworks for studying and assessing the development of computational thinking. In Proceedings of the 2012 Annual Meeting of the American Educational Research Association, Vancouver, BC, Canada, 13–17 April 2012; p. 25.
3. Lye, S.Y.; Koh, J.H.L. Review on teaching and learning of computational thinking through programming: What is next for K−12? Comput. Hum. Behav. 2014, 41, 51–61.
4. Zhong, B.; Wang, Q.; Chen, J.; Li, Y. An exploration of three-dimensional integrated assessment for computational thinking. J. Educ. Comput. Res. 2016, 53, 562–590.
5. Ardito, G.; Czerkawski, B.; Scollins, L. Learning computational thinking together: Effects of gender differences in collaborative middle school robotics program. TechTrends 2020, 64, 373–387.
6. Hsu, T.; Chang, S.; Hung, Y. How to learn and how to teach computational thinking: Suggestions based on a review of the literature. Comput. Educ. 2018, 126, 296–310.
7. Jong, M.S.; Geng, J.; Chai, C.S.; Lin, P. Development and predictive validity of the computational thinking disposition questionnaire. Sustainability 2020, 12, 4459.
8. Papert, S.A. Mindstorms. Children, Computers and Powerful Ideas; Basic Books: New York, NY, USA, 1980.
9. Tucker-Raymond, E.; Puttick, G.; Cassidy, M.; Harteveld, C.; Troiano, G.M. "I Broke Your Game!": Critique among middle schoolers designing computer games about climate change. Int. J. STEM Educ. 2019, 6.
10. Fronza, I.; Ioini, N.E.; Corral, L. Teaching computational thinking using agile software engineering methods: A framework for middle schools. ACM Trans. Comput. Educ. 2017, 17, 1–28.
11. Thomas, J.O.; Rankin, Y.; Minor, R.; Sun, L. Exploring the difficulties African-American middle school girls face enacting computational algorithmic thinking over three years while designing games for social change. Comput. Support. Coop. Work 2017, 26, 389–421.
12. Saritepeci, M. Developing computational thinking skills of high school students: Design-based learning activities and programming tasks. Asia-Pac. Educ. Res. 2020, 29, 35–54.
13. Settle, A.; Franke, B.; Hansen, R.; Spaltro, F.; Jurisson, C.; Rennert-May, C.; Wildeman, B. Infusing computational thinking into the middle- and high-school curriculum. In Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education, New York, NY, USA, 3–5 July 2012.
14. Grover, S. Assessing algorithmic and computational thinking in K-12: Lessons from a middle school classroom. In Emerging Research, Practice, and Policy on Computational Thinking. Educational Communications and Technology: Issues and Innovations; Rich, P., Hodges, C., Eds.; Springer: Cham, Switzerland, 2017.
15. Korkmaz, Ö.; Çakir, R.; Özden, M.Y. A validity and reliability study of the computational thinking scales (CTS). Comput. Hum. Behav. 2017, 72, 558–569.
16. Yaşar, O. A new perspective on computational thinking. Commun. ACM 2018, 61, 33–39.
17. Steier, F. Reflexivity and methodology: An Ecological Constructionism; Sage: Thousand Oaks, CA, USA, 1991.

18. Birch, M.; Miller, T. Inviting intimacy: The interview as therapeutic opportunity. Int. J. Soc. Res. Methodol. 2000, 3, 189–202.

19. Chi, F. Turning experiences into critical reflections: Examples from Taiwanese in-service teachers. Asia-Pac. J. Teach. Educ. 2013, 41, 28–40.

20. Matsuo, M. Goal orientation, critical reflection, and unlearning: An individual-level study. Hum. Resour. Dev. Q. 2018, 29, 49–66.

21. Williams, B. Developing critical reflection for professional practice through problem-based learning. J. Adv. Nurs. 2001, 34, 27–34.

22. Gibson, A.; Aitken, A.; Sándor, A.; Shum, S.B.; Tsingos-Lucas, C.; Knight, S. Reflective writing analytics for actionable feedback. In Proceedings of the 7th International Learning Analytics & Knowledge Conference, Vancouver, BC, Canada, 13–17 March 2017.

23. Kember, D.; McKay, J.; Sinclair, K.; Wong, F.K.Y. A four-category scheme for coding and assessing the level of reflection in written work. Assess. Eval. High. Educ. 2008, 33, 369–379.

24. Colbert, C.Y.; Graham, L.; West, C.; White, B.A.; Arroliga, A.C.; Myers, J.D.; Ogden, P.E.; Archer, J.; Mohammad, Z.T.A.; Clark, J. Teaching metacognitive skills: Helping your physician trainees in the quest to ‘know what they don’t know’. Am. J. Med. 2015, 128, 318–324.

25. Medina, M.S.; Castleberry, A.N.; Persky, A.M. Strategies for improving learner metacognition in health professional education. Am. J. Pharm. Educ. 2017, 81, 78.

26. Lee, C.B.; Koh, N.K.; Cai, X.L.; Quek, C.L. Children’s use of meta-cognition in solving everyday problems: Children’s monetary decision-making. Aust. J. Educ. 2012, 56, 22–39.

27. Flavell, J.H. Metacognition and cognitive monitoring: A new area of cognitive–developmental inquiry. Am. Psychol. 1979, 34, 906–911.

28. Fleming, S.M.; Dolan, R.J.; Frith, C.D. Metacognition: Computation, biology and function. Philos. Trans. R. Soc. Lond. 2012, 367, 1280–1286.

29. Wing, J.M. Computational thinking. Commun. ACM 2006, 49, 33–35.

30. Boud, D.; Keogh, R.; Walker, D. Reflection: Turning Experience into Learning, 1st ed.; Routledge: Oxfordshire, UK, 1985.

31. Kwon, K.; Jonassen, D.H. The influence of reflective self-explanations on problem-solving performance. J. Educ. Comput. Res. 2011, 44, 247–263.

32. Howie, P.; Bagnall, R. A beautiful metaphor: Transformative learning theory. Int. J. Lifelong Educ. 2013, 32, 816–836.

33. Robertson, J. The educational affordances of blogs for self-directed learning. Comput. Educ. 2011, 57, 1628–1644.

34. Miller, L.D.; Soh, L.; Chiriacescu, V.; Ingraham, E.; Shell, D.F.; Hazley, M.P. Integrating computational and creative thinking to improve learning and performance in CS1. In Proceedings of the 45th ACM Technical Symposium on Computer Science (SIGCSE’2014), New York, NY, USA, 5–8 March 2014.

35. Alayoub, H.W.M.; Nouby, A.M.; Amer, A.M. The effect of e-learning based on meta-cognition strategy on students’ achievement and awareness of creative problems solving processes in a C++ course at Kuwait University. In Proceedings of the International Conference on e-Learning “Best Practices in Management, Design and Development of e-Courses: Standards of Excellence and Creativity”, Manama, Bahrain, 7–9 May 2013.