Journal of Education and Practice www.iiste.org
ISSN 2222-1735 (Paper) ISSN 2222-288X (Online)
Vol.17, No.1, 2026

Fostering Computational Thinking through Whole-Task Learning: A 4C/ID-Based Instructional Framework

Rongce Liu

Zhejiang Key Laboratory of Intelligent Education Technology and Application, Zhejiang Normal University, Jinhua, 321004, Zhejiang, China

E-mail: roryyy@163.com

Abstract

In the rapidly evolving digital era, cultivating innovative talents with the ability to solve complex real-world problems has become a strategic priority for global education reform. As a ubiquitous problem-solving methodology, Computational Thinking (CT) serves as a vital cognitive bridge between digital technologies and human reasoning. Computational Thinking is globally recognized as a critical competency for the 21st century and a cornerstone of STEM education. However, current K-12 programming instruction often suffers from a "syntax-driven" approach and knowledge fragmentation, where novice learners experience cognitive overload when facing authentic problems. This creates a barrier to the transfer of algorithmic skills. To address these challenges, this study proposes a holistic instructional design framework based on the Four-Component Instructional Design (4C/ID) model. We constructed a theoretical mapping between the cognitive mechanisms of CT and the four components of the model. Specifically, Learning Tasks provide the authentic context for problem decomposition; Supportive Information scaffolds non-recurrent cognitive strategies, including abstraction, generalization, and iteration; while Procedural Information and Part-task Practice facilitate the automation of recurrent skills such as algorithms and debugging. By integrating these components, the framework effectively manages cognitive load and shifts the pedagogical focus from isolated coding drills to deep, transferable problem-solving capabilities. This study provides a systematic blueprint and theoretical basis for educators to design effective, whole-task CT instruction.

Keywords: Computational Thinking; 4C/ID Model; Instructional Design; Cognitive Load Theory; Whole-task Learning

DOI: 10.7176/JEP/17-1-09
Publication date: January 30th 2026

1. Introduction

In the digital age, Computational Thinking is widely regarded as an essential component of Science, Technology, Engineering, and Mathematics (STEM) learning, holding irreplaceable value particularly in K-12 education (Palop et al., 2025). CT not only enables students to solve problems effectively and efficiently, transferring solutions across different contexts, but also fosters creative and constructive thinking when exploring complex issues. Reflecting this global educational trend, China’s Curriculum Schemes and Standards for Compulsory Education (2022 Edition) has established CT as a core competency in information technology courses, emphasizing a transition from mere skill acquisition to competency-oriented development.

However, current teaching practices face significant challenges. A common misconception equates "programming education" with "code syntax education," often neglecting the cultivation of higher-order thinking skills such as decomposition, abstraction, and iteration (Selby, 2014). This fragmented teaching mode often leads to a failure in knowledge integration: students struggle to apply isolated knowledge to systematically solve complex, real-world problems. Fundamentally, CT is a complex cognitive skill, and its learning process is associated with high cognitive load. Promoting the integration of knowledge, skills, and attitudes without exceeding students' cognitive capacity remains an urgent problem to be solved.

To address this issue, introducing the Four-Component Instructional Design (4C/ID) model holds significant theoretical and practical value. As a holistic instructional design theory tailored for complex learning, the 4C/ID model has been proven effective in supporting the acquisition of complex knowledge and skills. By designing "whole-task" sequences and providing "supportive information," the model effectively manages learners' cognitive load, showing particular advantages in fostering analogical reasoning and systems thinking (e.g., in clinical reasoning and circuit analysis). Furthermore, the model addresses learners' motivational characteristics, enhancing cognitive engagement and learning outcomes through authentic contexts and timely feedback mechanisms.

Although the effectiveness of the 4C/ID model has been verified in professional domains, its application framework for K-12 CT cultivation has not yet been systematically constructed. Therefore, this study aims to build a CT instructional framework for K-12 students based on the 4C/ID model and Cognitive Load Theory. By integrating the theoretical architecture of Van Merriënboer and Kirschner (2018), this framework attempts to resolve the traditional dichotomy between "thinking" and "skills," providing a viable path for cultivating critical thinkers and innovators for the future society.

2. Conceptual Definition and Theoretical Basis

2.1 Deconstructing Computational Thinking

2.1.1 The Definition and Core Elements of Computational Thinking

The concept of CT has evolved with the digital era. In her seminal work, Wing (2006) defined CT as a series of thought processes involved in formulating problems so their solutions can be represented as computational steps and algorithms. She emphasized that this is a universally applicable thinking style, not just a skill for computer scientists. Building on this, Khine (2018) added that the core of CT lies in formulating ambiguous problems into executable steps. From operational and cognitive perspectives, Tedre and Denning (2016) defined CT as a set of computational concepts and thinking patterns derived from programming, software design, and simulation. Rojas-López and García-Peñalvo (2018) further pointed out that it is essentially a cognitive process of generating solutions through abstraction, decomposition, and algorithmic design skills.

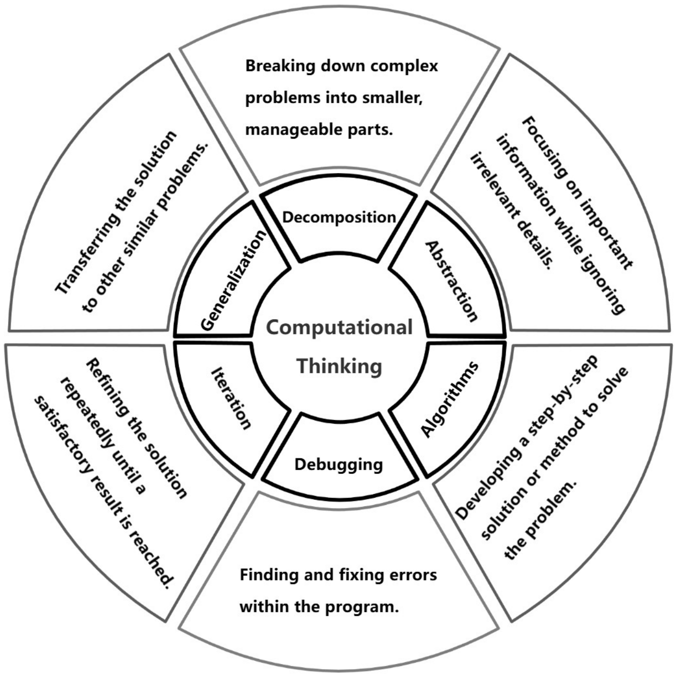
Breaking down complex problems into smaller, manageable parts.
Focusing on important information while ignoring irrelevant details.
Developing a step-by-step solution or method to solve the problem.
Finding and fixing errors within the program.
Refining the solution repeatedly until a satisfactory result is reached.
Transferring the solution to other similar problems.
Decomposition
Abstraction
Algorithms
Debugging
Iteration
Generalization
Computational Thinking

Figure 1. The six elements of CT

To implement CT cultivation in teaching practice, this study deconstructs it into six core elements, as shown in Figure 1. Specifically, decomposition refers to breaking down large, complex problems into smaller, manageable parts to reduce cognitive difficulty. Abstraction involves extracting key information from specific problems while ignoring irrelevant details to build general models. Algorithms involve designing precise steps, methods, or rule sequences to solve problems. Debugging is a metacognitive process of systematically finding and fixing errors during program execution. Iteration emphasizes refining solutions through repeated testing and modification until the expected goal is reached. Finally, generalization refers to transferring a solution or pattern from a specific problem to other similar contexts. These six elements are intertwined, constituting a complete closed loop from problem analysis to solution optimization.

2.1.2 The Definition and Core Elements of Computational Thinking

CT plays an irreplaceable role in K-12 education, widely considered a core 21st-century skill that profoundly impacts students' creativity and problem-solving abilities. Grover and Pea (2013) emphasize that CT has become a key component of STEM education, crucial for developing logical analysis capabilities. It not only helps students solve specific problems efficiently but also, through mechanisms of generalization and transfer, enables them to reuse solutions across different contexts, thereby stimulating constructive higher-order thinking. Furthermore, the educational value of CT extends far beyond programming skills; its core lies in promoting comprehensive cognitive development. As Wing (2008) noted, cultivating students' CT is essentially guiding them to "think like computer scientists"—using abstraction, recursion, and logical reasoning to handle information-based problems. Barr and Stephenson (2011) added that this thinking mode has significant interdisciplinary attributes, enhancing students' ability to apply computational tools in fields such as physics, mathematics, and social sciences. Through such interdisciplinary practice and creative activities, CT education not only stimulates learning motivation but also equips students with key skills to navigate career challenges in an increasingly digitized future.

2.2 Theoretical Architecture of the 4C/ID Model

2.2.1 The Definition and Core Elements of Computational Thinking

The 4C/ID model, proposed by Van Merriënboer et al. (2002), is a holistic design framework specifically built for the acquisition of complex skills. Unlike traditional approaches that fragment complex tasks into isolated knowledge points, this model is based on Cognitive Load Theory, emphasizing whole-task learning and effective management of cognitive load. As shown in Figure 2, the model consists of four interrelated components: learning tasks, supportive information, procedural information, and part-task practice. These components work synergistically to promote the integration of knowledge, skills, and attitudes, enabling learners to solve complex problems in authentic contexts. Table 1 further illustrates these four components and their corresponding ten design steps.

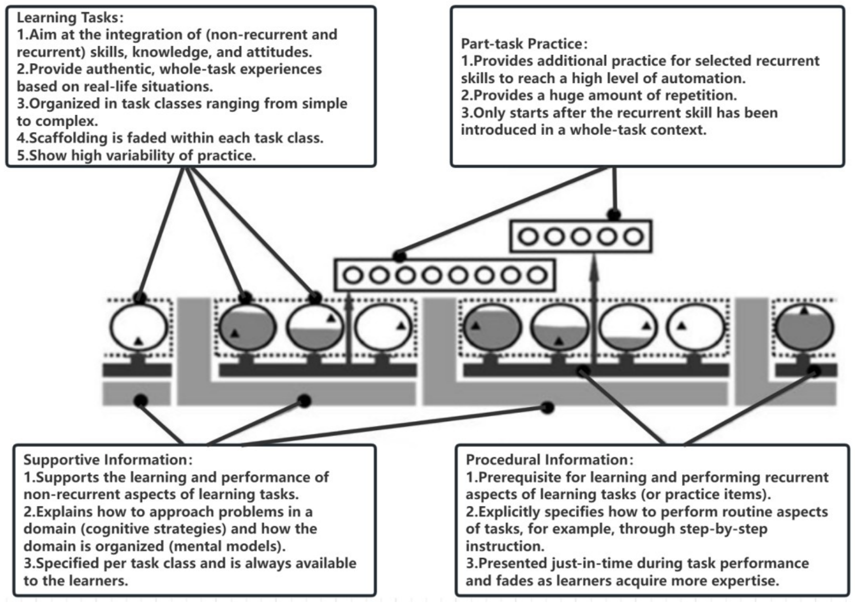
Learning Tasks:
1.Aim at the integration of (non-recurrent and recurrent) skills, knowledge, and attitudes.
2.Provide authentic, whole-task experiences based on real-life situations.
3.Organized in task classes ranging from simple to complex.
4.Scaffolding is faded within each task class.
5.Show high variability of practice.
Part-task Practice:
1.Provides additional practice for selected recurrent skills to reach a high level of automation.
2.Provides a huge amount of repetition.
3.Only starts after the recurrent skill has been introduced in a whole-task context.
Supportive Information:
1.Supports the learning and performance of non-recurrent aspects of learning tasks.
2.Explains how to approach problems in a domain (cognitive strategies) and how the domain is organized (mental models).
3.Specified per task class and is always available to the learners.
Procedural Information:
1.Prerequisite for learning and performing recurrent aspects of learning tasks (or practice items).
2.Explicitly specifies how to perform routine aspects of tasks, for example, through step-by-step instruction.
3.Presented just-in-time during task performance and fades as learners acquire more expertise.

Figure 2. The 4C/ID instructional design model

The specific mechanisms of the components are as follows:

Learning Tasks serve as the core vehicle of the 4C/ID model, aimed at integrating non-recurrent and recurrent skills. These are not abstract exercises but concrete, whole-task experiences based on real-life situations. To prevent cognitive overload in novices, task design follows a simple-to-complex principle, organizing tasks into different task classes. Within each class, scaffolding provided by teachers fades as learners' expertise increases.

Table 1. The four components and ten steps of the 4C/ID model

Four components	Ten steps
Learning tasks	Design learning tasks
	Sequence task classes
	Set performance objectives
Supportive information	Analyze cognitive strategies
	Analyze mental models
	Design supportive information
Procedural information	Analyze cognitive rules
	Analyze prerequisite knowledge
	Design procedural information
Part-task practice	Design part-task practice

Supportive Information is designed to address the challenge of knowledge transfer, primarily supporting the learning of non-recurrent skills such as reasoning, decision-making, and design. Usually provided before task execution, it helps learners construct mental models and cognitive strategies for the domain. It explains how the domain is organized and how to approach problems systematically, bridging prior knowledge and new tasks to facilitate deep understanding.

Procedural Information supports the learning of recurrent skills, such as specific rules and operational procedures, to help learners achieve proficiency. Unlike supportive information, procedural information is presented just-in-time during task execution. It explicitly specifies routine aspects of task completion, such as specific code syntax or software steps. As expertise grows, this information is gradually removed via a fading strategy.

Part-task Practice aims to help learners achieve a high level of automation for specific recurrent skills. When certain sub-skills (e.g., typing, specific algorithmic syntax) require high proficiency to release working memory resources, intensive repetitive practice is necessary. Crucially, part-task practice does not start immediately but is introduced only after the skill has been presented within a whole-task context, ensuring learners understand the background and significance of the practice.

2.2.2 Advantages and Applicability of the 4C/ID Model in the Field of Education

The 4C/ID model provides a systematic educational blueprint for acquiring complex skills by fusing the situational view of social constructivism with the sequencing strategies of cognitivism. Its core advantage lies in managing intrinsic cognitive load through whole-task design, thereby solving the pain point where fragmented knowledge fails to transfer in traditional teaching. This advantage is particularly evident in medical education, where Postma and White (2015) showed that the model significantly improved students' clinical reasoning abilities by providing authentic clinical tasks, managing cognitive load, and implementing effective assessment feedback. This reasoning ability shares a high degree of cognitive isomorphism with debugging and algorithmic design in CT, both of which are non-recurrent complex problem-solving skills.

Regarding multimedia learning and cognitive principles, the model aligns closely with Cognitive Load Theory. Low and Sweller (2005) pointed out that rational multimedia presentation principles, such as integrating context and knowledge, can effectively reduce extraneous cognitive load, promoting inductive learning and conscious abstraction. This is critical for cultivating abstraction and modeling capabilities in CT, as they often involve translating concrete code logic into general mental models. Furthermore, the model's applicability in teacher professional development and K-12 education is supported by empirical evidence. Frerejean et al. (2021) found that the model improved teachers' differentiation skills in mathematics and enhanced student outcomes. In primary education, Zhou et al. (2020) applied the 4C/ID model to oral English teaching, effectively narrowing the achievement gap and improving autonomous learning capabilities. This demonstrates the model's robustness and adaptability across different age groups and disciplinary backgrounds.

3. Adaptability Analysis of Applying 4C/ID to CT Cultivation

Literature analysis indicates that the 4C/ID model, as a representative of holistic instructional design, is highly compatible with the nature of CT as a complex cognitive skill. Hsu et al. (2018) point out that effective instructional design is key to enhancing CT, whereas isolated code training often fails. The 4C/ID model, by focusing learning tasks on authentic contexts, effectively bridges the gap between skill training and problem-solving. This study posits that the model and CT cultivation share a deep compatibility in three dimensions: addressing educational demands, managing cognitive load, and mapping logical elements.

3.1 International Consensus and Challenges in the Development of Computational Thinking

Since Wing (2006) proposed CT, the international education community generally agrees that its core is not just the instrumental application of computer science, but a universal problem-solving mindset. To implement this in K-12 education, Barr and Stephenson (2011) clarified the operational definition of CT, emphasizing capabilities like data analysis, abstraction, and algorithmic automation rather than mere syntax acquisition. However, strictly "coding" is not equivalent to CT. As programming tools evolve rapidly, specific language environments become obsolete. Therefore, Grover and Pea (2013) emphasize that CT education should shift from specific programming skills to the "transfer of thinking". Educators urgently need a model that helps students strip away linguistic appearances to master the transferable core of CT. Current K-12 programming education faces the dual dilemmas of being syntax-driven and fragmented. Traditional bottom-up paths sever internal knowledge connections, preventing students from integrating rules to solve real problems. Meanwhile, existing Project-Based Learning (PBL) often faces a trade-off between authenticity and operability: simplified projects fail to stimulate deep thinking, while complex whole tasks can lead to cognitive overload in novices.

3.2 Cognitive Load Reduction in Complex Computational Thinking Tasks via the 4C/ID Model

In terms of cognitive mechanisms, the learning process of CT involves the concurrent processing of decomposition, abstraction, and algorithm design. According to Cognitive Load Theory, this easily overloads the extraneous cognitive load of novices. The 4C/ID model offers a unique load-reduction mechanism. Unlike traditional linear teaching, it uses sequenced whole-task design, providing high-intensity scaffolding initially and fading it as competence grows. Van Es and Jeuring (2017) confirmed in Scratch teaching that 4C/ID-based design significantly reduces extraneous load during complex programming, allowing students to invest limited cognitive resources into logical construction rather than syntax error correction.

3.3 Classification Support for Recurrent and Non-recurrent CT Skills via the 4C/ID Model

Another advantage is the precise distinction between recurrent and non-recurrent skills. Vandewaetere et al. (2015) noted that effective design must provide differentiated support based on skill attributes. Specifically, CT includes non-recurrent skills like algorithmic design and logic debugging, which rely on problem contexts. The 4C/ID model supports these via supportive information, helping students build cognitive schemas and mental models. Conversely, for recurrent skills like code syntax, the model provides procedural information and part-task practice to promote automation. This mechanism effectively resolves the confusion between strategic knowledge and rule-based knowledge.

3.4 Deep Integration of the Four Components of 4C/ID and the Six Elements of Computational Thinking

Beyond cognitive load and skill classification, the 4C/ID model provides precise anchor points for the six core elements of CT. As shown in Figure 3, this study constructs a logical mapping between the four components of 4C/ID and the six elements of CT. Learning tasks serve as the core vehicle, where decomposition corresponds to the decomposition capability in CT—the logical starting point for reducing load. Supportive information supports non-recurrent high-order thinking: abstraction requires extracting key variables, generalization requires transferring patterns, and iteration is repositioned as a metacognitive strategy. Procedural information provides specific coding rules. Following these rules corresponds to executing algorithms and performing debugging operations. Concurrently, part-task practice ensures the automation of these basic skills, releasing working memory resources to ensure the effective implementation of upper-level thinking strategies.

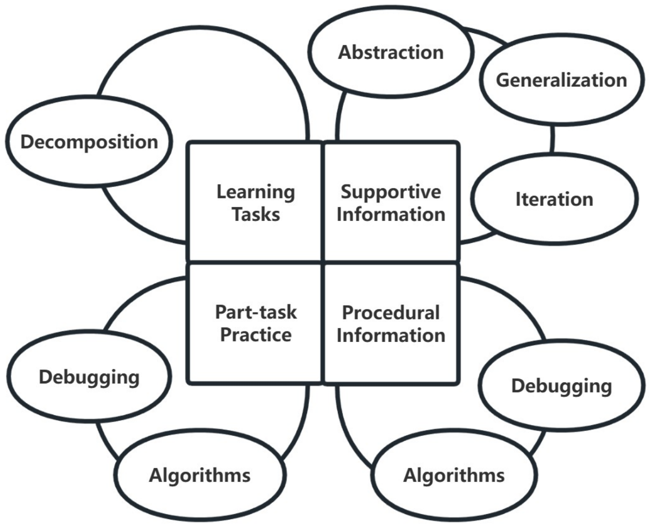

Figure 3. The correspondence between the four components and the six elements

4. Construction of the CT Cultivation Framework Based on 4C/ID

4.1 General Framework Structure

Based on the 4C/ID theoretical architecture and CT cognitive characteristics, this study constructs a hierarchical CT cultivation framework, as shown in Figure 4. The framework consists of four layers from bottom to top: Theoretical Foundation Layer, Instructional Design Layer, Learning Activity Layer, and Assessment & Reflection Layer. These layers interact dynamically to serve the Goal Layer—the comprehensive enhancement of students' CT competency. The framework integrates Holism, Complexity Theory, Situated Learning Theory, and Collaborative Learning Theory. Guided by holistic theory, teaching goals are viewed macroscopically; complexity theory ensures scientific differentiation of task sequences; situated learning emphasizes the connection to social practice; and collaborative learning guides the fusion of individual and group tasks.

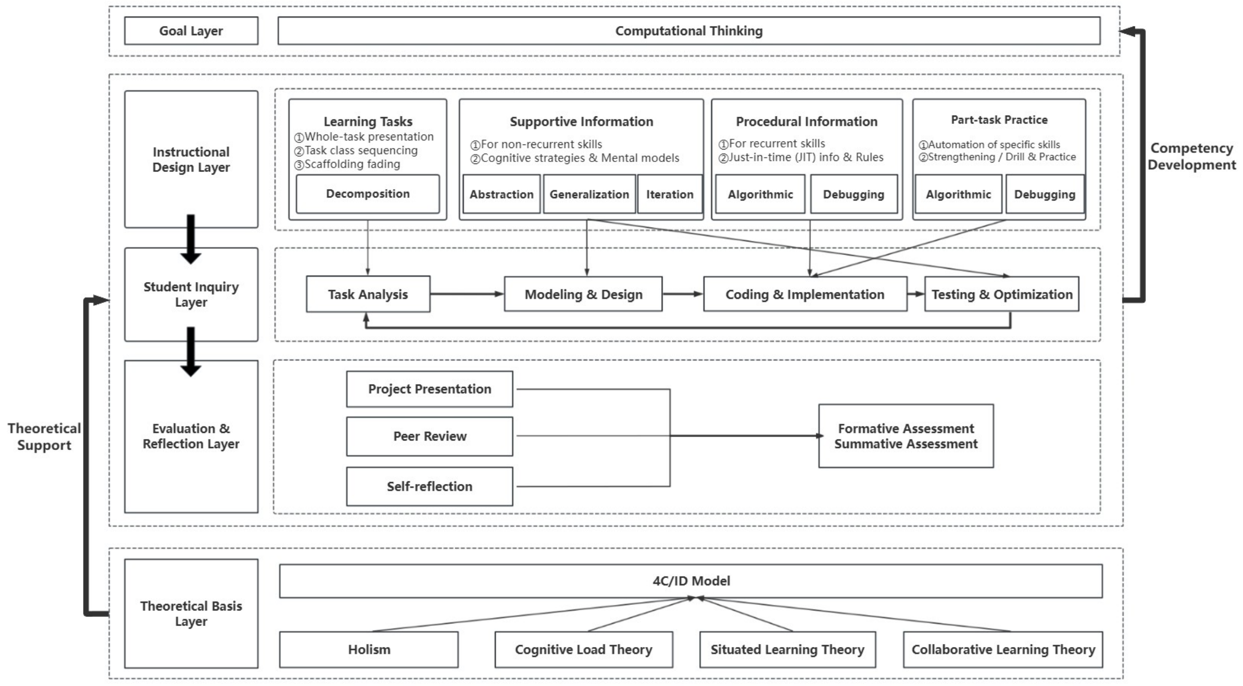

Figure 4. A framework for the development of computational thinking based on the 4C/ID model

4.2 Instructional Design Layer

As the core engine, the instructional design layer strictly follows the four components of 4C/ID. The primary task is to establish performance objectives, integrating knowledge, skills, and attitudes into a whole-task objective, replacing fragmented goals. Learning task design serves as the core vehicle, with principles emphasizing inductive learning. Tasks follow a simple-to-complex sequence (task classes). This sequence helps students acquire knowledge through continuous decomposition practice, promoting deep integration and transfer of CT. This approach has been proven to significantly promote the deep integration and transfer of complex cognitive skills such as CT.

Supportive information design targets non-recurrent complex skills, such as reasoning and decision-making. In CT, this directly corresponds to abstraction, generalization, and iteration. Lye and Koh (2014) suggest that explicit cognitive strategies are crucial in programming. These strategies, serving as mental models, are provided before the task, guiding students to establish systematic problem-solving schemas. Procedural information and part-task practice design focus on recurrent skills. Procedural information is presented just-in-time (e.g., syntax guides). Concurrently, part-task practice provides intensive repetition for specific algorithms and debugging steps. This automation releases working memory, allowing students to allocate resources to high-order thinking, such as abstraction and modeling, achieving a transition from skill proficiency to deep thinking.

4.3 Learning Activity Layer

This layer maps the design into student behaviors during the inquiry phase. The process begins with task analysis and decomposition, where learners analyze the problem structure using cognitive strategies from supportive information. This is followed by modeling and solution design, where students work collaboratively to design solutions. The next stage is coding and implementation, which involves creating artifacts to externalize thinking. Finally, in the testing and optimization stage, students iterate their solutions based on feedback.

4.4 Assessment & Reflection Layer

This layer combines formative and summative assessment. Formative assessment (self, peer, teacher) tracks engagement and CT application during the process. Summative assessment evaluates final artifacts and theoretical mastery. Crucially, this layer encourages reflection, helping students identify knowledge gaps and optimize future strategies.

5. Discussion and Conclusion

5.1 General Framework Structure

This study addresses the skill fragmentation and transfer difficulty in current K-12 CT education by constructing a systematic framework based on the 4C/ID model. The core theoretical contribution lies in clarifying the deep cognitive mapping between CT elements and 4C/ID components. Unlike previous studies that view CT as mere coding, this study innovatively defines iteration, abstraction, and generalization as metacognitive strategies (supported by supportive information) and algorithms and debugging as operational skills (supported by procedural information and part-task practice). This structured design, based on Cognitive Load Theory, provides a logical whole-task solution for moving CT from abstract concepts to concrete implementation.

5.2 Implementation Challenges

Practical implementation in K-12 faces challenges. First is ecological adaptation: The 4C/ID model requires long-term project-based learning, demanding high-performance facilities and flexible curriculum structures. Second is teacher PCK transformation: Teachers must transition from lecturers to instructional designers. The most critical challenge is the dynamic management of scaffolding. As Kalyuga (2007) noted, the timing of fading is critical: too early leads to overload; too late causes the "Expertise Reversal Effect".

5.3 Limitations and Future Prospects

This study is primarily theoretical. Future work should focus on technological integration, such as developing adaptive learning systems that use AI to track cognitive states and automate the fading of support, addressing the difficulty of differentiated instruction in large classes. Additionally, there is a need for interdisciplinary expansion to apply CT beyond IT to STEM fields. Weintrop et al. (2016) emphasized that CT is a core tool for modern scientific and mathematical exploration. Future research should develop 4C/ID-based interdisciplinary task banks (e.g., 3D printing, environmental monitoring) to verify transfer effectiveness. Finally, establishing multi-dimensional evaluation systems that track the trajectory of students' thinking processes over long periods is essential for longitudinal assessment.

References

Aghasaleh, R., Enderle, P., & Puvirajah, A. (2019). From computational thinking to political resistance: Reciprocal lessons from urban Latinx middle school students. Journal for Activist Science and Technology Education, 10(1).

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? ACM Inroads, 2(1), 48–54. https://doi.org/10.1145/1929887.1929905

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing computational thinking. Proceedings of the 2012 Annual Meeting of the American Educational Research Association, Vancouver, Canada, 1-25.

Frerejean, J., van Geel, M., Keuning, T., et al. (2021). Ten steps to 4C/ID: Training differentiation skills in a professional development program for teachers. Instructional Science, 49(3), 395-418.

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. Educational Researcher, 42(1), 38-43.

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.

Kalyuga, S. (2007). Expertise reversal effect and its implications for learner-tailored instruction. Educational Psychology Review, 19(4), 509–539.

Khine, M. S. (2018). Computational thinking in the STEM disciplines: Foundations and research highlights. Springer International Publishing.

Low, R., & Sweller, J. (2005). The modality principle in multimedia learning. In R. Mayer (Ed.), The Cambridge Handbook of Multimedia Learning (pp. 147-158). Cambridge University Press.

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.

Melo, M., & Miranda, G. L. (2018). The effects of 4C-ID model approach on acquisition and transfer of knowledge about electric circuits. International Journal of Web-Based Learning and Teaching Technologies, 13(1), 94-110.

Palop, B., Díaz, I., Rodríguez-Muñiz, L. J., & Santaengracia, J. J. (2025). Redefining computational thinking: A holistic framework and its implications for K-12 education. Education and Information Technologies, 30(10), 13385–13410. https://doi.org/10.1007/s10639-024-13297-4

Park, S., & Yun, H. (2017). The influence of motivational regulation strategies on online students' behavioral, emotional, and cognitive engagement. American Journal of Distance Education, 32(1), 43-56.

Postma, T. C., & White, J. G. (2015). Developing clinical reasoning in the classroom – Analysis of the 4C/ID-model. European Journal of Dental Education, 19(2), 74-80.

Rojas-López, A., & García-Peñalvo, F. J. (2018). Learning scenarios for the subject methodology of programming from evaluating the computational thinking of new students. IEEE Revista Iberoamericana de Tecnologias del Aprendizaje, 13(1), 30-36.

Selby, C. C. (2014). How can the teaching of programming be used to enhance computational thinking skills? [Phd, University of Southampton]. https://eprints.soton.ac.uk/366256/

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. Educational Research Review, 22, 142-158.

Tedre, M., & Denning, P. J. (2016). The long quest for computational thinking. Proceedings of the 16th Koli Calling International Conference on Computing Education Research (pp. 120-129). ACM.

Vandewaetere, M., Manhaeve, D., Aertgeerts, B., Clarebout, G., Van Merriënboer, J. J. G., & Roex, A. (2015). 4C/ID in medical education: How to design an educational program based on whole-task learning: AMEE guide no. 93. Medical Teacher, 37(1), 4–20.

Van Es, N., & Jeuring, J. (2017). Designing and comparing two Scratch-based teaching approaches for students aged 10-12 years. Proceedings of the 17th Koli Calling International Conference on Computing Education Research (pp. 178-182). ACM.

Van Merriënboer, J. J. G., Clark, R. E., & De Croock, M. B. M. (2002). Blueprints for complex learning: The 4C/ID-model. Educational Technology Research and Development, 50(2), 39-61.

Van Merriënboer, J. J. G., & Kirschner, P. A. (2018). Ten steps to complex learning: A systematic approach to four-component instructional design (3rd ed.). Routledge.

Weintrop, D., Beheshti, E., Horn, M., et al. (2016). Defining computational thinking for mathematics and science classrooms. Journal of Science Education and Technology, 25(1), 127–147.

Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(3), 33–35.

Wing, J. M. (2008). Computational thinking and thinking about computing. Philosophical Transactions of the Royal Society A, 366(1881), 3717–3725.

Zhou, S., Zhang, Y., Liu, X., et al. (2020). Empirical research of oral English teaching in primary school based on 4C/ID model. Journal of Higher Education Research, 1(4).