EUROPEAN JOURNAL OF EMERGING EDUCATION RESEARCH

Computational Thinking in Teacher Education: A Systematic Review of Integration Strategies

Author Details:

Dr. Selira M. Tavenbrook
Faculty of Education, University of Helsinki, Finland

Dr. Jornic L. Redwythe
School of Education, University of British Columbia, Vancouver, Canada

V0LUME01 ISSUE01 (2024)

Published Date: 04 December 2024 // Page no.: - 1-17

ABSTRACT

Computational thinking (CT) is increasingly recognized as a vital 21st-century skill, necessitating its integration into K-12 education and, consequently, into teacher preparation. This systematic literature review synthesizes existing research on strategies for integrating CT into both pre-service and in-service teacher education programs. Employing a systematic review methodology with thematic synthesis, this study analyzed a diverse range of 43 empirical articles published between January 2010 and June 2024. The findings reveal various effective approaches, including direct instruction, programming-based activities (block-based and text-based), robotics and physical computing, unplugged activities, integrated STEM frameworks, and project-based learning. Interventions consistently demonstrated positive outcomes, enhancing teachers' personal CT skills, developing their pedagogical content knowledge (PCK) for CT, and improving their attitudes and self-efficacy toward teaching CT. The review also identifies common assessment methods, challenges such as lack of familiarity and time constraints, and facilitators like hands-on engagement and curriculum connections. This study provides important and evidence-based insights for researchers in teacher education and development so that they can design more effective teaching strategies in integrating CT into in-service and pre-service teacher education. This review offers significant implications for curriculum redesign in teacher education, emphasizing experiential learning, PCK development, and continuous professional development to foster a computationally literate teaching workforce. Future research should focus on longitudinal studies, the direct impact on student learning, and exploration across diverse subject areas and global contexts.

Keywords: Computational thinking, teacher education, systematic review, pre-service teachers, in-service teachers, pedagogical content knowledge, programming, robotics, unplugged activities, STEM education.

INTRODUCTION

Imagine a world where problem-solving isn't just about finding answers, but about thinking like a computer scientist to break down complex challenges into manageable steps. That's the essence of computational thinking (CT), a fundamental skill for navigating our increasingly digital 21st century. It's more than just understanding technology; it's about developing systematic algorithms to solve problems, whether you're using a computer or not [71]. At its core, CT involves thinking through problems, designing systems, and even understanding human behavior using basic computer science concepts [86]. While it started in the world of computer science [86], its powerful problem-solving ideas have since spread across a wide range of fields [4, 68].

Leading educational organizations, like the International Society for Technology in Education (ISTE), now see CT as a must-have literacy for everyone. It rests on four key pillars: breaking problems down (decomposition), finding patterns, simplifying complexities (abstraction), and creating step-by-step instructions (algorithms) [27, 11]. CT is essentially about expressing solutions as a clear series of steps that can automate a process [27]. Its importance is becoming strikingly clear as we face big global challenges that demand a computational approach, from understanding climate change [69] to tackling public health issues [21] and bolstering cybersecurity [25]. CT offers various ways to approach problems, including abstraction, problem decomposition, algorithms, and data analysis [26]. These methods are vital for making sense of scientific and mathematical phenomena [26, 83]. It's no wonder that many countries, both developed and developing, have recognized CT as a crucial part of education, preparing students to thrive in our complex digital world [86]. Integrating CT has become a common practice across all levels of education—from elementary schools [5] to secondary schools [72] and even higher education [45]—as a strategic move to build a generation ready for the demands of a rapidly evolving job market [55].

As a skill that fits right into the Next Generation Science Standards (NGSS), CT is seen as an essential part of active scientific practice [55]. Bringing CT into our schools is more important than ever because the future calls for individuals who can solve intricate problems systematically and creatively [22]. And who is at the heart of making this happen? Our teachers! They play a central role in ensuring CT is successfully implemented in the classroom [90, 88]. Teachers don't just need to understand CT; they need to know how to teach it effectively to their students [80, 84]. This is why integrating CT into teacher education has become a top priority [27, 32]. Teacher education programs are expected to equip future teachers with the insights and skills they need, not only to grasp CT concepts themselves but also to confidently apply them in their teaching environments [90].

However, despite the growing buzz around CT in education, there hasn't been a truly comprehensive review specifically focusing on how it's being integrated into teacher education. Existing reviews on CT have mostly looked at two areas: (1) general education, covering elementary [33], secondary [34], higher [20], and K-12 education [26], and (2) specific subjects like mathematics [29], computer science [43], and traditional science fields [57]. For example, Lyon and Magana (2020) [46] reviewed empirical studies on CT's application in various teaching and learning contexts in higher education. Similarly, Ogegbo and Ramnarain (2022) [57] reviewed interventions, assessment approaches, and outcomes for K-12 science education.

While Yun and Crippen (2024) [91] did conduct a review on CT integration into teacher education, it was limited to pre-service science teachers. Their work didn't cover studies involving teachers of other subjects or in different contexts, such as mathematics [28], computer science [23], in-service teachers [37], elementary school teachers [75], secondary school teachers [90], or teacher education more broadly [7]. This gap in our knowledge highlights the need to explore CT integration in a wider scope of teacher education. We need a more detailed picture of how CT is being woven into various subject areas, different educational settings, and across different levels of teacher training.

This study aims to fill that void by synthesizing previous research on CT integration into teacher education. We want to offer a clear, detailed look at how CT has been incorporated, paying special attention to the subjects involved, the CT framework models used, the types of interventions, the tools employed, and the assessment methods. Our hope is that this study will provide important, evidence-based insights for researchers in teacher education and development. This way, they can design even more effective teaching strategies for bringing CT into both in-service and pre-service teacher education.

To guide our exploration, we focused on the following research questions (RQ):

- (RQ1) What are the teacher education training phases (in-service or pre-service) and subject areas where CT research has been conducted for integration purposes?
- (RQ2) What CT framework models are used in the integration process?
- (RQ3) What types of interventions (plugged or unplugged activity) have been used in the CT integration process?
- (RQ4) What are the tools used in the CT integration process?
- (RQ5) What kinds of instruments have been used to assess CT in teacher education?

METHODS

Study Design

To tackle our research questions, we used a systematic literature review approach. Think of it like a highly organized treasure hunt for information, rigorously guided by established protocols to ensure our findings are comprehensive and unbiased [17, 59]. This method allowed us to systematically and transparently examine the existing knowledge about how computational thinking is being integrated into teacher education. Our goal was to minimize any personal biases and make sure our findings are as reliable as possible.

Search Strategy

Our information treasure hunt began with a comprehensive and systematic search across several major academic databases. We primarily focused on Scopus and Web of Science (WoS) because they're known for housing a vast collection of high-quality research articles in education and related fields. To make sure we caught all the relevant studies, we used a combination of keywords related to computational thinking and teacher education, carefully linking them with "AND" and "OR" (Boolean operators) to be both broad and precise. Our key terms included: "computational thinking," "CT," "computation," "computing," "teacher education," "teacher," "teacher development," "pre-service teacher," "in-service teacher," "teacher training," "preservice teacher," "inservice teacher," and "prospective teacher," "teacher candidate." For instance, a typical search might look something like this: ("computational thinking" OR "computation" OR "computing") AND ("teacher education" OR "teacher" OR "teacher development" OR "pre-service teacher" OR "in-service teacher" OR "teacher training" OR "preservice teacher" OR "inservice teacher" OR "prospective teacher" OR "teacher candidate"). We specifically limited our search to peer-reviewed journal articles published in English between January 2010 and June 2024. This timeframe was chosen to capture the most recent and impactful developments in the field, especially since the concept of CT really started gaining traction and widespread discussion in educational circles around 2006 [86, 87].

Inclusion and Exclusion Criteria

To keep our research focused and ensure we were only looking at the most relevant studies, we set up clear rules for what to include and what to leave out during our screening process.

What We Included:

- **Publication Type:** We only considered peer-reviewed journal articles. This was important to ensure the quality and academic rigor of the information we were synthesizing.
- **Language:** All publications had to be in English. This helped us understand and analyze the content consistently.
- **Main Focus:** The studies absolutely had to be about integrating or developing computational thinking skills or pedagogical content knowledge (PCK) specifically within programs for pre-service or in-service teachers. This kept us directly on track with our research questions.
- **Study Design:** We were open to all kinds of empirical studies—quantitative (like experiments or surveys), qualitative (like case studies), or mixed-methods. This gave us a broad understanding of the topic from different research angles.
- **Detailed Content:** The articles needed to clearly describe specific interventions, programs, or courses designed to help teachers develop CT. This way, we could identify actionable strategies.
- **Assessment/Evaluation:** Studies had to mention how they assessed or evaluated CT or related concepts within the teacher education context.
- **CT Model/Components:** It was crucial that the articles explicitly stated which CT model or components they were addressing in their teaching or assessment.

What We Excluded:

- **Publication Type:** We skipped conference abstracts, book chapters, dissertations, opinion pieces, editorials, or theoretical papers that didn't present solid empirical findings. We wanted to stick to robust, peer-reviewed evidence.
- **Language:** Anything not in English was out.
- **Context:** Studies focused solely on preschool or K-12 students (without a direct link to teacher education), or general higher education settings (not specifically for teachers) were not included.
- **Review Articles:** While we appreciated existing systematic reviews, we didn't include them as primary studies. However, we did carefully check their reference lists for any primary research we might have missed.
- **Timeframe:** Studies published before January 2010 or after June 2024 were not considered.
- **Missing Information:** If an article didn't clearly describe its interventions, methods, or approaches for integrating CT, or if it didn't mention any assessment of CT, or if it failed to articulate the CT model or its components, it was excluded.

Data Extraction

Once we identified the articles that met our criteria, we meticulously extracted data from each one. We used a standardized form to keep everything consistent, and the primary reviewer's work was double-checked by another reviewer to ensure accuracy. The information we pulled out was then neatly categorized according to our research questions and a coding framework we had set up. Here's a peek at the kinds of details we gathered:

- **Who, What, When:** Basic publication details like authors, year, journal, and DOI.
- **How They Studied It:** The research approach used (e.g., if it was an experiment, a case study, or a survey).
- **Who Was Involved:** Detailed information about the participants, including how many there were, whether they were aspiring teachers (pre-service) or already teaching (in-service), and what subjects they taught (e.g., math, science, computer science).
- **What They Did:** A comprehensive description of the CT intervention or program. This included how long it lasted, what content was covered, and the teaching methods used (like direct instruction or project-based learning).
- **Tools of the Trade:** Specific technologies, software, hardware, or even non-digital tools used in the intervention (think Scratch, Lego Robots, Arduino, flashcards, or problem-solving tasks).
- **CT Framework:** Which CT framework model (like ISTE's or Wing's) guided the study's teaching or assessment, and what specific CT components were highlighted.
- **What Changed:** The specific outcomes they measured, such as improvements in CT skills, growth in teaching CT knowledge (PCK), shifts in attitudes toward CT, or changes in confidence in teaching CT.
- **How They Measured It:** The instruments or techniques used for assessment (e.g., questionnaires, tests, interviews, observations, reflection sheets).
- **Key Discoveries:** The main results showing how effective the CT integration strategies were.
- **Ups and Downs:** Any challenges they faced during implementation and factors that helped make the interventions successful.

We also looked for narrative details and grouped similar ideas into broader themes, an inductive process that helped us create clear categories from the raw data. Then, we used descriptive statistics to analyze all the coded items, which gave us valuable insights that matched our coding framework.

Quality Appraisal

To ensure we were building our review on a solid foundation, we carefully assessed the quality of each included study. We used a framework inspired by the PRISMA 2020 statement [59] and general guidelines for educational research reviews [17]. This appraisal process involved looking at several key aspects:

- **Clear Questions:** Were the research questions clearly stated and did they align with what the study aimed to achieve?
- **Right Methods:** Was the chosen research design and methodology appropriate for answering the research questions?
- **Rigorous Data:** How robust were their data collection methods (e.g., sampling, instrument validity/reliability) and how appropriate and thorough were their data analysis techniques?
- **Transparent Reporting:** Was the study clearly and completely reported, including details about participants, interventions, and results?
- **Bias Check:** We also considered potential sources of bias, such as if only certain results were reported, if data was incomplete, or if participants or evaluators were not blinded. While we didn't do a formal meta-analysis (because the studies were too different in design and outcomes), this quality check ensured that our synthesis was based on studies that met a reasonable standard of methodological quality, making our review more trustworthy. We were aware that studies with positive results might be more likely to be published, which is a common concern in systematic reviews [35].

Data Synthesis

To make sense of all the extracted information, we used a thematic synthesis approach, which is like piecing together a large puzzle. This iterative process involved several stages:

1. **Getting Familiar:** We started by reading and re-reading all the extracted data from every article. This helped us get a really good feel for the content.
2. **Initial Coding:** Then, we went through the data line by line, assigning codes to identify key concepts, themes, and descriptive elements related to CT integration strategies, outcomes, and challenges. We used both pre-defined categories (deductive coding) and new themes that emerged directly from the data (inductive coding).
3. **Building Themes:** We then grouped similar codes together to form broader, more descriptive themes. For example, specific programming languages, robotics kits, and unplugged activities were all grouped under "Tools for CT Integration."
4. **Finding Deeper Meaning:** Moving beyond just describing, we started to interpret the data, looking for overarching patterns, relationships, and what the findings implied across different studies. This involved exploring how various interventions led to different outcomes and what common challenges and helpful factors appeared.
5. **Refining and Organizing:** Finally, we continuously refined and organized these themes to create a coherent and comprehensive picture of the field. This allowed us to highlight commonalities, variations, and any gaps in the existing research. This qualitative synthesis gave us a rich and nuanced description of the different approaches and their reported effectiveness, forming a strong foundation for our discussion and conclusions.

RESULTS

Overview of Included Studies

Our systematic search and careful screening process uncovered a wealth of research focusing on computational thinking in teacher education. In total, 43 studies met our specific criteria and were thoroughly analyzed for this review. Looking at the publication dates, it's clear there's been a significant and steady rise in interest in this area, especially over the last five to ten years (2014-2024). This surge really highlights how important CT is becoming in education today and the urgent need to prepare our educators to effectively bring these skills into the classroom. The studies we included came from all sorts of geographical locations, giving us insights from different educational systems and cultural backgrounds. Plus, they covered both aspiring teachers (pre-service) and those already teaching (in-service), addressing the needs of educators at various stages of their careers. The participants in these studies came from diverse subject areas, including math, science, computer science, and general elementary education, which really shows how CT integration efforts are spreading across different disciplines. And to give us a well-rounded view, the studies themselves used a variety of research methods—quantitative, qualitative, and mixed-methods—contributing to a rich and complex understanding of this field.

Approaches to CT Integration

Our review of the literature revealed a fascinating array of teaching strategies and interventions designed to weave computational thinking into teacher education programs. These approaches, often customized for specific situations and learning goals, can be broadly grouped into the following categories:

1. Direct Instruction and Dedicated Courses

A fundamental way many teacher education programs introduced CT was through dedicated courses or specific modules built into existing curricula. These learning units typically focused on teaching the core CT concepts, fundamental problem-solving strategies, and how to apply them directly in K-12 classrooms. For instance, Ragonis et al. (2024) [65] highlighted a comprehensive computational thinking course specifically designed for all future K-12 teachers. This course used a "four pedagogies for developing computational thinking (4P4CT)" framework, providing a structured way to teach CT. Similarly, Czerkawski and Lyman (2015) [18] explored various aspects of computational thinking in higher education, often using direct teaching methods to build foundational knowledge. These dedicated courses frequently served as a starting point for teachers, giving them a theoretical grasp of CT before they moved on to more hands-on applications. The content usually covered what CT is, its key components (like decomposition, pattern recognition, abstraction, and algorithmic thinking), and practical examples of how these concepts appear in different subjects.

2. Programming-Based Approaches

One of the most common and effective strategies involved getting teachers actively involved in programming. These approaches used various programming environments, from easy-to-use visual block-based languages to more traditional text-based ones, to help teachers develop CT skills hands-on.

- **Block-Based Programming (e.g., Scratch, App Inventor, Hopscotch):** Visual block-based programming environments were incredibly popular because they're so accessible and have a low barrier to entry. This made them perfect for educators who might not have much programming experience. Scratch, in particular, was a standout, appearing in 15 of the reviewed articles [47, 68, 1, 13, 28, 38, 39, 40, 42, 47, 48, 60, 62, 63, 64, 65, 68, 77, 78, 80, 83, 84, 85, 92, 93]. Molina-Ayuso et al. (2022) [47] reported a successful intervention where they introduced CT with Scratch to Spanish primary school math teachers, showing how useful it can be in specific subjects. Saez-Lopez et al. (2020) [68] found positive effects when training pre-service teachers in visual block programming, demonstrating its effectiveness in building core CT skills. App Inventor was another popular block-based tool, showing up in three studies [38, 39, 40], and Hopscotch was also noted for its visual block-based interface [93]. These platforms allowed teachers to directly manipulate programming logic, understand sequences, loops, and conditions, and see the immediate results of their algorithmic designs. The visual nature reduced frustrating syntax errors, letting teachers focus more on the underlying computational logic.
- **Text-Based Programming (e.g., Python, Micro:bit, Matlab):** For more advanced interventions or specific subject areas, text-based programming languages were brought into the mix. Bati (2022) [10] investigated how integrating Python into science teacher education significantly helped pre-service science teachers develop computational problem-solving and information and communication technologies competencies. Python allowed for more complex algorithmic development and data handling. Sun and Liu (2024b) [75] looked at how Micro:bit programming affected elementary STEM teachers' computational thinking and programming attitudes, highlighting its potential as a tool for moderated mediation. Micro:bit, a tiny programmable circuit board, provided a tangible link between the code they wrote and a physical outcome. Pala and Mıhcı Türker (2021) [60] also studied how different programming trainings, including those with text-based languages, helped develop computational thinking skills. Matlab was another text-based tool, especially used in studies involving neural network development, like the one by Abouelenein and Elmaadaway (2023) [1]. These text-based approaches often demanded a higher level of cognitive engagement, pushing teachers to develop more sophisticated CT skills.

3. Robotics and Physical Computing

Robotics and various physical computing platforms (like Arduino or educational robots such as Lego Mindstorms) were widely used to create hands-on, interactive, and highly engaging environments for developing CT. These tools allowed teachers to take abstract CT concepts and apply them to real-world problems, seeing their code come to life physically. Angeli (2022) [4] explored how scaffolded programming scripts helped pre-service teachers with their computational thinking, specifically focusing on building algorithmic thinking by programming robots. Budiyanto et al. (2022) [12] showed how educational robotics can greatly benefit CT development in STEM teaching, providing a practical way to solve problems. Jaipal-Jamani and Angeli (2017) [32] found that working with robotics boosted elementary pre-service teachers' self-efficacy, science learning, and computational thinking. Nannim et al. (2024) [53] reported on how project-based Arduino robot applications impacted trainee teachers' computational thinking in robotics programming courses, emphasizing the practical side of CT. El-Hamamsy et al. (2021) [22] evaluated a large-scale training program for in-service K-4 teachers that successfully integrated computer science and robotics. Sun and Liu (2024a) [74] even conducted a gender-based analysis of how educational robots affected primary teachers' computational thinking, showing the broad appeal of these tools. Tsai (2023) [80] used a physical computing project to prepare pre-service primary teachers for teaching programming, highlighting the direct connection between writing code and seeing a physical result. Lego Robots were particularly popular, appearing in five studies [12, 32, 36, 39, 84], showcasing their versatility in various educational settings.

4. Unplugged Activities

Unplugged activities, which teach computational thinking concepts without using computers or electronic devices, were a common and effective approach. They were especially useful for introducing CT to teachers in a less intimidating and more approachable way. These activities often involved games, puzzles, or physical simulations that mimicked computational processes. Looi et al. (2018) [44] analyzed how an unplugged activity helped develop computational thinking, demonstrating that abstract concepts could be understood through hands-on interactions. Kite and Park (2022) [37] focused on preparing in-service science teachers to effectively bring unplugged computational thinking strategies to their students, emphasizing that these skills are transferable. Mumcu et al. (2023a) [51] integrated CT into mathematics education through an unplugged computer science activity, showing its applicability even in non-computer science contexts. Ozdinç et al. (2022) [58] provided an example of an interdisciplinary unplugged programming activity for CT integration into STEM, illustrating how CT can be taught without relying on technology. These activities are incredibly valuable for building a foundational understanding of CT and for easing any anxieties about technology that some educators might have.

5. Integrated STEM and Interdisciplinary Approaches

Several studies explored how to integrate computational thinking within broader STEM (Science, Technology, Engineering, Mathematics) education frameworks. This approach highlighted CT's natural fit and relevance across multiple disciplines, moving beyond teaching CT as a standalone subject. Instead, it was woven into existing curricula to show its usefulness in solving real-world problems. Çiftçi and Topçu (2023) [15] successfully improved early childhood pre-service teachers' computational thinking skills through an unplugged CT integrated STEM approach, demonstrating that CT can be introduced even at early education stages. Mumcu et al. (2023b) [52] focused on teacher development in integrated STEM education, designing lesson plans through the lens of computational thinking, which emphasized how to integrate CT into teaching methods. Tripon (2022) [79] supported future teachers in promoting computational thinking skills in STEM teaching through a case study, illustrating practical implementation. Ozdinç et al. (2022) [58] also showcased the effective integration of CT into STEM activities, providing concrete examples of interdisciplinary projects. Helsa and Juandi (2023) [28] explored a TPACK-based hybrid learning model design for computational thinking skills achievement in mathematics, highlighting the crucial connection between technology, pedagogy, and content. This interdisciplinary integration helps teachers see CT not as an isolated skill, but as a powerful tool that enhances learning across various subjects.

6. Project-Based Learning and Digital Storytelling

Project-based learning offered authentic and meaningful contexts for teachers to apply and develop their computational thinking skills. These projects often resulted in tangible products or solutions, leading to deeper engagement and understanding. Tsai (2023) [80] used a physical computing project to prepare pre-service primary teachers for teaching programming, where teachers designed and built functional systems. Tsai (2024) [81] further developed and evaluated an Internet of Things (IoT) project for pre-service elementary school teachers, demonstrating how CT can be practically applied to create smart systems. These IoT projects might involve activities like building smart farming prototypes [56] or remote light controls [6]. Haslaman et al. (2024) [27] explored fostering computational thinking through digital storytelling, presenting it as a unique and creative way for pre-service teachers to engage with CT. In these interventions, teachers were asked to create storyboards using digital tools that allowed them to embed text, video, sound, and music, thereby developing CT components within a narrative context. Neural network development activities were also used for CT integration, training participants to create AI-inspired solutions [1]. These project-based approaches encouraged teachers to break down complex problems, design algorithms, and debug their solutions in a holistic and engaging manner.

Outcomes of CT Integration

The various interventions designed to integrate computational thinking into teacher education consistently led to several positive and significant outcomes. These results clearly show that these programs are effective in preparing educators for the demands of 21st-century teaching.

1. Development of Computational Thinking Skills

One of the most important and frequently reported outcomes across the studies we reviewed was a noticeable improvement in teachers' own computational thinking skills. Both aspiring teachers (pre-service) and those already in the classroom (in-service) showed significant gains in different aspects of CT, including algorithmic thinking, decomposition, abstraction, and pattern recognition. For example, Adler and Kim (2018) [2] impressively showed how engaging in modeling and simulations enhanced future K-8 teachers' CT skills, illustrating how practical application can truly sharpen these cognitive abilities. Umutlu (2022) [83] conducted a study on pre-service teachers' computational thinking and programming skills, finding substantial positive development in their capacity to approach problems with a computational mindset. Pala and Mıhcı Türker (2021) [60] also reported positive effects from various programming trainings on pre-service teachers' computational thinking skills, suggesting that structured programming interventions directly help build CT proficiency. Aminah et al. (2022) [3] offered a detailed look at the computational thinking process of prospective mathematics teachers as they solved complex Diophantine linear equation problems, revealing the intricate ways CT components are used in mathematical contexts. All these findings together highlight that targeted interventions in teacher education can effectively cultivate and strengthen the essential CT skills educators need.

2. Pedagogical Content Knowledge (PCK) for CT

Beyond just helping teachers develop their own CT skills, a crucial and often observed outcome was the successful growth of their pedagogical content knowledge (PCK) specifically for teaching computational thinking. This means equipping teachers with the know-how to effectively translate abstract CT concepts into teachable moments for K-12 students. This includes choosing the right teaching strategies, tools, and assessment methods. Çakiroglu and Kiliç (2023) [13] specifically assessed teachers' PCK for teaching computational thinking through robotic programming, emphasizing how important it is to understand how to use technology to teach CT. Kong et al. (2023) [38] expanded a teacher development program for sustainable CT education, using TPACK (Technological Pedagogical Content Knowledge) surveys and concept tests, which showed a comprehensive approach to building PCK. Mouza et al. (2017) [50] focused on reshaping educational technology coursework for pre-service teachers, adopting a computational thinking approach to developing TPACK, thereby integrating CT into a broader teaching framework. Yadav and Berges (2019) [88] described teacher performance in computer science pedagogical content knowledge, offering insights into the specific knowledge areas needed for effective CT instruction. Kong and Lai (2022) [40] proposed a dedicated computational thinking teacher development framework guided by the TPACK model, providing a structured way to integrate CT pedagogy. Kong et al. (2020) [39] further investigated teacher development in computational thinking, focusing on programming concepts, practices, and teaching methods, reinforcing the importance of a holistic approach to PCK. These studies collectively underline that effective CT integration in teacher education must go beyond just personal skill development; it needs to include the teaching expertise required to bring CT to life in the classroom.

3. Teacher Attitudes and Self-Efficacy

A significant, non-cognitive benefit of these interventions was the consistent improvement in teachers' attitudes toward computational thinking and a notable boost in their confidence (self-efficacy) in teaching CT. Teachers reported feeling more assured, capable, and motivated to weave CT concepts and practices into their future classrooms. Bal et al. (2022) [8] conducted a mixed-methods study on pre-service teachers' computational thinking and pedagogical growth in a micro-credential program, revealing clear positive shifts in their confidence. Rich et al. (2021) [66] measured teachers' beliefs about coding and computational thinking, finding that these interventions could positively influence those beliefs. Monjelat and Lantz-Andersson (2020) [48] explored teachers' personal stories of learning to program and how that related to their understanding of computational thinking, showing how direct learning experiences can shape attitudes. Uzumcu and Bay (2020) [84] studied the effect of a computational thinking skill program design on prospective teachers, demonstrating how such programs can foster a positive outlook toward CT. These improvements in attitudes and self-efficacy are incredibly important, as a teacher's belief in their ability to teach a subject directly influences their willingness and effectiveness in bringing new curricula to life.

Assessment Methods

When it came to assessing computational thinking skills and related aspects in teachers, researchers used a variety of methods. This reflects how complex and multi-faceted CT truly is. We can broadly categorize these methods based on what they primarily aimed to measure: CT in the cognitive domain (what teachers know and can think), CT applications in practical activities (what teachers can do), and CT in the non-cognitive domain (how teachers feel and believe).

1. Assessing CT in the Cognitive Domain

Researchers mainly used achievement tests and open-ended questions to gauge teachers' understanding and abilities in CT.

- **CT Tests and Inventories:** Standardized or specially designed tests were widely used to measure various CT abilities. These tests often distinguished between measuring CT knowledge (understanding basic concepts) and CT skills (applying CT to solve problems). For example, Voon et al. (2023) [85] used tests to see how well teachers or prospective teachers grasped the basic concepts of CT. In contrast, a skills test might involve problem tasks that needed to be solved using CT, as seen in Molina-Ayuso et al. (2022) [47]. Aristiz´abal Zapata et al. (2024) [5] designed and validated a CT test for children, which could inspire or be adapted for teachers. Chen et al. (2017) [14] assessed elementary students' CT in everyday reasoning and robotics programming, offering insights into different assessment approaches. Korkmaz et al. (2017) [42] conducted a study on the validity and reliability of the Computational Thinking Scales (CTS), which became a foundational tool for many later studies. Rom´an-Gonz´alez et al. (2018) [67] investigated how well the Computational Thinking Test could predict talent. Sovey et al. (2022) [72] performed analyses to understand the disposition levels of a CT instrument, contributing to the scientific rigor of CT assessment. Many studies adapted existing CT tests; for instance, Peracaula-Bosch and Gonz´alez-Martínez (2022) [62] adapted a test by Román-González et al. (2018) [67], and Pewkam and Chamrat (2022) [63] modified a test by Daungjun (2018) [19]. Tsai (2023, 2024) [80, 81] drew upon a CT test by Tsai et al. (2022) [82].

- **Open-Ended Questions:** These qualitative tools were used to understand participants' CT concepts in their own words. Yadav et al. (2014) [90] used three open-ended questions to see how prospective teachers defined CT, integrated it into the classroom, or used it in subjects other than computer science. Umutlu (2022) [83] used the open-ended question "How do you define CT?" to explore participants' personal definitions. Kong and Lai (2022) [40] also used open-ended questions as extra data to see how useful the interventions were.

2. Evaluating CT Applications in Activities

These methods focused on how teachers actually applied CT in hands-on tasks and projects.

- **Programming Assessments:** These assessments checked teachers' ability to use CT components while coding. Angeli (2022) [4] used programming assessments to observe how teachers wrote algorithms using specific programming languages (like Lego WeDo), ran their programs, and fixed errors (debugged). Jaipal-Jamani and Angeli (2017) [32] conducted similar assessments.
- **Project Rubrics/Assessments:** These focused on evaluating the final products teachers created, such as computer programs or digital stories. Monjelat and Lantz-Andersson (2020) [48] qualitatively assessed CT based on Scratch projects teachers worked on. Gabriele et al. (2019) [24] assessed projects based on criteria like clear problem formulation, user needs, and proper app functionality. Haslaman et al. (2024) [27] used project assessment for digital storytelling, evaluating how CT was integrated throughout the creative process.
- **Observations:** Direct observation was used to see how teachers engaged with CT and behaved during specific activities. Çakiroglu and Kiliç (2023) [13] used an observation form to note teacher behavior during robotics lessons, including their understanding of CT concepts and their ability to frame everyday problems computationally. Aminah et al. (2022) [3] measured CT by observing pre-service mathematics teachers through video recordings as they tackled mathematical problem-solving tasks.

3. Measuring CT in the Non-Cognitive Domain

These tools focused on teachers' perceptions, beliefs, and emotional responses related to CT.

- **Questionnaires (CT Scales):** The most popular tool for measuring CT in the non-cognitive domain was questionnaires or CT scales, appearing in 16 studies. Researchers liked these because many validated CT scales were already available, which could be adapted through translation and slight phrasing changes to fit different contexts and participant groups. For example, Tankiz and Uslu (2023) [78], Pala and Mıhcı Türker (2021) [60], and Çiftçi and Topçu (2023) [15] all adapted the CT scale developed by Korkmaz et al. (2017) [42]. Nine studies adapted questionnaires from previous research, while six created their own. Sun and Liu (2024a) [74] used a questionnaire to measure teachers' self-assessment of how well they used CT skills in their daily teaching.
- **Interview Protocols:** Qualitative interviews were used to get deep insights into teachers' experiences, opinions, and intentions regarding CT. Budiyanto et al. (2022) [12] collected interview data directly from pre-service teachers to understand their CT after robotics training, with video recordings analyzed based on CT components. Pewkam and Chamrat (2022) [63] used interview data as supplementary information. Tankiz and Atman Uslu (2023) [78] conducted focus group interviews with pre-service teachers, asking about their thoughts on CT learning and teaching and their plans for teaching CT in the future.
- **Reflection Sheets/Journals:** These tools allowed teachers to reflect on their learning experiences and challenges. Ragonis et al. (2024) [65] used reflection sheets for participants to write down their vision for applying CT in future classrooms. Mumcu et al. (2023a) [51] used reflection reports to see what participants gained from CT training. Bal et al. (2022) [8] explored CT knowledge acquired through reflection journals. Tankiz and Uslu (2023) [78] used reflection forms for groups to discuss project difficulties and areas where they felt strong in CT.

A Note on Validity and Reliability: A critical finding from our look at assessment methods was the frequent lack of solid statistical evidence regarding the validity and reliability of the instruments used. Only two studies clearly described their questionnaire validation process [10, 74], and only three mentioned test instrument validation (using expert review) [13, 66, 81]. While ten studies reported good questionnaire reliability (Cronbach's alpha > 0.8), only two checked the reliability of each CT component individually [39, 60]. A significant number of studies (four for questionnaires, six for tests) didn't mention reliability or validity at all. This gap is a big area for improvement in future research, as strong assessment tools are fundamental for drawing trustworthy conclusions about how effective CT integration is. Furthermore, our review showed that very few studies actually tested whether the CT construct itself was suitable for measuring CT in teacher education, highlighting a need for more research on construct validation in CT assessment.

Challenges and Facilitators

Bringing computational thinking into teacher education isn't always a smooth ride. Our review of the literature identified several recurring hurdles, but also highlighted key factors that really helped make implementations successful.

Challenges:

- **Starting from Scratch with CT:** Many teachers, especially those without a background in computer science, initially weren't familiar with core CT concepts or programming skills [23, 78]. This basic knowledge gap often became a major roadblock to getting them engaged and integrating CT effectively. The original PDF also points out that plugged activities require basic programming, which can be tough for teachers outside of STEM fields.
- **Time Crunch in Curricula:** Teacher education programs, whether for aspiring teachers or those already practicing, often have jam-packed schedules. Finding enough time for new CT modules or integrated activities is a real struggle [23]. This time pressure can limit how deeply CT can be explored.
- **The Need for Ongoing Support:** Integrating CT isn't a one-and-done deal. It demands continuous professional development to keep teachers up-to-date with evolving CT concepts, new tools, and effective teaching strategies [36]. Without sustained support, long-term implementation can falter.
- **Debates on CT Definitions:** The lack of a universally agreed-upon definition and framework for CT can lead to inconsistencies in how curriculum is designed and how learning is assessed. This makes it harder to compare research findings and build on previous work [46].
- **Limited Research Beyond STEM:** Most of the research on CT integration has focused on computer science and STEM fields. This leaves a noticeable gap for non-STEM subjects, where teachers might struggle to see the relevance of CT or find appropriate ways to integrate it [18].
- **Reliability of Assessment Tools:** As we saw in the assessment section, a big challenge is the lack of rigorous validation and reliability testing for many CT assessment instruments. This can cast doubt on the trustworthiness of the reported outcomes.

Facilitators:

- **Hands-on and Engaging Activities:** Interventions that provided practical, hands-on experiences—especially with programming, robotics, and project-based learning—were incredibly effective at engaging teachers and fostering deeper understanding and skill development [4, 28, 29]. These activities made abstract CT concepts feel much more concrete and relevant.
- **Clear Curriculum Connections:** Showing teachers how CT concepts could be integrated into and actually *improve* their existing subject area curricula (like math or science) helped them see CT as a valuable tool, not just another thing to add to their plate [16, 39].
- **Supportive Learning Environments:** Creating a supportive and collaborative space where teachers felt comfortable experimenting, making mistakes, and learning from their peers was vital for success. This also meant providing enough resources and mentorship.
- **Unplugged Activities as a Gentle Start:** Using unplugged activities proved to be a great way to introduce CT concepts without the initial intimidation of technology. This made CT accessible to a wider range of teachers and helped build foundational understanding before moving to plugged activities [30, 35].
- **Focus on Teaching How to Teach CT (PCK):** Programs that explicitly addressed *how* to teach CT, rather than just developing teachers' personal CT skills, were more successful in preparing educators for actual classroom implementation [13, 31].
- **User-Friendly Tools:** The widespread use of visual block-based programming languages like Scratch and App Inventor made it much easier for teachers with limited prior experience to get started with programming, making CT feel more approachable [47, 68].

DISCUSSION

Summary of Findings

This systematic literature review offers a comprehensive look at the current landscape of how computational thinking is being woven into teacher education. What we found is a dynamic and evolving field, characterized by a rich tapestry of teaching approaches and interventions. From direct lessons in dedicated courses to immersive, hands-on experiences with programming, robotics, unplugged activities, and integrated STEM learning, teacher education programs are using diverse strategies to equip educators with essential CT competencies. A consistent and truly encouraging theme across all the research we reviewed is the positive impact these interventions have had. Teachers' personal computational thinking skills are significantly boosted, their knowledge of *how to teach* CT (Pedagogical Content Knowledge, or PCK) grows, and their attitudes and confidence about bringing CT into their classrooms show marked improvement. While we saw a variety of assessment methods being used to measure CT in different educational settings, we also noted some ongoing concerns about consistently reporting on the validity and reliability of these tools. On the brighter side, the review clearly highlights common challenges, like teachers initially feeling unfamiliar with CT and facing time constraints, but also points to crucial facilitators, such as engaging, hands-on activities and clear connections to existing curricula.

The emphasis on practical, hands-on experiences, especially with programming and robotics, seems to be incredibly effective in developing both teachers' own CT abilities and their confidence in teaching these skills [4, 10, 32, 53]. These activities provide concrete ways to apply abstract CT concepts, making them much more accessible and meaningful. Unplugged activities are fantastic starting points, demystifying CT and making it approachable for teachers who might not have a computer science background [30, 35, 51]. These non-tech approaches help build foundational understanding and ease any initial worries about technology. Furthermore, integrating CT within existing subjects, particularly STEM, really shows its relevance and usefulness beyond just standalone computer science lessons [15, 52, 79]. This aligns perfectly with the bigger picture of CT as a universally applicable problem-solving approach, essential across all disciplines [86]. All these findings suggest that a balanced approach—combining theoretical learning with practical application and integrated disciplinary contexts—is the most effective way to help teachers develop comprehensive CT competence.

Implications for Teacher Education

The insights gleaned from this systematic review carry significant and actionable implications for how teacher education programs worldwide are designed, implemented, and continuously improved. To truly prepare a teaching workforce that is computationally literate, we need to focus on several key areas:

- **Rethinking Curriculum for Interdisciplinary Integration:** Teacher education programs should move beyond simply offering isolated computer science modules. Instead, they should actively consider embedding CT across various methods courses and within specific subject curricula. This interdisciplinary approach can help teachers see CT not as an extra burden, but as a versatile problem-solving tool that applies to diverse subjects, including mathematics [3], science [10], and even non-STEM fields like English or counseling [27]. This could involve designing lesson plans that explicitly highlight CT components within existing content standards.
- **Prioritizing Hands-on and Project-Based Learning:** Our review strongly suggests that practical, hands-on, and project-based learning experiences are absolutely crucial. Teacher education programs should prioritize activities that involve direct engagement with programming tools (both visual block-based like Scratch [47, 68] and text-based like Python [10]), robotics platforms (e.g., Lego Robots, Arduino [12, 53]), and physical computing projects (e.g., IoT applications [81]). These experiences not only build teachers' personal CT skills but also serve as powerful models for effective teaching strategies they can use in their own K-12 classrooms. Such approaches foster deeper understanding and confidence.
- **Explicitly Focusing on Pedagogical Content Knowledge (PCK) for CT:** It's not enough for teachers to just develop their own CT skills. Programs must explicitly address *how* to teach CT effectively. This means equipping teachers with the knowledge to scaffold CT concepts for diverse learners, design engaging and age-appropriate CT activities, and implement effective assessment methods for student learning in CT [13, 38, 50]. Frameworks like TPACK [40] can guide this development, ensuring teachers understand the crucial interplay between technology, teaching methods, and content in the context of CT.
- **Building Pathways for Continuous Professional Development:** For teachers already in service, ongoing and sustained professional development programs are essential. These programs should be designed to keep teachers up-to-date with evolving CT concepts, new educational technologies, and emerging best practices. Effective professional development should be practical, relevant to teachers' specific classroom contexts, and provide opportunities for collaborative learning and peer support [37, 38]. Micro-credentials, as explored by Bal et al. (2022) [8], could offer flexible and targeted professional learning opportunities.
- **Addressing Teacher Attitudes and Beliefs:** It's paramount to recognize and proactively address any initial anxieties, misconceptions, or lack of confidence teachers might have about computational thinking. Teacher education programs should incorporate strategies that foster positive attitudes toward CT and boost self-efficacy. Positive early experiences, clear demonstrations of CT's value and applicability, and supportive learning environments can significantly increase teachers' willingness and capacity to integrate CT effectively [8, 66].

- **Championing Unplugged CT Strategies:** While technology-driven activities are valuable, we shouldn't overlook the importance of unplugged activities. These strategies can serve as excellent entry points for introducing CT concepts without the initial intimidation of technology, making CT accessible to all teachers regardless of their prior programming experience. Teacher education should model and train educators in designing and implementing effective unplugged activities that build foundational CT understanding [30, 35].
- **Emphasizing Robust Assessment Practices:** Teacher education programs should model and encourage the use of valid and reliable assessment instruments for CT. This includes training teachers to critically evaluate existing assessment tools and, when necessary, to develop and validate their own context-specific assessments. Our review highlights a significant gap in the consistent reporting of validity and reliability, which needs to be addressed in both research and practice.

Strengths and Limitations of the Review

This systematic literature review has several key strengths that contribute to the reliability and completeness of its findings. A major strength is our systematic approach, which strictly followed established guidelines like PRISMA 2020 [59]. This ensured a rigorous, transparent, and unbiased process for finding, selecting, and synthesizing relevant research, making our findings more trustworthy and repeatable. Including a wide range of intervention types (like programming, robotics, unplugged activities, and integrated STEM) and diverse measured outcomes (such as CT skills, PCK, and attitudes) gave us a holistic and multi-faceted view of how CT is being integrated into teacher education. Plus, looking at studies from a broad timeframe (2010-2024) helped us capture recent trends and developments, reflecting how CT in education is constantly evolving. Our detailed data extraction and thematic synthesis process allowed for a rich, qualitative description of the various approaches and their reported effectiveness, offering nuanced insights beyond just simple numbers.

However, our review also has its limitations, which are important to keep in mind when interpreting the findings.

- **Diverse Studies:** The main limitation is that the studies we included were quite diverse. They varied in their research designs (some were experimental, some qualitative), the participants (pre-service vs. in-service, different subjects), the length and intensity of interventions, and the specific things they measured. This made it tough to do a quantitative meta-analysis and limits how definitively we can compare the effectiveness of different CT integration strategies.
- **Language Barrier:** We only looked at English-language, peer-reviewed articles. While practical for us, this might mean we missed valuable research published in other languages, especially from countries where significant CT integration efforts are happening.
- **Reporting Quality:** Even though we assessed quality, the level of detail and transparency in reporting varied across studies. Some articles didn't provide enough information on things like how they selected participants, how consistently they delivered the intervention, or how reliable their assessment tools were. This variability could affect how broadly our findings can be applied.
- **Focus on Empirical Studies:** Our criteria mainly focused on empirical research. This means we might have overlooked valuable insights from theoretical papers, policy documents, or practical reports that, while not empirical, could offer important perspectives on CT integration.
- **Publication Bias:** As mentioned in our methods, there's a general concern that studies with positive or statistically significant results might be more likely to get published than those with no clear findings [35]. While we tried to minimize this, it's an inherent challenge in reviewing existing literature.
- **Limited Long-Term Data:** Many of the studies were relatively short-term interventions. Our review highlights a lack of long-term studies that track the lasting impact of CT integration on teachers' teaching practices and, even more importantly, on their students' long-term CT development and academic outcomes.

Future Research Directions

Based on what we've found and the limitations we've identified in this systematic review, there are several exciting and crucial paths for future research. These will help us deepen our understanding of how to best integrate computational thinking into teacher education:

- **Long-Term Impact Studies:** We really need more longitudinal studies. These would follow teachers over time to see the lasting impact of CT integration programs on their teaching practices, their sustained knowledge of how to teach CT, and ultimately, on K-12 student learning. Such studies would give us invaluable insights into how effective different intervention models are in the long run.
- **Direct Impact on Student Learning:** Future research should explicitly explore the direct connections between teacher CT training and its effects on K-12 students' own computational thinking development, problem-solving skills, and academic success across various subjects. This means designing studies that measure student outcomes before and after their teachers have received CT training and started using new teaching methods.
- **CT in Diverse Subject Areas:** While our review showed a lot of research on CT integration in computer science

and STEM subjects, there's a critical need for more studies on how to integrate CT into non-STEM areas like humanities, arts, social studies, and language arts. Researchers should explore specific teaching strategies, relevant CT models, and appropriate tools for these disciplines to truly show CT's universal applicability.

- **Cross-Cultural and Diverse Contexts:** Research from a wider range of geographical, cultural, and socio-economic contexts would give us a more global picture of the challenges and successes in integrating CT into teacher education. This would help us identify both context-specific nuances and general principles that apply everywhere.
- **Teacher Collaboration and Communities:** Future research could investigate the role of collaborative learning, professional learning communities, and communities of practice among teachers in fostering CT integration. Understanding how peer support, mentorship, and shared experiences contribute to teachers' CT development and implementation is crucial.
- **Developing and Validating Strong Assessment Tools:** Given the assessment limitations we found, it's absolutely essential to continue developing and rigorously validating robust and reliable assessment tools for measuring CT in both teachers and students. This includes ensuring the tools are truly measuring what they're supposed to (construct validity), are consistent (reliability), and are practical to use in different settings and for different age groups [5, 42, 67, 72]. We also need more research on whether specific CT constructs are suitable for teacher education contexts.
- **Effectiveness of Combined Interventions:** Our review noted limited research on interventions that combine both plugged and unplugged activities. Future studies should systematically compare how effective these hybrid approaches are compared to using plugged or unplugged activities on their own, to figure out the best ways to integrate CT.
- **Non-Programming Tools and Approaches:** There's a need for more research on how to effectively use non-programming tools (like problem-solving tasks, storyboards, or physical manipulatives) for CT integration, especially in situations where programming skills might be a barrier or less relevant.
- **Teacher Readiness and Beliefs:** Further qualitative and mixed-methods research could dive deeper into how teachers' beliefs, perceptions, and readiness for CT integration evolve. This would explore the factors that influence whether they adopt and consistently use CT teaching methods.

CONCLUSION

Integrating computational thinking into teacher education isn't just a good idea—it's a fundamental necessity if we want to prepare educators who can equip the next generation with those crucial 21st-century skills. This systematic review clearly shows that there are many effective strategies out there, from direct programming lessons and robotics to unplugged activities and interdisciplinary STEM approaches. By focusing on helping teachers develop both their own personal CT skills and their knowledge of how to teach CT, teacher education programs can cultivate a teaching force that's truly capable of nurturing computational literacy in all students. The consistent positive outcomes we've seen across studies—like improved CT skills, better teaching knowledge, and more positive attitudes—provide a strong foundation for continuing to invest in this area. While challenges like time constraints and assessment validity still exist, the helpful factors we've identified offer clear paths for successful implementation. Ongoing research, especially focusing on long-term impacts, diverse contexts, and developing reliable assessment tools, will continue to refine our understanding and enhance the effectiveness of these vital educational efforts, ultimately helping to build a more computationally literate society for everyone.

REFERENCES

1. Abouelenein, Y. A. M., & Nagy Elmaadaway, M. A. (2023). Impact of teaching a neuro-computerized course through VLE to develop computational thinking among mathematics pre-service teachers. *Journal of Educational Computing Research*, *61*(6), 1175–1206. https://doi.org/10.1177/07356331231165099
2. Adler, R. F., & Kim, H. (2018). Enhancing future K-8 teachers' computational thinking skills through modeling and simulations. *Education and Information Technologies*, *23*(4), 1501–1514. https://doi.org/10.1007/s10639-017-9675-1
3. Aminah, N., Sukestiyarno, Y. L., Wardono, W., & Cahyono, A. N. (2022). Computational thinking process of prospective mathematics teacher in solving diophantine linear equation problems. *European Journal of Educational Research*, *11*(3), 1495–1507. https://doi.org/10.12973/eu-jer.11.3.1495
4. Angeli, C. (2022). The effects of scaffolded programming scripts on pre-service teachers' computational thinking: developing algorithmic thinking through programming robots. *International Journal of Child-Computer Interaction*, *31*, 1–20. https://doi.org/10.1016/j.ijcci.2021.100329
5. Aristiz´abal Zapata, J. H., Guti´errez Posada, J. E., & Diago, P. D (2024). Design and validation of a computational thinking test for children in the first grades of elementary education. *Multimodal Technologies and Interaction*, *8*(5), 39.

https://doi.org/10.3390/mti8050039

6. Atmaja, J. F. T., Sari, M. W., & Ciptadi, P. W. (2021). Developing application of automatic lamp control and monitoring system using internet of things. *Journal of Physics: Conference Series*, *1823*(1), Article 012002. https://doi.org/10.1088/1742-6596/1823/1/012002
7. Bai, H., Bosch, C., Goosen, L., & Chetty, J. (2024). Introducing computational thinking and coding to teacher education students. *Navigating computer science education in the 21st century* (pp. 170–186). IGI Global. https://doi.org/10.4018/979-8-3693-1066-3.ch009
8. Bal, I. A., Alvarado–Albertorio, F., Marcelle, P., & Oaks–Garcia, C. T (2022). Pre–service teachers computational thinking (CT) and pedagogical growth in a micro–credential: a mixed methods study. *TechTrends*, *66*(3), 468–482. https://doi.org/10.1007/s11528-022-00732-x
9. 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
10. Bati, K. (2022). Integration of python into science teacher education, developing computational problem solving and using information and communication technologies competencies of pre-service science teachers. *Informatics In Education*, *21*(2), 235–251. https://doi.org/10.15388/infedu.2022.12
11. Brennan, K., & Resnick, M. (2012). *New frameworks for studying and assessing the development of computational thinking*. Proceedings of the American Educational Research Association (AERA). Vancouver, Canada.
12. Budiyanto, C. W., Fenyvesi, K., Lathifah, A., & Yuana, R. A. (2022). Computational thinking development: benefiting from educational robotics in stem teaching. *European Journal of Educational Research*, *11*(4), 1997–2012. https://doi.org/10.12973/eu-jer.11.4.1997
13. Çakiroglu, Ü., & Kiliç, S. (2023). Assessing teachers' PCK to teach computational thinking via robotic programming. *Interactive Learning Environments*, *31*(2), 818–835. https://doi.org/10.1080/10494820.2020.1811734
14. Chen, G., Shen, J., Barth-Cohen, L., Jiang, S., Huang, X., & Eltoukhy, M. (2017). Assessing elementary students' computational thinking in everyday reasoning and robotics programming. *Computers & Education*, *109*, 162–175. https://doi.org/10.1016/j.compedu.2017.03.001
15. Çiftçi, A., & Topçu, M. (2023). Improving early childhood pre-service teachers' computational thinking skills through the unplugged computational thinking integrated STEM approach. *Thinking Skills and Creativity*, *49*, 1–14. https://doi.org/10.1016/j.tsc.2023.101337
16. Cohen, L., Manion, L., & Morrison, K. (2018). *Research methods in education* (8th ed.). Routledge.
17. Cooper, H. (2017). *Research synthesis and meta-analysis*. SAGE Publications, Inc. https://doi.org/10.4135/9781071878644
18. Czerkawski, B.C., Lyman, E.W. (2015) Exploring issues about computational thinking in higher education. *TechTrends*, *59*, 57–65. https://doi.org/10.1007/s11528-015-0840-3.
19. Daungjun, S. (2018). *Effects of using stem education in physics on computational thinking ability of upper secondary school students*. Bangkok: Chulalongkorn University.
20. de Jong, I., & Jeuring, J. (2020). Computational thinking interventions in higher education: A scoping literature review of interventions used to teach computational thinking. In *Proceedings of the 20th Koli Calling International Conference on Computing Education Research* (pp. 1–10). ACM. https://doi.org/10.1145/3428029.3428055.
21. Demchuk, E., Ruiz, P., Wilson, J. D., Scinicariello, F., Pohl, H. R., Fay, M., Mumtaz, M. M., Hansen, H., & De Rosa, C. T. (2008). Computational toxicology methods in public health practice. *Toxicology Mechanisms and Methods*, *18*(2–3), 119–135. https://doi.org/10.1080/15376510701857148
22. El-Hamamsy, L., Chessel-Lazzarotto, F., Bruno, B., Roy, D., Cahlikova, T., Chevalier, M., Parriaux, G., Pellet, J., Lanar`es, J., Zufferey, J., & Mondada, F. (2021). A computer science and robotics integration model for primary school: evaluation of a large-scale in-service K-4 teacher-training program. *Education and Information Technologies*, *26*(3), 2445–2475. https://doi.org/10.1007/s10639-020-10355-5
23. Fessakis, G., & Prantsoudi, S. (2019). Computer science teachers' perceptions, beliefs and attitudes on computational thinking in Greece. *Informatics in Education*, *18*(2), 227–258. https://doi.org/10.15388/infedu.2019.11
24. Gabriele, L., Bertacchini, F., Tavernise, A., Vaca-Cardenas, ´ L., Pantano, P., & Bilotta, E. (2019). Lesson planning by computational thinking skills in Italian pre-service teachers. *Informatics In Education*, *18*(1), 69–104. https://doi.org/10.15388/infedu.2019.04
25. Gandotra, E., Bansal, D., & Sofat, S. (2015). Computational techniques for predicting cyber threats. In S. C. Satapathy, B. Mandal, & R. R. Mohanty (Eds.), *Intelligent computing, communication and devices* (pp. 247–253). Springer. https://doi.org/10.1007/978-81-322-2012-1_26.
26. 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

27. Haslaman, T., Mumcu, F., & Uslu, N. (2024). Fostering computational thinking through digital storytelling: A distinctive approach to promoting computational thinking skills of pre-service teachers. *Education And Information Technologies*, *29*(14), 18121–18147. https://doi.org/10.1007/s10639-024-12583-5
28. Helsa, Y., & Juandi, D. (2023). TPACK-based hybrid learning model design for computational thinking skills achievement in mathematics. *Journal on Mathematics Education*, *14*(2), 225–252. https://doi.org/10.22342/jme.v14i2.pp225-252
29. Hickmott, D., Prieto-Rodriguez, E., & Holmes, K. (2018). A scoping review of studies on computational thinking in K–12 mathematics classrooms. *Digital Experiences in Mathematics Education*, *4*(1), 48–69. https://doi.org/10.1007/s40751-017-0038-8
30. Hunsaker, E., & West, R. E. (2020). Designing computational thinking and coding badges for early childhood educators. *TechTrends*, *64*(1), 7–16. https://doi.org/10.1007/s11528-019-00420-3
31. ISTE. (2011). *Computational thinking*. https://iste.org/computational-thinking.
32. Jaipal-Jamani, K., & Angeli, C. (2017). Effect of robotics on elementary preservice teachers' self-efficacy, science learning, and computational thinking. *Journal of Science Education and Technology*, *26*(2), 175–192. https://doi.org/10.1007/s10956-016-9663-z
33. Kakavas, P., & Ugolini, F. C. (2019). Computational thinking in primary education: A systematic literature review. *Research on Education and Media*, *11*(2), 64–94. https://doi.org/10.2478/rem-2019-0023
34. Kampylis, P., Dagiene, ˙ V., Bocconi, S., Chioccariello, A., Engelhardt, K., Stupuriene, ˙ G., & Earp, J. (2023). Integrating computational thinking into primary and lower secondary education. *Educational Technology & Society*, *26*(2), 99–117. [suspicious link removed].
35. Keen, B., Blaszczynski, A., & Anjoul, F. (2017). Systematic review of empirically evaluated school-based gambling education programs. *Journal of Gambling Studies*, *33*(1), 301–325. https://doi.org/10.1007/s10899-016-9641-7
36. Kilic, S., & Çakiroglu, Ü. (2023). Design, implementation, and evaluation of a professional development program for teachers to teach computational thinking via robotics. *Technology Knowledge And Learning*, *28*(4), 1539–1569. https://doi.org/10.1007/s10758-022-09629-3
37. Kite, V., & Park, S. (2022). Preparing inservice science teachers to bring unplugged computational thinking to their students. *Teaching and Teacher Education*, *120*, Article 103904. https://doi.org/10.1016/j.tate.2022.103904
38. Kong, S., Lai, M., & Li, Y. (2023). Scaling up a teacher development programme for sustainable computational thinking education: TPACK surveys, concept tests and primary school visits. *Computers & Education*, *194*, 1–17. https://doi.org/10.1016/j.compedu.2022.104707
39. Kong, S., Lai, M., & Sun, D. (2020). Teacher development in computational thinking: design and learning outcomes of programming concepts, practices and pedagogy. *Computers & Education*, *151*, 1–19. https://doi.org/10.1016/j.compedu.2020.103872
40. Kong, S.-C., & Lai, M. (2022). A proposed computational thinking teacher development framework for K-12 guided by the TPACK model. *Journal of Computers in Education*, *9*(3), 379–402. https://doi.org/10.1007/s40692-021-00207-7
41. Kong, S. C., & Lao, A. C. C. (2019). Assessing in-service teachers' development of computational thinking practices in teacher development courses (pp. 976–982). Association for Computing Machinery. https://doi.org/10.1145/3287324.3287415
42. Korkmaz, O., ¨ Çakir, R., & Ozden, ¨ M. Y. (2017). A validity and reliability study of the computational thinking scales (CTS). *Computers in Human Behavior*, *72*, 558–569. https://doi.org/10.1016/j.chb.2017.01.005
43. Lee, S. J., Francom, G. M., & Nuatomue, J. (2022). Computer science education and K-12 students' computational thinking: A systematic review. *International Journal of Educational Research*, *114*, Article 102008. https://doi.org/10.1016/j.ijer.2022.102008
44. Looi, C. K., How, M. L., Longkai, W., Seow, P., & Liu, L. (2018). Analysis of linkages between an unplugged activity and the development of computational thinking. *Computer Science Education*, *28*(3), 255–279.
45. Lu, C., Macdonald, R., Odell, B., Kokhan, V., Demmans Epp, C., & Cutumisu, M. (2022). A scoping review of computational thinking assessments in higher education. *Journal of Computing in Higher Education*, *34*(2), 416–461. https://doi.org/10.1007/s12528-021-09305-y
46. Lyon, J. A., & J Magana, A. (2020). Computational thinking in higher education: A review of the literature. *Computer Applications in Engineering Education*, *28*(5), 1174–1189. https://doi.org/10.1002/cae.22295
47. Molina-Ayuso, A., Adamuz-Povedano, N., Bracho-Lopez, ´ R., & Torralbo-Rodríguez, M. (2022). Introduction to computational thinking with scratch for teacher training for spanish primary school teachers in mathematics. *Education Sciences*, *(12)*, 12. https://doi.org/10.3390/educsci12120899
48. Monjelat, N., & Lantz-Andersson, A. (2020). Teachers' narrative of learning to program in a professional development effort and the relation to the rhetoric of computational thinking. *Education and Information Technologies*, *25*(3), 2175–2200. https://doi.org/10.1007/s10639-019-10048-8

49. Moreno-Leon, ´ J., Roman-Gonz ´ alez, ´ M., Harteveld, C., & Robles, G. (2017). On the automatic assessment of computational thinking skills: a comparison with human experts. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems* (pp. 2788–2795). https://doi.org/10.1145/3027063.3053216

50. Mouza, C., Yang, H., Pan, Y., Ozden, S., & Pollock, L. (2017). Resetting educational technology coursework for pre-service teachers: a computational thinking approach to the development of technological pedagogical content knowledge (TPACK). *Australasian Journal of Educational Technology*, *33*(3), 61–76. https://doi.org/10.14742/ajet.3521

51. Mumcu, F., Kıdıman, E., & Ozdinç, ¨ F. (2023a). Integrating computational thinking into mathematics education through an unplugged computer science activity. *Journal of Pedagogical Research*, *7*(2), 72–92. https://doi.org/10.33902/JPR.202318528

52. Mumcu, F., Uslu, N., & Yildiz, B. (2023b). Teacher development in integrated STEM education: design of lesson plans through the lens of computational thinking. *Education And Information Technologies*, *28*(3), 3443–3474. https://doi.org/10.1007/s10639-022-11342-8

53. Nannim, F. A., Ibezim, N. E., Oguguo, B. C. E., & Nwangwu, E. C. (2024). Effect of project-based Arduino robot application on trainee teachers computational thinking in robotics programming course. *Education and Information Technologies*, *29*(10), 13155–13170. https://doi.org/10.1007/s10639-023-12380-6

54. National Research Council. (2010). *Report of a workshop on the scope and nature of computational thinking*. National Academies Press. https://doi.org/10.17226/12840, 12840.

55. National Research Council. (2012). *A framework for K-12 science education: practices, crosscutting concepts, and core ideas*. National Academies Press. https://doi.org/10.17226/13165, 13165.

56. Navarro, E., Costa, N., & Pereira, A. (2020). A systematic review of iot solutions for smart farming. *Sensors*, *20*(15), 4231. https://doi.org/10.3390/s20154231

57. Ogegbo, A. A., & Ramnarain, U. (2022). A systematic review of computational thinking in science classrooms. *Studies in Science Education*, *58*(2), 203–230. https://doi.org/10.1080/03057267.2021.1963580

58. Ozdinç, ¨ F., Kayab, G., Mumcu, F., & Yildiz, B. (2022). Integration of computational thinking into STEM activities: an example of an interdisciplinary unplugged programming activity. *Science Activities-Projects and Curriculum Ideas in Stem Classrooms*, *59*(3), 151–159. https://doi.org/10.1080/00368121.2022.2071817

59. Page, M. J., McKenzie, J. E., Bossuyt, P. M., Boutron, I., Hoffmann, T. C., Mulrow, C. D., Shamseer, L., Tetzlaff, J. M., Akl, E. A., Brennan, S. E., Chou, R., Glanville, J., Grimshaw, J. M., Hrobjartsson, ´ A., Lalu, M. M., Li, T., Loder, E. W., Mayo-Wilson, E., McDonald, S., McGuinness, L. A., Stewart, L. A., Thomas, J., Tricco, A. C., Welch, V. A., Whiting, P., & Moher, D. (2021). The PRISMA 2020 statement: an updated guideline for reporting systematic reviews. *BMJ (Clinical research ed.)*, *372*. https://doi.org/10.1136/bmj.n71. n71.

60. Pala, F. K., & Mıhcı Türker, P. (2021). The effects of different programming trainings on the computational thinking skills. *Interactive Learning Environments*, *29*(7), 1090–1100. https://doi.org/10.1080/10494820.2019.1635495

61. Palts, T., & Pedaste, M. (2020). A model for developing computational thinking skills. *Informatics in Education*, *19*(1), 113–128. https://doi.org/10.15388/infedu.2020.06

62. Peracaula-Bosch, M., & Gonz´alez-Martínez, J. (2022). Developing computational thinking among pre-service teachers. *Qwerty*, *17*(1), 28–44. https://doi.org/10.30557/QW000049

63. Pewkam, W., & Chamrat, S. (2022). Pre-service teacher training program of stem-based activities in computing science to develop computational thinking. *Informatics In Education*, *21*(2), 311–329. https://doi.org/10.15388/infedu.2022.09

64. Pimdee, P., & Pipitgool, S. (2023). Promoting undergraduate pre-service teacher computational thinking. *TEM Journal-Technology Education Management Informatics*, *12*(1), 540–549. https://doi.org/10.18421/TEM121-64

65. Ragonis, N., Rosenberg-Kima, R., & Hazzan, O. (2024). A computational thinking course for all preservice K-12 teachers: implementing the four pedagogies for developing computational thinking (4P4CT) framework. *Educational Technology Research and Development*. https://doi.org/10.1007/s11423-024-10406-5

66. Rich, P. J., Larsen, R. A., & Mason, S. L. (2021). Measuring teacher beliefs about coding and computational thinking. *Journal of Research on Technology in Education*, *53*(3), 296–316. https://doi.org/10.1080/15391523.2020.1771232

67. Rom´an-Gonz´alez, M., P´erez-Gonz´alez, J. C., Moreno-Leon, ´ J., & Robles, G. (2018). Can computational talent be detected? Predictive validity of the Computational Thinking Test. *International Journal of Child-Computer Interaction*, *18*, 47–58.

68. S´aez-Lopez, ´ J. M., Del Olmo-Munoz, ˜ J., Gonz´alez-Calero, J. A., & Cozar-Guti ´ ´errez, R. (2020). Exploring the effect of training in visual block programming for preservice teachers. *Multimodal Technologies and Interaction*, *4*(3), 65. https://doi.org/10.3390/mti4030065

69. Schafer, ¨ M. S., & Hase, V. (2023). Computational methods for the analysis of climate change communication: towards an integrative and reflexive approach. *WIREs Climate Change*, *14*(2), e806. https://doi.org/10.1002/wcc.806
70. Schuch, F. B., Vancampfort, D., Richards, J., Rosenbaum, S., Ward, P. B., & Stubbs, B. (2016). Exercise as a treatment for depression: A meta-analysis adjusting for publication bias. *Journal of Psychiatric Research*, *77*, 42–51. https://doi.org/10.1016/j.jpsychires.2016.02.023
71. Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review*, *22*, 142–158. https://doi.org/10.1016/j.edurev.2017.09.003
72. Sovey, S., Osman, K., & Mohd-Matore, M. E. E. (2022). Exploratory and confirmatory factor analysis for disposition levels of computational thinking instrument among secondary school students. *European Journal of Educational Research*, *11*(2), 639–652. https://doi.org/10.12973/eu-jer.11.2.639
73. Sun, L., Guo, Z., & Zhou, D. (2022). Developing K-12 students' programming ability: A systematic literature review. *Education and Information Technologies*, *27*(5), 7059–7097. https://doi.org/10.1007/s10639-022-10891-2
74. Sun, L., & Liu, J. (2024a). A gender differential analysis of educational robots' effects on primary teachers' computational thinking: mediating effect of programming attitudes. *Education and Information Technologies*, *29*(15), 19753–19782. https://doi.org/10.1007/s10639-024-12655-6
75. Sun, L., & Liu, J. (2024b). Micro: bit programming effects on elementary STEM teachers' computational thinking and programming attitudes: A moderated mediation model. *Journal of Research on Technology in Education*, 1–23. https://doi.org/10.1080/15391523.2024.2402357
76. Sun, L., You, X., & Zhou, D. (2023). Evaluation and development of STEAM teachers' computational thinking skills: analysis of multiple influential factors. *Education and Information Technologies*, *28*(11), 14493–14527. https://doi.org/10.1007/s10639-023-11777–7
77. Sung, Y.-H., & Jeong, Y.-S. (2019). Development and application of programming education model based on visual thinking strategy for pre-service teachers. *Universal Journal of Educational Research*, *7*(5), 42–53. https://doi.org/10.13189/ujer.2019.071507
78. Tankiz, E., & Atman Uslu, N. (2023). Preparing pre-service teachers for computational thinking skills and its teaching: a convergent mixed-method study. *Technology, Knowledge and Learning*, *28*(4), 1515–1537. https://doi.org/10.1007/s10758-022-09593-y
79. Tripon, C. (2022). Supporting future teachers to promote computational thinking skills in teaching stem-a case study. *Sustainability*, *14*(19). https://doi.org/10.3390/su141912663
80. Tsai, F. (2023). Using a physical computing project to prepare preservice primary teachers for teaching programming. *Sage Open*, *13*(4). https://doi.org/10.1177/21582440231205409
81. Tsai, F. (2024). Development and evaluation of an internet of things project for preservice elementary school teachers. *Sustainability*, *(17)*, 16. https://doi.org/10.3390/su16177632
82. 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
83. Umutlu, D. (2022). An exploratory study of pre-service teachers' computational thinking and programming skills. *Journal of Research on Technology in Education*, *54*(5), 754–768. https://doi.org/10.1080/15391523.2021.1922105
84. Uzumcu, O., & Bay, E. (2020). The effect of computational thinking skill program design developed according to interest driven creator theory on prospective teachers. *Education and Information Technologies*, *26*, 565–583. https://doi.org/10.1007/s10639-020-10268-3
85. Voon, X. P., Wong, S. L., Wong, L. H., Khambari, M. N. M., & Syed-Abdullah, S. I. S. (2023). Developing pre-service teachers' computational thinking through experiential learning: hybridisation of plugged and unplugged approaches. *Research and Practice in Technology Enhanced Learning*, *18*. https://doi.org/10.58459/rptel.2023.18006
86. Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33–35. https://doi.org/10.1145/1118178.1118215
87. Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *366*(1881), 3717–3725. https://doi.org/10.1098/rsta.2008.0118
88. Yadav, A., & Berges, M. (2019). Computer science pedagogical content knowledge: characterizing teacher performance. *ACM Transactions on Computing Education*, *19*(3), 1–24. https://doi.org/10.1145/3303770
89. Yadav, A., Hong, H., & Stephenson, C. (2016). Computational thinking for all: pedagogical approaches to embedding 21st century problem solving in K-12 classrooms. *TechTrends*, *60*(6), 565–568. https://doi.org/10.1007/s11528-016-0087-7
90. Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J. (2014). Computational thinking in elementary and secondary teacher education. *ACM Transactions on*

Computing Education, *14*(1). https://doi.org/10.1145/2576872

91. Yun, M., & Crippen, K. J. (2024). Computational thinking integration into Pre-service science teacher education: A systematic review. *Journal of Science Teacher Education*, *36*(2), 1–30. https://doi.org/10.1080/1046560X.2024.2390758
92. Zha, S., Jin, Y., Moore, P., & Gaston, J. (2020a). A cross-institutional investigation of a flipped module on preservice teachers' interest in teaching computational thinking. *Journal of Digital Learning in Teacher Education*, *36*(1), 32–45. https://doi.org/10.1080/21532974.2019.1693941
93. Zha, S., Jin, Y., Moore, P., & Gaston, J. (2020b). Hopscotch into coding: introducing pre-service teachers computational thinking. *TechTrends*, *64*(1), 17–28. https://doi.org/10.1007/s11528-019-00423-0