Journal of Research in Science, Mathematics and Technology Education

https://jrsmte.com/

Exploring Fifth-Grade Teachers' Integration of Computational Thinking, Instruction Strategies, and Efficacy Following Professional Development

Susan German

University of Missouri, Columbia, United States

Abstract: This study examined how fifth-grade teachers integrated computational thinking (CT) following participation in the eSTEM for Designing Games for Education (eDGE) professional development program. Using a qualitative-dominant mixed methods design, data were collected from two cohorts of teachers across multiple states and included qualitative artifacts, interviews, survey measures of self-efficacy, and Computational Thinking Pattern Analysis (CTPA) of coded projects. Findings indicate that teachers integrated CT primarily through interdisciplinary, student-centered instruction using simulation-based and problem-based approaches. Teachers demonstrated growth in foundational computational thinking practices—such as decomposition and algorithmic reasoning—alongside moderate increases in coding self-efficacy. However, more advanced computational concepts, including abstraction and dynamic system modeling, remained challenging, with minimal gains in computational thinking efficacy. Integrated findings reveal a patterned relationship between efficacy and proficiency: convergence in foundational, structured aspects of CT and divergence in more complex computational practices. These results support a partial conceptualization of computational thinking, in which teachers emphasize cognitive problem-solving processes without consistently translating them into computationally executable forms. Implications suggest that professional development should more explicitly connect problem-solving strategies to computational implementation and provide sustained opportunities for teachers to engage with complex computational practices.

Keywords: Coding; Computational Thinking; Mathematics Education; Professional Development; Teacher Efficacy
DOI: https://doi.org/10.31756/jrsmte.512SI

Introduction and Background Information

Computational Thinking in a Digital Society

Contemporary culture is increasingly shaped by digital systems governed by software (Manovich, 2013). Rushkoff (2012) captures this reality with the warning that individuals must "program or be programmed," while Gardner (2014) characterizes society as increasing "app-enabled or app-dependent." In response to these shifts, Missouri enacted Senate Bill 894 in 2018, establishing the Missouri Computer Science Education Program and encouraging the integration of computer science (CS) concepts across subject areas, particularly in grades K–8 (Mo. Rev. Stat. § 170.018, 2018). The legislation emphasizes teacher professional development, curriculum alignment, and expanding access to computing experiences for students statewide.

Computational thinking (CT) has emerged as a basic cognitive skill in a technology-driven society. It involves approaching complex problems systematically by decomposing them into smaller components, identifying patterns,

abstracting key features, and developing algorithmic solutions (Brennan & Resnick, 2012; Grover & Pea, 2013; Weintrop et al., 2016; Wing, 2006). CT should be transferable beyond programming itself and applied to many types of problems that do not involve writing code (Wing, 2008). Although CT originated in CS, its applications extend across disciplines, including mathematics, science, engineering, and social sciences.

Acquisition of CT Components

Although CT is often defined through components such as decomposition, pattern recognition, and algorithmic reasoning, research suggests that teachers' confidence in applying these components remains uneven (Rich et al., 2021). This variability highlights a critical challenge: understanding CT conceptually does not necessarily translate into effective instructional practice.

Scalable Game Design as an Approach to Learning CT

Despite the increasing importance of computing, computer science has continued to face challenges related to public perception and participation. Between 2000 and 2004, enrollment in undergraduate CS programs in the United States declined by approximately 60% (NSF/NCES, 2017). Factors contributing to low enrollment include limited exposure to computing in K–12 education, the absence of uniform CS learning standards across states, and narrow curricular offerings that emphasized syntax-heavy programming with few motivating real-world applications. Repenning and Ioannidou (2008) proposed that addressing the "pipeline problem" requires moving the focus from traditional programming instruction toward design-centered learning environments. Drawing on constructionist approaches and game-based learning research (Kafai, 2006), Repenning and Ioannidou argued that game design can provide meaningful contexts that motivate learners while embedding fundamental CT concepts. By embedding computing into school curricula rather than limiting it to elective or extracurricular programs, Scalable Game Design (SGD) sought to broaden participation and reach students who might otherwise have limited access to CS experiences (Gootman, 2007). This integrated approach positions CT as a conceptual foundation for learning that can connect to disciplinary content and authentic problem-solving.

Built on this perspective, the Scalable Game Design (SGD) "project first" approach integrates motivation theory, structured competency frameworks, and accessible authoring tools to support progressive learning. First, SGD attends to motivation through carefully designed challenges intended to sustain learners' engagement and support an optimal "flow" state (Csikszentmihalyi, 1990). Second, competency frameworks emphasize CT skills such as abstraction, decomposition, and algorithmic reasoning rather than mastery of a specific programming language (CSTA & ISTE, 2011; Wing, 2006). Third, authoring tools such as AgentSheets and related platforms provide environments that are accessible for novices while still capable of supporting complex games and simulations (Ioannidou et al., 2003; Repenning et al., 2010). This project-first design also served a pedagogical purpose: teachers experienced learning conditions like those they could replicate with students—guided discovery, incremental complexity, and iterative debugging.

Computational Thinking Pattern Analysis

Computational Thinking Pattern Analysis (CTPA) is a complementary analytic tool focused on coded project artifacts. In SGD, CT patterns (e.g., diffusion, collision, generation) are intended to bridge game design to simulation design. For example, programming for diffusion movement can create a simulation for how particles of liquids move as well as how the ghost of a PacMan style game moves (Ioannidou et al., 2011). CTPA produces comparable outputs even when projects differ structurally (e.g., number of agents, scoring behaviors) because the analysis targets the presence and strength of predefined computational patterns. In this study, CTPA outputs supported interpretation of which patterns teachers successfully implemented in their code and whether those outputs aligned with tutorial expectations. This provided an additional perspective on learning CT-related concepts, a measure of computational proficiency, beyond self-report. Computational proficiency refers to teachers' demonstrated ability to implement CT in coded artifacts, as evidenced by the presence, accuracy, and complexity of CT patterns identified through CTPA.

Problem Statement

Over the past decade, research on CT has expanded, encompassing conceptual definitions, tool development, instructional strategies, and assessment approaches (Berland & Wilensky, 2015; Selby & Woollard, 2013). Review studies have documented trends in CT research, including participant demographics, instructional tools, and methodological approaches (Kalelioglu et al., 2016; Lockwood & Mooney, 2017; Rodrigues et al., 2024; Taslibeyaz et al., 2020). Many studies have examined how CT can be embedded in classroom activities or supported by technological platforms (Grover & Pea, 2013). However, comparatively little research has focused on educators themselves—specifically, how teachers understand CT, how confident they feel implementing it, and how professional learning shapes classroom practice (Portelance & Bers, 2015; Voogt et al., 2015; Wilson & Guzdial, 2010; Yadav et al., 2011).

In addition to gaps in understanding how teachers enact CT, there is limited research examining how professional development influences teachers' sense of efficacy in implementing CT practices (Rich et al., 2021; Yurdakok & Kaleioglu, 2024). To address these gaps, this study examines how fifth-grade teachers conceptualize and enact CT during professional development, with attention to their instructional choices, areas of challenge, and development of teacher efficacy.

Purpose of Study

The purpose of this qualitative-dominant mixed-methods study was to explore how fifth-grade teachers plan for and integrate computational thinking (CT) into their instruction while participating in professional development. In this concurrent qualitative-dominant design (QUAL + quan), qualitative data served as the primary source of evidence, while quantitative data provided a secondary, complementary strand. The qualitative and quantitative data were collected concurrently, analyzed separately, and subsequently integrated during the final interpretation phase to provide a comprehensive understanding of the phenomenon.

The primary qualitative strand was prioritized to provide in-depth insight into teachers' internal planning processes, instructional enactment, and lived learning experiences as they integrated CT into their classrooms. The secondary quantitative strand examined broader contextual shifts, specifically changes in how teachers learned CT concepts via Computational Thinking Pattern Analysis (CTPA) data and changes in teacher efficacy using survey data. By nesting the secondary quantitative metrics within a dominant qualitative framework, this approach leverages the depth of teacher narratives while utilizing complementary statistical evidence to capture growth. This combined approach provides a more complete understanding of how professional development supports CT integration, informing the design of future professional development models and implementation guidance.

Theoretical Framework

The implementation of this study is grounded in Bandura's theory of self-efficacy, which defines efficacy beliefs as individuals' judgments about their capabilities to organize and execute actions required to achieve specific goals (Bandura, 1994). These beliefs influence motivation, persistence, emotional responses, and performance, and therefore may influence whether new instructional practices are adopted and sustained. Rittmayer and Beier (2008) further specify that self-efficacy judgments are domain-specific and tied to tasks. Within education, teaching efficacy refers to teachers' beliefs in their capacity to promote student learning and engagement, including among students who are disengaged or struggling (Bandura, 1977; Tschannen-Moran & Hoy, 2001).

Bandura (1994) identifies four sources of self-efficacy: mastery experiences, vicarious experiences, social persuasion, and the regulation of physiological and emotional responses. Self-efficacy develops through a feedback cycle in which beliefs influence effort and persistence, shaping performance outcomes that, in turn, inform future efficacy beliefs. Over time, this process stabilizes unless disrupted by new experiences (Tschannen-Moran & McMaster, 2009). Täschner et al. (2025), in a review and meta-analysis of 115 studies, found both pre-service and in-service teachers increased teacher efficacy when they engaged in mastery experiences that included reflection. Consistent with this framework, effective professional development should be responsive to teachers' needs and emphasize content, curriculum, and instructional support (Lee et al., 2020).

Teaching efficacy can vary across contexts and tasks, and it is influenced by personal characteristics, classroom conditions, and institutional support (Fackler & Malmberg, 2016). Research indicates that higher teacher self-efficacy is associated with improved student achievement, stronger instructional practices, greater professional commitment, and reduced burnout (Chesnut & Burley, 2015; Eells, 2011; Klassen & Tze, 2014; Shoji et al., 2015; Zee & Koomen, 2016). Accordingly, professional development aimed at strengthening teachers' CT and coding expertise is expected to increase teachers' confidence and likelihood of implementing computational practices in elementary classrooms. In the context of CT, teaching efficacy may influence how teachers design lessons, select instructional strategies, and persist through challenges associated with coding and technology integration.

Methods

Study Design

The eSTEM Designing Games for Education (eDGE) project, funded through the Education Innovation and Research program of the U.S. Department of Education and implemented by enhancing Missouri Instructional Networked Teaching Strategies (eMINTS), served as the study context. Teachers participating in eDGE learned to connect CT practices to mathematical concepts such as probability, proportional reasoning, statistical reasoning, and the use of variables within artifacts. Data were collected across two cohorts at different stages of program completion. This timing enabled comparison of teacher learning and enactment across cohorts at different stages while leveraging existing program artifacts as a substantial data source. The first cohort of participants had completed both courses, and the second cohort was nearing completion of Course 1 and preparing to begin Course 2 at the start of 2024.

Professional Development Context and Materials

The instructional context for eDGE was grounded in principles established through the Scalable Game Design (SGD) research program. Within the broader eDGE project goals, teachers were expected to:

- teach CT and CS concepts through 3D game creation and adapt those ideas to science simulations using AgentCubes Online;
- align CT objectives with existing curriculum and state/national standards in CS, mathematics, and science;
- develop and implement student-centered lessons across instructional modalities;
- use CT assessment tools such as CTPA to interpret student learning; and
- collect student feedback data related to motivation and continued engagement in CS/STEM learning.

Table 1 summarizes participation across data sources.

Table 1

Participant Counts by Data Source.

Data Source	Group 1	Group 2
Number of eDGE participants	20	26
Discussion Board Posts (from archived courses)	20	26
Lesson Plans (teacher consent)	2	6
Interviews (teacher consent)	2	6
CTPA tool (teacher consent)	2	11
TBACCT responses (teacher consent)	9	24
Number of participating teachers in all five sources	1	4

eDGE Course 1

Course 1 focused on foundational skills in Scalable Game Design using AgentCubes Online. They progressed through structured activities that introduced CT patterns while building and modifying a basic game. Course activities included transforming a narrative into code, building a Frogger-style game, and exploring mathematical ideas such as probability and variable use within game behavior. Teachers practiced troubleshooting and debugging strategies and were prompted to consider classroom translation, including how to scaffold student work and how to connect gameplay behaviors to mathematical reasoning. Teachers then built a PacMan-style game that introduced more advanced techniques. Across the course, tutorials were scaffolded to increase teachers' coding and computational thinking skills (Table 2). CTPA was used to identify which patterns teachers successfully implemented in their code and whether those outputs aligned with tutorial expectations.

Table 2

Comparison of Frogger, PacMan, and Advanced PacMan Tutorials

Category	**Frogger**	**PacMan**	**Advanced PacMan**
Game Focus	Obstacle navigation (crossing roads/rivers)	Maze navigation (avoid ghosts, collect pellets)	Agent-based modeling gameplay (ghosts track PacMan)
CT Patterns	Cursor control, collision, generate, absorb, basic probability	Cursor control, collision, absorption	Diffusion, hill climbing, polling, variables, modeling
Prerequisite Skills	Basic computer skills	Prior Frogger experience	Advanced programming concepts (loops, variables, conditionals)
Structure and Scaffolding	Guided, step-by-step (8 handouts)	Moderately guided (5 handouts)	Extended, complex scaffolding (10 handouts)
Mathematics Integration	Probability, directional logic	Simple averaging, positioning	Variables, inequalities, iterative modeling
Troubleshooting	Peer discussion, common misconceptions	Common errors (rule order, resets)	Advanced debugging (logic, syntax, efficiency)
Pedagogical Approach	Guided discovery, collaborative	Independent practice with support	Iterative debugging, collaboration, and conceptual depth
Creative Application	Customize game design	Modify maze and gameplay	Design complex agent-based modeling behaviors and simulations

eDGE Course 2

Course 2 built upon foundational skills developed in Course 1 and emphasized more advanced simulation design and deeper alignment to CS principles. Teachers began by creating simulations and revisiting guided discovery as an instructional strategy, then explored concepts such as abstraction, data gathering, and the use of local and global variables. The course positioned simulations as tools for analyzing data, testing claims, and exploring real-world phenomena. The final assignment was a lesson plan that incorporated implementation supports to help students create simulations and engage in computational modeling and data reasoning.

Participants

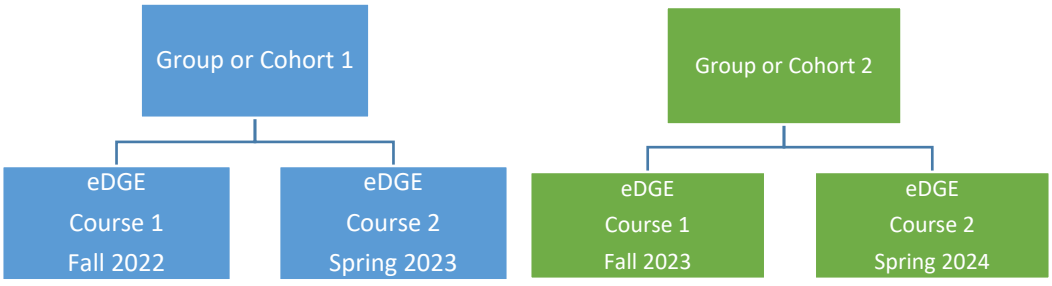
Recruitment Procedures

Recruitment occurred through university email and affiliated communication channels. The email described the purpose of the dissertation study, participation expectations, confidentiality safeguards, and the researcher's contact information. Participants who completed study activities were offered a $30 Amazon gift card.

The target population included licensed educators teaching fifth grade in high-poverty school districts who participated in the eDGE professional development initiative. The grant supported two cohorts of teachers. Group 1 consisted of n = 20 teachers who participated during the 2022–2023 school year, while Group 2 consisted of n = 26 teachers who participated during the 2023–2024 school year (Figure 1). Within the eDGE model, teachers engaged in online coursework supported by mentors and coaches while beginning classroom implementation with students. The professional learning design positioned teachers not only as learners of coding and CT concepts, but also as designers of classroom experiences that connect CT practices to mathematics learning goals.

Figure 1

Participant Groups and When They Engaged in the eDGE Two Professional Development Courses.



Participants were recruited from both cohorts (N = 46) to participate in the research study. Demographic information was drawn from two sources: survey responses and self-reported teaching assignments from discussion board posts. Because the survey was anonymous, it was not possible to directly link survey responses to discussion board data. In Group 1, nine participants reported teaching experience, while in Group 2, 24 participants reported teaching experience (Table 3). The lower number of Group 1 respondents likely reflects the fact that this cohort completed the

professional development approximately one year before follow-up data collection began. Across both groups, most participants reported limited prior coding experience with the trend more pronounced in Group 2, where nearly all respondents indicated minimal or no coding background.

Table 3

Teaching Experience by Participant Group.

Question	Group	N	0-4 years	5-9 years	10 to 14 years	More than 15 years
Years experience teaching	1	9	2	4	1	2
	2	24	9	3	4	8
Years experience teaching current grade	1	9	2	4	1	2
	2	24*	10	7	2	4
Years experience coding	1	9	6	1	2	
	2	24*	22	1		

*Note *One teacher chose not to respond to the question*

Teaching assignments (Table 4) varied widely and included self-contained elementary classrooms, librarians/technology specialists, mathematics, science, STEM, special education, and combined subject roles. Group 1 participants were more likely to serve in self-contained classroom roles, whereas Group 2 included a higher proportion of specialized positions such as librarians and technology instructors.

Table 4

Group 1 and Group 2 Teaching Duties from Discussion Board Posts

Subjects Taught	Group 1 (N = 15)	Group 2 (N = 18)
All subjects	9	6
Librarian/Technology	1	4
Math	1	2
Technology/Computer Literacy/STEM	1	1
Science and Social Studies	1	3
Special Education Math/Reading	1	
Math/Science		1
ELA/Science	1	
Math/Social Studies		1

Across both cohorts, regular computing instruction prior to eDGE participation was limited. Only two teachers in Group 2 reported daily computing instruction. In Group 1, five teachers reported rarely teaching computing activities, and only four reported teaching computing weekly or more frequently. In Group 2, most teachers reported rare or monthly instruction, with a smaller subset integrating computing weekly or daily. Taken together, these findings

suggest that both cohorts entered eDGE professional development with limited experience in CT and coding. These baseline conditions contextualize the findings, as most participants entered the program with limited formal experience in coding or CT instruction.

Research Questions

Data collection and analysis were designed to address the following questions:

1. How did eDGE participant teachers plan for and integrate computational thinking into instruction following professional development?
2. What instructional strategies did eDGE participants use when implementing computational thinking in their classroom?
3. How do eDGE teachers' self-efficacy and computational thinking proficiency change during and following professional development?
4. How do quantitative changes in efficacy and proficiency relate to eDGE teachers' instructional practices and experiences?

Data Sources, Collection Tools, and Analysis

Qualitative data sources included professional development artifacts (discussion posts, coded projects, lesson plans) and semi-structured interviews. Quantitative data sources included CTPA graphs generated by AgentSheets as educators completed tutorials, created simulations, and information collected through the Teacher Beliefs about Coding and Computational Thinking (TBACCT) instrument (Rich et al., 2021).

TBACCT (Rich et al., 2021) is a 33-item instrument designed to measure elementary school teachers' beliefs regarding coding and CT. The survey used a six-point Likert Scale with four subscales: coding self-efficacy, CT self-efficacy, teaching efficacy, and value beliefs. The instrument has a Cronbach alpha of 0.984 (Mumcu et al., 2025).

Thematic Analysis

Qualitative data from documents (discussion posts, course materials, lesson plans) and interviews were analyzed using Braun and Clarke's (2022) six-phase thematic analysis. Initial deductive codes were drawn from the Digital Promise (2022) CT framework, organized into three categories: skills, processes, and pedagogies. A subset of coding decisions was reviewed with the dissertation advisor to support consistency in code application. A codebook defined each code with examples, and multiple overlapping codes were allowed per response (Table 5).

Table 5

Example Codebook Entry for the Code, Creative.

Code	Definition	Examples	Categories	Theme
Creative	Original Ideas	I agree that creativity in the classroom does allow students to work in a different manner. Being a facilitator is new, but I do think students appreciate the change. There are all sorts of variations they could add to their game/levels to make it their own. One of the greatest benefits in this design and programming is that the students are building upon their own critical thinking, problem solving, and using their own imagination on how to create a world divergent from the tutorial.	Computational Thinking	Computational Thinking

After initial coding, additional codes not captured in the original framework were grouped into new categories, such as *enthusiasm* with engagement. Themes were then developed by examining relationships among codes across data sources using visual mapping and a qualitative data analysis tool. While co-occurrence informed pattern identification (Table 6), themes were refined through iterative review of the full dataset (Creswell & Creswell, 2018) to ensure they reflect meaningful distinctions.

Table 6

Codes Related to CT Across Documents and Interviews.

Code System	Problem-Solve	STEM	Inquiry	Critical Thinking	Engagement	Decomposition	Persevere	Algorithmic Thinking
CT	57	35	33	21	15	14	13	6

Initial themes were then reviewed to ensure sufficient supporting data and coherent patterns (Braun & Clarke, 2022; Nowell et al., 2017). A table linking codes, definitions, categories, and themes supported this process. Themes were refined, defined, and aligned to the research questions. Findings were reported using concise narrative with illustrative quotes to support credibility and demonstrate how themes emerged from the data.

Quantitative Analysis

Quantitative data was analyzed using descriptive statistics and effect size estimates. CTPA data were analyzed by calculating the mean and standard deviation for each CT pattern by cohort. For the Frogger tutorial, mean occurrences were compared to expected values established by AgentSheets (Ioannidou et al., 2011). For other tutorials, where

benchmarks were not available, comparisons focused on differences in mean and variability between cohorts. TBACCT data (Rich et al., 2021) were analyzed by calculating mean scores for each subscale.

Mixed Methods Analysis

Qualitative and quantitative data were integrated through a qualitative-dominant concurrent framework, with primary qualitative findings presented first and subsequently contextually enriched by secondary quantitative results using a side-by-side presentation strategy (Creswell & Creswell, 2018; Creswell & Plano Clark, 2018). By embedding a secondary quantitative strand within a prioritized qualitative design, this approach supports data complementarity and expansion, preserving the rich, interpretive depth of teacher experiences while anchoring those narratives with targeted statistical measures (Creswell & Plano Clark, 2018). Integrating these datasets using a joint display allowed patterns of convergence, divergence, and complementarity across data sources to be systematically examined to address the core research questions.

A logistical constraint of the study was the unequal sample sizes across data sources because quantitative data collection began after the professional development had already commenced. However, a qualitative-dominant mixed methods design is uniquely resilient to unequal sample sizes or incomplete quantitative data strands, as the secondary quantitative metrics serve to illustrate, expand upon, or test the primary qualitative themes rather than match them in statistical weight (Fetters et al., 2013; Younas et al., 2021). By integrating these heterogeneous data sources, the final meta-inferences are strengthened through multi-faceted evidence without undermining the primary qualitative priority of the study.

The researcher acknowledges that prior experience in STEM education and teacher professional development may have influenced the interpretation of both qualitative and quantitative findings. Reflexive practices were used throughout data analysis to examine how prior assumptions and experiences may have shaped coding decisions and integration of results.

Procedure

In the dissertation study, the focus narrowed to teacher planning, implementation strategies, and patterns of learning trajectories and areas of challenge in CT and coding concepts.

Dissertation Data Collection Steps

Data collection began with document analysis of archived course content, organized into three categories: (1) discussion prompt responses, (2) course structure and learning materials, and (3) coded project artifacts created by teachers. While document analysis progressed, the researcher contacted teachers who completed the TBACCT and indicated they would agree to an interview to schedule interviews. Interviews were conducted via an online meeting platform. During interviews, teachers were asked to share any relevant lesson materials, including previously used lesson plans or plans for near future implementation.

Results

Qualitiative Data

Discussion Board Findings

Group 1 discussion responses yielded 625 coded instances, while Group 2 responses yielded 1,128 (Table 7). According to Monroe et al. (2019), these patterns suggest teachers conceptualized CT as both technical skill and as a pedagogical approach associated with motivation, persistence, and student-centered learning. Across both groups, CT was described as a structured approach to problem solving, emphasizing decomposition, cause-and-effect reasoning, and stepwise logic. Teachers also consistently linked CT activities to increased student motivation and participation.

Table 7

Most Common Codes for Each Group

Group 1	Group 2
Problem Solving (78) 12.48%	Enthusiasm (177) 15.69%
Computational Thinking (47) 7.52%	Perseverance (103) 9.13%
Collaboration (43) 6.88%	Problem Solving (99) 8.7%
Debugging (40) 6.4%	Engagement (95) 8.4%
Engagement (38) 6.08%	Computational Thinking (80) 7.09%
Enthusiasm (37) 5.92%	Creativity (61) 5.41%

Interview Findings

Eight teachers participated in semi-structured interviews. Dominant codes included problem solving (27), integration across subjects (27), algorithmic thinking (22), decomposition (15), computational thinking (13), student choice and interest (13), collaboration (12), abstraction (7), and debugging (7).

Interview narratives provided concrete classroom examples that extended discussion board themes. Teachers described using CT to structure math problem solving, science simulations, robotics activities, and interdisciplinary projects. Many reported shifts toward facilitation, emphasizing peer support, experimentation, and iterative improvement.

Lesson Plan Artifact Findings

Eight lesson plans were analyzed using deductive coding. Examples of deductive coding included simulation-based learning using agent interactions or ecosystems, emphasis on cause-and-effect relationships, and inclusion of research, prediction, testing, and iterative investigation.

All eight lessons involved student-designed simulations. Six focused on ecological systems (e.g., predator–prey relationships), one modeled school-based social systems, and one simulated laboratory conditions. Students designed agents with programmed behaviors, tested outcomes, adjusted variables, and refined rules. Five lessons included formal research components in which students gathered background information before constructing models. The artifacts demonstrate alignment between PD content and classroom enactment. Teachers operationalized CT through modeling, experimentation, and data analysis rather than isolated coding drills.

Themes Across Qualitative Data

Six major themes were identified across discussion boards, lesson plan artifacts, and interviews for both cohorts.

Theme 1: Computational Thinking as Logical and Algorithmic Problem Solving.

Across discussion posts and interviews, teachers conceptualized CT as a structured approach to problem-solving grounded in logic and algorithmic reasoning. Participants described CT as the decomposition of complex tasks, the identification of patterns, and the application of stepwise procedures to arrive at solutions.

Teachers connected these practices to existing routines in mathematics. For example, one teacher described CT integration as requiring students to engage in "a multi-step math word problem that required students to break down the steps, identify what was working and what needed to be done differently," highlighting decomposition and iterative reasoning in student learning. Similarly, teachers referenced incorporating "algorithms, decomposition, and generalization" into instruction as strategies, demonstrating an emerging alignment between CT practices and disciplinary problem-solving approaches.

The emphasis on structured reasoning links to broader educational goals. One teacher noted, "computational thinking relates to STEM by challenging students to use multiple strategies to understand concepts and develop deeper problem-solving skills." These patterns are consistent with Bandura's (1994) sources of self-efficacy, particularly as teachers described developing confidence through hands-on problem solving (mastery experiences) and peer interaction (vicarious experiences) (Täschner et al., 2025; Tschannen-Moran & McMaster, 2009).

Theme 2: Creativity Through Design and Simulation.

Teachers emphasized creativity as a central affordance of CT through game design and simulation-based learning. One participant noted that students could "add to their game/levels to make it their own," underscoring how design environments supported creative expression and ownership of learning. Teachers observed that CT activities "fostered creativity, critical thinking, and engagement," suggesting that design-based approaches supported both CT development and student engagement. Students translated concepts into rule-based systems and examined resulting changes.

Teachers also described how simulation–based learning facilitated integration across disciplines. For example, a teacher described a project in which students "designed and built models of different energy sources," integrating CT

with science content through creative modeling and simulation. Students modified and extended projects, as one teacher noted that learners "begun to change things up and make their frogger worlds more unique to what they want. It encourages them to be creative and create their own "if/then" situations." These examples illustrate how design environments support conceptual understanding and computational expression. The open-ended nature of design tasks provided opportunities for repeated mastery experiences, allowing teachers and students to build confidence through iterative success (Täschner et al., 2025).

Theme 3: Inquiry-Based, Student-Centered Learning.

Teachers described moving away from direct instruction toward facilitation, emphasizing exploration, questioning, and collaborative problem solving. Participants highlighted the use of "open-ended questions that encourage deeper thinking and multiple paths to a solution." Teachers described restructuring classroom roles to emphasize that instruction increasingly involved "peer support, experimentation, and iterative improvement rather than as direct providers of answers".

This shift was implemented through collaborative structures, such as group work and peer tutoring, where "students work in groups to apply what they've learned… using peer tutors to assist those who are struggling". A teacher commented on the discussion boards, "Guided inquiry allows students to receive help but also work through their problems on their own after receiving help once. It also gives them the opportunity to help their classmates solve their problems as well." Collaborative structures such as peer support and group problem solving also reflect vicarious experiences (Bandura, 1994; Täschner et al., 2025), where observing others' success contributes to developing efficacy beliefs.

Theme 4: Engagement and Persistence Through CT Practices.

Teachers reported that CT activities promoted engagement and persistence. Students were described as "very willing to experiment and try different approaches, even if they made mistakes," supporting risk-taking and productive struggles.

Teachers also reported increased confidence. One participant reflected that, although initially uncertain while developing a Frogger simulation, the process of completing the task and supporting students led to increased confidence and a greater willingness to troubleshoot and refine designs. This experience was further reinforced during subsequent tasks, which were described as "much smoother and far less frustrating," indicating growth through practice. Teachers also observed students collaborating and "teaching each about their own problem solving and critical thinking skills through programming," indicating that engagement was both individual and collective.

CT practices foster environments in which both students and teachers engage in sustained effort through cycles of experimentation, feedback, and revision. From a self-efficacy perspective, this reflects the role of mastery experiences and the regulation of frustration in building confidence during challenging tasks (Bandura, 1994; Täschner et al., 2025; Tschannen-Moran & McMaster, 2009).

Theme 5: Teacher Enthusiasm and Efficacy as Drivers of Student Motivation.

Teachers described growth in confidence. Initial uncertainty gave way to confidence through practice. One participant noted that after completing initial tasks, subsequent work became "much smoother and far less frustrating," reflecting growing competence through practice.

This shift in teacher efficacy was closely connected to classroom practice. As teachers became more confident, they reported greater willingness to support students in navigating challenges and to facilitate collaborative learning environments. Students were observed "teaching each other about their own problem solving and critical thinking skills through programming," suggesting that teacher confidence contributed to more distributed and student-centered forms of learning.

These findings indicate that iterative engagement supported teacher self-efficacy, which, in turn, influenced instructional decision-making and classroom dynamics. From a self-efficacy perspective, these patterns reflect the role of mastery experiences and the regulation of challenge in shaping teachers' beliefs in their instructional capabilities (Bandura, 1994; Tschannen-Moran & McMaster, 2009).

Theme 6: Iteration and Debugging as Core Learning Processes.

Learning in CT was characterized by cycles of testing, debugging, and refinement. Teachers emphasized that students were expected to evaluate their solutions, "identify what was working and what needed to be done differently," and revise their approaches accordingly. Teachers described practices as central to developing problem-solving capacity, noting that students' mistakes and subsequent revisions "sharpen their abilities" and support higher-level thinking. In this way, debugging was not treated as error correction alone but as a critical mechanism for learning, requiring students to engage in reflection, analysis, and continuous improvement.

Iterative refinement is foundational to CT learning, supporting both conceptual understanding and persistence. From a self-efficacy perspective, debugging and revision represent key moments of emotional regulation, as students and teachers navigate frustration and uncertainty while building confidence through successful problem resolution (Bandura, 1994; Täschner et al., 2025; Tschannen-Moran & McMaster, 2009).

Quantitative Findings

Computational Thinking Pattern Analysis (CTPA) findings

CTPA outputs indicate how participants used computational thinking patterns as well as conditions and actions within AgentCubes. The CT patterns include cursor control, generation, absorption, collision, transportation, push, pull, diffusion, and hill climbing. Programming in AgentCubes is structured around visual IF/THEN rules, in which users stack conditions ("IF" blocks: see, key, percent chance, and stacked) and actions ("THEN" blocks: move, new, play sound, change, wait, show message, reload world, and transport). CTPA results are reported in terms of alignment with expected CT patterns and the frequency of conditions and actions falling outside expected ranges based on tutorial completion patterns (Ioannidou et al., 2011).

Frogger: Computational Thinking Patterns. Participants demonstrated strong alignment across foundational CT patterns (cursor control, generation, absorption, collision, transportation, push, and pull). Mean values aligned with AgentSheets benchmarks, with low variability across participants.

In contrast, hill climbing emerged as a persistent area of difficulty. All Group 1 participants and all 13 Group 2 participants fell outside expected range for this pattern (Table 8). Hill climbing, which involves goal-directed or gradient-based movement, represents a more advanced computational concept requiring conditional logic and dynamic decision-making. These findings suggest that teachers developed foundational CT patterns, while more complex, adaptive behaviors such as hill climbing remained underdeveloped. This pattern may indicate that mastery experiences were limited, potentially constraining the development of self-efficacy for more complex computational concepts.

Table 8

Summary of Most Frequent Outside-Expected CT Patterns, Conditions, and Actions in Frogger and PacMan

Domain	Frogger Group 1 (N=2)	Frogger Group 2 (N=13)	PacMan Group 1 (N=2)	PacMan Group 2 (N=11)
Most Frequent CT Pattern	Hill Climbing (2)	Hill Climbing (13)	Diffusion (2)	Diffusion (5), Hill Climbing (3), and Collision (3)
Most Frequent Conditions (IF)	Key (2), and Percent Chance (2)	Key (6), See (5), and Stacked (3)	None, all were within expected range	See (1)
Most Frequent Actions (THEN)	None, all were within expected range	New (3), and Play Sound (3)	None, all were within expected range	Move (1) and Show Message (1)

Frogger: Conditions and Actions. The Frogger results (Table 8) suggest meaningful differences in how the two groups used conditions and actions. Group 1 more closely approximated the Frogger standard across most listed conditions and actions, with outside-expected use observed only for the Key and Percent Chance condition blocks. By contrast, Group 2 showed outside-expected use across all listed conditions and actions, with the greatest divergence appearing in the condition blocks Key and See, followed by more moderate variation in Stacked, New, and Play Sound. Because condition blocks such as See, Key, and Percent Chance help organize event-based logic and decision structures, closer alignment with the expected Frogger pattern may indicate greater consistency in applying these elements. At the same time, Group 2's broader distribution of outside-expected use across both conditions and actions may reflect a less uniform application of these patterns, although it may also indicate greater experimentation with available coding options.

From a CT efficacy perspective, these findings may suggest that learners who more consistently approximate expected uses of conditions and actions are better positioned to experience successful task performance, which could support stronger efficacy beliefs. However, given the small size of Group 1 and the descriptive nature of these comparisons, these interpretations should be viewed cautiously. Overall, the pattern of results points to the potential importance of condition structures in supporting both accurate task completion and emerging confidence in computational thinking.

Pac-Man: Computational Thinking Patterns. Because predefined performance benchmarks were not available for PacMan, results were interpreted using means and standard deviations and counting patterns outside expected ranges. Participants demonstrated consistent use of cursor control, push, pull, and collision, indicating transfer of foundational skills from the Frogger task. However, diffusion and hill climbing again showed the greatest variability, particularly among Group 2 participants (Table 8). These findings reinforce Frogger results, indicating that advanced CT patterns involving probabilistic behavior (diffusion) and goal-directed movement (hill climbing) remain challenging.

Pac-Man: Conditions and Actions. The PacMan results suggest more consistent use of conditions and actions across both groups (Table 8). Group 1 closely matched the expected PacMan structure across all listed conditions and actions, with no outside-expected use observed. Group 2 also showed relatively consistent performance, with outside-expected use limited to the See condition and the Move, Play Sound, and Show Message actions, each occurring for one participant. From a CT efficacy perspective, the greater consistency shown in PacMan may suggest that participants were better able to apply familiar programming structures in this task, which could support stronger perceptions of capability. However, given the small Group 1 sample and the descriptive nature of these comparisons, these interpretations should be viewed cautiously.

Teacher Beliefs About Coding and Computational Thinking Survey Results

Effect sizes (Cohen's d) were calculated to estimate the magnitude of changes in teacher efficacy from mid- to post-intervention. Due to unmatched data, effect sizes were estimated using pooled standard deviations and should be interpreted as approximate indicators of change (Lakens, 2013).

The change represents a moderate size effect ($d \approx 0.62$), suggesting that professional development had a meaningful impact on teachers' confidence in their coding abilities (Table 9). However, confidence for the year-post measurement was lower ($M = 4.0$), accompanied by higher variability ($SD = 0.93$), indicating that teachers' efficacy experiences diverged over time.

The change corresponds to a small effect size ($d \approx 0.21$), indicating a modest improvement in teachers' instructional confidence (Table 10). At year-post, the mean was 4.2, with a higher variability ($SD = 1.39$). This suggests that without sustained support, teachers' confidence in teaching coding may diminish, and differences in implementation experiences become more pronounced.

Table 9

Efficacy in Coding Skills

Group	Survey Timing	Mean	Standard Deviation
2	Mid (n= 21)	4.4	0.48
	Post (n=19)	4.7	0.49
1	Year-Post (n= 10)	4.0	0.93

Table 10

Efficacy in Teaching Coding Skills

Group	Survey Timing	Mean	Standard Deviation
2	Mid (n= 21)	4.7	0.40
	Post (n=19)	4.8	0.53
1	Year-Post (n= 10)	4.2	1.39

For the value of computing education (Table 11), the change reflects a moderate negative effect size ($d \approx -0.56$), although overall ratings remained high, indicating that teachers continued to value computing education despite slight shifts in perception. At the year-post measurement for Group 1, the mean remained relatively high ($M = 4.7$), although variability was slightly greater than in the earlier Group 2 time points. This indicates that teachers continued to recognize the importance of computing education, even as their confidence in implementation fluctuated.

Table 11

Value of Computing Education

Group	Survey Timing	Mean	Standard Deviation
2	Mid (n= 21)	5.1	0.37
	Post (n=19)	4.9	0.34
1	Year-Post (n= 10)	4.7	0.46

The difference corresponds to a negligible effect size ($d \approx 0.11$), suggesting minimal change in teachers' perceived CT efficacy during this period (Table 12). Again, group 1 results were lower in the year-post ($M = 4.4$) with increased variability ($SD = 1.15$). This pattern suggests that while teachers initially developed confidence in CT concepts, sustaining that confidence over time may require continued practice and support.

Table 12

Efficacy in CT

Group	Survey Timing	Mean	Standard Deviation
2	Mid (n= 21)	4.7	0.93
	Post (n=19)	4.8	0.84
1	Year-Post (n= 10)	4.4	1.15

Effect size estimates indicate the strongest impact on coding efficacy, with more limited effects on instructional efficacy and CT efficacy. These patterns suggest that while teachers develop confidence in their own coding abilities, translating that confidence into instructional practice and broader CT understanding may require additional support. The decline in efficacy at the year-post time point may reflect a reduction in sustained mastery experiences and support over time.

Mixed-Methods Findings

The relationship between efficacy and proficiency was not uniform. Increases in coding self-efficacy aligned with demonstrated proficiency in foundational patterns, whereas limited gains in CT efficacy corresponded with continued difficulty in advanced computational concepts. A joint display (Table 13) was developed to examine how teachers' reported experiences aligned with changes in self-efficacy.

Integrated findings indicate that the relationship between efficacy and computational proficiency is contingent on the level of conceptual complexity. Convergent evidence across data sources shows that teachers' self-efficacy and performance co-develop foundational, deterministic aspects of CT. However, divergence at higher levels of complexity, where abstraction, probabilistic reasoning, and dynamic behaviors are required, suggests a breakdown in this relationship. Interpreted through Bandura's (1994) framework, this pattern reflects an imbalance in sources of efficacy: while mastery experiences supported confidence in basic coding practices, more complex tasks likely lacked sufficient scaffolding, limiting opportunities for sustained efficacy development. This yields a meta-inference of partial conceptualization, wherein cognitive problem-solving competence develops without parallel growth in computational execution, constraining the translation of understanding into practice.

Table 13

Joint Display of CT Integration, Proficiency, and Teacher Efficacy

Construct / Theme	Qualitative Findings (What teachers said/did)	CTPA Evidence (What teachers demonstrated)	Survey Evidence (Efficacy Change)	Integrated Interpretation (What it means)
CT as Structured Problem Solving	Teachers described CT as breaking down problems, using step-by-step reasoning, and guiding students through solutions	Strong performance on foundational patterns (cursor control, collision, generation, absorption)	Moderate increase in coding efficacy ($d \approx 0.62$)	Convergence: Teachers developed confidence and proficiency in foundational, structured aspects of CT aligned with problem-solving practices
Design & Simulation as Learning Contexts	Teachers used simulations and game design to model systems (ecosystems, math relationships)	Consistent use of deterministic CT patterns; successful implementation of basic agent behaviors	Small increase in teaching efficacy ($d \approx 0.21$)	Partial convergence: Teachers enacted CT through design, but instructional confidence grew more slowly than personal skill
Inquiry & Collaborative Learning	Classrooms shifted toward student-centered learning, peer collaboration, and facilitation roles	Not directly measured in CTPA but reflected in iterative project development	Not directly measured	Expansion: Qualitative findings extend beyond quantitative measures, highlighting pedagogical shifts not captured by CTPA or surveys
Engagement & Productive Struggle	Teachers reported increased student persistence, experimentation, and willingness to debug	Evidence of repeated attempts and refinement in coding patterns	Not directly measured	Complementarity: Engagement supports iterative learning processes required for CT development
Iteration & Debugging as Core Processes	Teachers emphasized testing, revising, and troubleshooting as key learning experiences	Variability in advanced pattern use; inconsistent application of complex logic	Minimal change in CT efficacy ($d \approx 0.11$)	Divergence: Teachers value iteration conceptually, but struggle to implement complex computational logic consistently
Advanced CT Challenges (Abstraction & Complexity)	Teachers reported difficulty with abstraction, probabilistic reasoning, and debugging complex systems	Weak performance in hill climbing, diffusion, and probabilistic actions	No meaningful gain in CT efficacy	Strong convergence: Both qualitative and quantitative data indicate persistent difficulty with advanced CT concepts
Partial Conceptualization of CT	Teachers defined CT as problem solving but rarely referenced computational execution	Difficulty translating concepts into complex coded behaviors	Modest gains in efficacy, but not in deeper CT understanding	Meta-inference: Evidence may suggest a decoupling of cognitive and computational dimensions of CT, where problem-solving is developed more than computational implementation

Discussions

Research Question 1: How Did eDGE Participant Teachers Plan for and Integrate Computational Thinking into Instruction Following Professional Development?

Teachers primarily integrated CT through interdisciplinary, student-centered instructional approaches rather than as a standalone content area. Across qualitative data sources, teachers embedded CT within mathematics and science contexts, most often through simulation-based learning and problem-based tasks. They designed collaborative learning environments where students shared strategies and engaged in debugging. These approaches engaged students in activities that required them to model systems, test outcomes, and iteratively refine solutions.

Teachers predominantly defined CT in terms of problem-solving processes while largely omitting the role of technology in executing solutions. This reflects a partial conceptualization of CT, in which cognitive problem-solving is foregrounded while computational execution is underemphasized.

Research Question 2: What Instructional Strategies Did eDGE Participants Use When Implementing Computational Thinking in Their Classroom?

Teachers framed instruction around authentic problems or scenarios, positioning students as designers who developed and tested solutions. A central strategy was the use of simulation and game-based modeling, in which students created and modified digital agents to represent systems. Through this process, students translated conceptual understanding into rule-based behaviors, tested system outcomes, and revised their models.

Instruction emphasized open-ended questioning, peer support, and iterative problem solving, as students collaborated to debug code, test alternative approaches, and refine solutions. Teachers described shifting their role from direct instruction to facilitation. These strategies reflect an instructional shift toward student-centered, design-oriented learning environments that prioritize exploration, iteration, and collaboration. These instructional strategies align with modeling and simulation practices described in CT frameworks (Weintrop et al., 2016).

Research Question 3: How Do eDGE Teachers' Self-Efficacy and Computational Thinking Proficiency Change Following Professional Development?

Teachers' self-efficacy and CT proficiency improved following professional development, although gains were uneven. Survey results showed the largest increase in coding self-efficacy from mid- to post-intervention ($d \approx 0.62$), indicating a meaningful gain in teachers' coding confidence. In contrast, teaching efficacy increased only slightly ($d \approx 0.21$), and CT efficacy showed minimal change ($d \approx 0.11$), suggesting more limited growth in applying CT concepts and instructional practices.

CTPA results showed a parallel pattern. Teachers demonstrated strong proficiency in foundational computational thinking patterns, including cursor control, collision, generation, and absorption, which involve direct, deterministic interactions. However, more advanced patterns such as hill climbing and diffusion remained challenging, with many

participants falling below expected levels. These findings indicate that teachers developed proficiency and confidence in structured, deterministic aspects of CT, but had trouble with more complex, adaptive computational processes which support the meta-inference of partial conceptualization of CT.

Research Question 4: How Do Quantitative Changes in Efficacy and Proficiency Relate to eDGE Teachers' Instructional Practices and Experiences?

Integrated findings reveal patterns of convergence, divergence, and complementarity across data sources.

Convergent findings were most evident in teachers' development of foundational CT skills. Increases in coding self-efficacy ($d \approx 0.62$) aligned with both qualitative descriptions of structured problem-solving and CTPA evidence demonstrating strong performance on foundational patterns such as cursor control, collision, and generation. These findings suggest that structured, scaffolded learning experiences support both confidence and proficiency in computational tasks.

Partial convergence was observed in instructional practices related to design and simulation. Smaller gains in teaching efficacy ($d \approx 0.21$) indicate that while teachers adopted these practices, their confidence in implementing them instructionally developed more gradually. This suggests a lag between acquiring personal skills and translating those skills into classroom practice.

Divergence across data sources emerged most clearly in relation to advanced CT concepts. While teachers emphasized the importance of iteration and debugging in qualitative accounts, CTPA results indicated continued difficulty with complex patterns such as hill climbing and diffusion, and CT efficacy showed minimal change ($d \approx 0.11$). This disconnect suggests that teachers recognized the value of these practices conceptually but struggled to implement them effectively in computational contexts.

Complementary findings highlight instructional shifts not fully captured by quantitative measures. Teachers described increased use of inquiry-based, collaborative learning environments that supported engagement and persistence. While these practices were not directly reflected in survey or CTPA measures, they provide important context for understanding how teachers enacted CT in the classroom.

Together, the findings indicate a partial conceptualization of CT, in which practices and efficacy gains aligned more with cognitive problem solving than computational execution. While professional development supported initial growth in both efficacy and proficiency, deeper integration of CT requires sustained opportunities to connect problem-solving strategies with computational implementation.

Conclusion and Limitations

Findings indicate that teachers embedded CT within interdisciplinary, student-centered instruction, using simulation-based, problem-based, and iterative design approaches. Teachers demonstrated growth in foundational CT practices (e.g., decomposition, algorithmic reasoning), supported by qualitative data, CTPA, and gains in coding self-efficacy. However, more advanced concepts, including abstraction and complex system modeling, remained challenging, particularly in translating conceptual understanding into computational implementation. Findings indicate a partial conceptualization of CT, in which cognitive problem-solving is emphasized over computational execution.

Limitations

Several limitations should be considered. First, the sample size was modest and drawn primarily from rural, high-poverty school contexts, which may limit the generalizability of the results. Participants were also self-selected into the professional development program, which may introduce bias toward teachers who were more motivated or open to innovation.

Second, data sources included self-reported perceptions, discussion board posts, and selected artifacts, which may not fully capture enacted classroom practice. Direct classroom observations were not included. As a result, conclusions about instructional implementation should be interpreted with caution.

Third, the quantitative component of the study relied on descriptive statistics and effect size estimates because matched-pair data were not available. This limits the ability to make strong inferential claims about the significance of observed differences.

Finally, the study did not include direct measures of student learning outcomes. The extent to which these instructional changes influenced students' achievement in mathematics, science, or CT remains unclear. Despite these limitations, triangulation across multiple data sources provides a comprehensive understanding of CT implementation.

Implications and Future Research

Findings from this study suggest several important implications for the design of professional development (PD), classroom implementation of CT, and future research.

Implications for Professional Development

Professional development should explicitly address the relationship between cognitive problem-solving processes and computational execution. While teachers readily adopted CT as a structured approach to problem solving, they did not consistently connect these processes to how solutions are expressed and executed computationally. PD should differentiate between learning to code and learning CT.

Professional development should differentiate between learning to code and learning CT. Although coding activities provided valuable mastery experiences that supported gains in self-efficacy, those experiences alone were insufficient for developing deeper CT understanding. Effective PD should therefore include opportunities for teachers to engage in multiple forms of CT, including unplugged modeling, algorithm design, and structured reflection on how computational processes operate across contexts.

In addition, findings highlight the importance of sustained and supported implementation opportunities. While teachers demonstrated gains in coding efficacy, growth in teaching efficacy and CT remained more limited. From a self-efficacy perspective (Bandura, 1994), this suggests that teachers require continued opportunities to enact instruction, receive feedback, and refine their practice. Ongoing coaching, collaborative planning, and classroom-based implementation support are critical for translating initial learning into sustained instructional change.

Implications for Classroom Practice

At the classroom level, findings support the use of simulation-based learning, problem-based instruction, and iterative design processes as effective entry points for integrating CT. These approaches align well with existing disciplinary practices in mathematics and science and provide meaningful contexts for student engagement.

However, teachers may require additional support in facilitating more complex CT practices, particularly those involving abstraction, probabilistic reasoning, and dynamic system modeling. Instructional routines should explicitly connect students' problem-solving processes to the underlying computational mechanisms, helping learners understand not only what solutions are, but how those solutions are executed within a computational system.

The emphasis on iteration, debugging, and productive struggle observed in this study also suggests that classroom environments should normalize error as part of the learning process. Structured opportunities for revision and reflection can support both conceptual understanding and persistence, particularly when paired with collaborative learning structures.

Implications for Research

First, the identification of a partial conceptualization of CT highlights the need for further investigation into how teachers develop and operationalize CT definitions. Future studies should examine whether explicit attention to computational execution improves both teacher understanding and instructional practice.

Second, findings suggest that coding-based PD alone may not be sufficient to fully develop CT. Research should explore how different instructional approaches—such as unplugged activities, modeling tasks, and interdisciplinary integration—contribute to more comprehensive CT development.

Third, longitudinal research is needed to examine how teacher efficacy and CT proficiency evolve over time, particularly in relation to sustained classroom implementation. The decline and variability observed in year-post measures suggest that initial gains may not persist without continued support.

Finally, future research should incorporate direct measures of student learning outcomes to better understand how CT integration impacts achievement in mathematics, science, and CT. Developing valid and reliable assessment tools for CT remains a critical need for the field.

Acknowledgements

This work was supported by the U.S. Department of Education, Education Innovation and Research (EIR) Program (Grant No. S411C200078). The opinions expressed are those of the author and do not necessarily reflect the views of the U.S. Department of Education.

References

Bandura, A. (1977). Self-efficacy: Toward a unifying theory of behavioral change. *Psychological Review*, 84(2), 191–215. https://doi.org/10.1037/0033-295X.84.2.191.

Bandura, A. (1994). Self-efficacy. In V. S. Ramachandran (Ed.), *Encyclopedia of human behavior* (Vol. 4, pp. 71–81). Academic Press.

Bandura, A. (1994). Social cognitive theory of mass communication. In J. Bryant & D. Zillmann (Eds.), *Media effects: Advances in theory and research* (pp. 61–90). Lawrence Erlbaum Associates.

Berland, M., & Wilensky, U. (2015). Comparing virtual and physical robotics environments for supporting complex systems and computational thinking. *Journal of Science Education and Technology*, 24(5), 628–647. https://doi.org/10.1007/s10956-015-9552-x.

Braun, V., & Clarke, V. (2022). *Thematic analysis: A practical guide*. SAGE.

Brennan, K., & Resnick, M. (2012, April 13–17). New frameworks for studying and assessing the development of computational thinking [Paper presentation]. *American Educational Research Association Annual Meeting*, Vancouver, Canada. http://scratched.gse.harvard.edu/ct/files/AERA2012.pdf.

Chesnut, S. R., & Burley, H. (2015). Self-efficacy as a predictor of commitment to the teaching profession: A meta-analysis. *Educational Research Review*, 15, 1–16. https://doi.org/10.1016/j.edurev.2015.02.001.

Creswell, J. W., & Creswell, J. D. (2018). *Research design: Qualitative, quantitative, and mixed methods approaches* (5th ed.). SAGE.

Creswell, J. W., & Plano Clark, V. L. (2017). *Designing and conducting mixed methods research* (3rd ed.). SAGE.

Csikszentmihalyi, M. (1990). Flow: The psychology of optimal experience. *Cambridge University Press*.

Computer Science Teachers Association, & International Society for Technology in Education. (2011). *Operational definition of computational thinking for K–12 education*. https://www.iste.org/docs/pdfs/Operational-Definition-of-Computational-Thinking.pdf.

Digital Promise. (2022, May 13). *Computational thinking*. https://digitalpromise.org/initiative/computational-thinking/.

Duncan, C., & Bell, T. (2015). A pilot computer science and programming course for primary school students. In *Proceedings of the Workshop in Primary and Secondary Computing Education* (pp. 39–48). ACM.

https://doi.org/10.1145/2818314.2818328.

Eells, R. J. (2011). *Meta-analysis of the relationship between collective teacher efficacy and student achievement* (Doctoral dissertation, Loyola University Chicago). Loyola eCommons. https://ecommons.luc.edu/luc_diss/133.

Fackler, S., & Malmberg, L.-E. (2016). Teachers' self-efficacy in 14 OECD countries: Teacher, student group, school and leadership effects. *Teaching and Teacher Education*, 56, 185–195. https://doi.org/10.1016/j.tate.2016.03.002.

Fetters, M. D., Curry, L. A., & Creswell, J. W. (2013). Achieving integration in mixed methods designs: Principles and practices. *Health Services Research*, 48(6, Pt. 2), 2134–2156. https://doi.org/10.1111/1475-6773.12117.

Gardner, H., & Davis, K. (2014). *The app generation: How today's youth navigate identity, iIntimacy, and iImagination in a digital world.* Yale University Press. http://www.jstor.org/stable/j.ctt5vm7dh.

Gootman, E. (2007, October 24). Taking the middle schoolers out of the middle. The New York Times. https://www.nytimes.com/2007/01/22/education/22middle.html.

Grover, S., & Pea, R. D. (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.

Ioannidou, A., Bennett, V., Repenning, A., Koh, K., & Basawapatna, A. (2011, April 8–12). Computational thinking patterns [Paper presentation]. *American Educational Research Association Annual Meeting*, New Orleans, LA, United States.

Ioannidou, A., Rader, C., Repenning, A., Lewis, C., & Cherry, G. (2003). Making constructionism work in the classroom. *International Journal of Computers for Mathematical Learning*, 8(1), 63–108. https://doi.org/10.1023/A:1025617704695.

Kafai, Y. B. (2006). Playing and making games for learning: Instructionist and constructionist perspectives for game studies. *Games and Culture*, 1(1), 36–40. https://doi.org/10.1177/1555412005281767.

Kalelioglu, F., Gülbahar, Y., & Kukul, V. (2016). A framework for computational thinking based on a systematic research review. Baltic *Journal of Modern Computing*, 4(3), 583–596.

Klassen, R. M., & Tze, V. M. C. (2014). Teachers' self-efficacy, personality, and teaching effectiveness: A meta-analysis. *Educational Research Review*, 12, 59–76. https://doi.org/10.1016/j.edurev.2014.06.001.

Lakens, D. (2013). Calculating and reporting effect sizes to facilitate cumulative science: A practical primer for t tests and ANOVAs. *Frontiers in Psychology*, 4, Article 863. https://doi.org/10.3389/fpsyg.2013.00863.

Lee, I., Grover, S., Martin, F., Pillai, S., & Malyn-Smith, J. (2020). Computational thinking from a disciplinary perspective: Integrating computational thinking in K–12 science, technology, engineering, and mathematics education. *Journal of Science Education and Technology*, 29(1), 1–8. https://doi.org/10.1007/s10956-019-09803-w.

Lockwood, J., & Mooney, A. (2017). *Computational thinking in education: Where does it fit? A systematic literary review*. arXiv. https://arxiv.org/abs/1703.07659.

Manovich, L. (2018). *Software takes command*. Bloomsbury Academic.

Mason, S. L., & Rich, P. J. (2019). Preparing elementary school teachers to teach computing, coding, and computational thinking. *Contemporary Issues in Technology and Teacher Education*, 19(4), 790–824. https://citejournal.org/volume-19/issue-4-19/general/preparing-elementary-school-teachers-to-teach-computing-coding-and-computational-thinking/.

Mumcu, F. K., Andic, B., Maricic, M., Tejera, M., & Lavicza, Z. (2025). Unpacking teachers' value beliefs about computational thinking and programming. *Journal of Educational Technology & Online Learning*, 8(1), 41–63. https://doi.org/10.31681/jetol.1497284.

National Science Foundation, National Center for Science and Engineering Statistics. (2017). *Women, minorities, and persons with disabilities in science and engineering: 2015* (Special Report NSF 15-311). https://www.nsf.gov/statistics/wmpd/.

Portelance, D. J., & Bers, M. U. (2015, June). Code and tell: Assessing young children's learning of computational thinking using peer video interviews with ScratchJr. In *Proceedings of the 14th International Conference on Interaction Design and Children*, ACM, 271–274. https://doi.org/10.1145/2771839.2771894.

Prensky, M. (2008, January 14*). Programming is the new literacy*. Edutopia. http://www.edutopia.org/literacy-computer-programming.

Repenning, A., & Ioannidou, A. (2008). Broadening participation through scalable game design. In *Proceedings of the 39th ACM Technical Symposium on Computer Science Education,* ACM, 305–309. https://doi.org/10.1145/1352135.135224.

Repenning, A., Webb, D., & Ioannidou, A. (2010). Scalable game design and the development of a checklist for getting computational thinking into public schools. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education*, ACM, 265–269. https://doi.org/10.1145/1734263.1734357.

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.

Rich, P. J., Mason, S. L., & O'Leary, J. (2021). Measuring the effect of continuous professional development on elementary teachers' self-efficacy to teach coding and computational thinking. *Computers & Education*, 168, Article 104196. https://doi.org/10.1016/j.compedu.2021.104196.

Rittmayer, M. A., & Beier, M. E. (2009). *Self-efficacy in STEM. In Applying research to practice (ARP) resources*. http://www.engr.psu.edu/AWE/ARPresources.aspx.

Rodrigues, R. N., Costa, C., & Martins, F. (2024). Integration of computational thinking in initial teacher training for primary schools: A systematic review. *Frontiers in Education*, 9, Article 1330065. https://doi.org/10.3389/feduc.2024.1330065.

Rushkoff, D. (2012, November 13). *Code literacy: A 21st-century requirement*. Edutopia. http://www.edutopia.org/blog/code-literacy-21st-century-requirement-douglas-rushkoff.

Selby, C., & Woollard, J. (2013). *Computational thinking: The developing definition*. University of Southampton.

Shoji, K., Cieslak, R., Smoktunowicz, E., Rogala, A., Benight, C. C., & Łuszczyńska, A. (2016). Associations between job burnout and self-efficacy: A meta-analysis. *Anxiety, Stress, & Coping*, 29(4), 367–386.

https://doi.org/10.1080/10615806.2015.1058369.

Täschner, J., Dicke, T., Reinhold, S., & Holzberger, D. (2025). "Yes, I can!" A systematic review and meta-analysis of intervention studies promoting teacher self-efficacy. *Review of Educational Research*, 95(1), 3–52. https://doi.org/10.3102/00346543231221499.

Taslibeyaz, E., Kursun, E., & Karaman, S. (2020). How to Develop Computational Thinking: A Systematic Review of Empirical Studies, *Informatics in Education*, 19(4), 701–719. https://doi.org/10.15388/infedu.2020.30.

Tschannen-Moran, M., & McMaster, P. (2009). Sources of self-efficacy: Four professional development formats and their relationship to self-efficacy and implementation of a new teaching strategy. *The Elementary School Journal*, 110(2), 228–245. https://doi.org/10.1086/605771.

Tschannen-Moran, M., & Woolfolk Hoy, A. (2001). Teacher efficacy: Capturing an elusive construct. *Teaching and Teacher Education*, 17(7), 783–805. https://doi.org/10.1016/S0742-051X(01)00036-1.

Voogt, J., Fisser, P., Good, J., Mishra, P., & Yadav, A. (2015). Computational thinking in compulsory education: Towards an agenda for research and practice. *Education and Information Technologies*, 20(4), 715–728. https://doi.org/10.1007/s10639-015-9412-6.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology*, 25(1), 127–147. https://doi.org/10.1007/s10956-015-9581-5.

Wilson, C., & Guzdial, M. (2010). How to make progress in computing education. *Communications of the ACM*, 53(5), 35–37. https://doi.org/10.1145/1735223.1735235.

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

Wing, J. M. (2008). Computational thinking and thinking about computing. In *2008 IEEE International Symposium on Parallel and Distributed Processing*, IEEE, 1–9. https://doi.org/10.1109/IPDPS.2008.4536091.

Yadav, A., Stephenson, C., & Hong, H. (2017). Computational thinking for teacher education. *Communications of the ACM*, 60(4), 55–62. https://doi.org/10.1145/2994591.

Yadav, A., Zhou, N., Mayfield, C., Hambrusch, S., & Korb, J. T. (2011). Introducing computational thinking in education courses. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education,* ACM,. 465–470. https://doi.org/10.1145/1953163.1953297.

Younas, A., Inayat, S., & Sundus, A. (2021). Joint displays for qualitative-quantitative synthesis in mixed methods reviews. *Research Methods in Medicine & Health Sciences*, 2(3), 91–101. https://doi.org/10.1177/2632084320984374.

Yurdakök, E. A., & Kalelioğlu, F. (2024). The effect of teaching physical programming on computational thinking skills and self-efficacy perceptions towards computational thinking. *Journal of Educational Computing Research*, 62(3), 785–815. https://doi.org/10.1177/07356331231220313.

Zee, M., & Koomen, H. M. Y. (2016). Teacher self-efficacy and its effects on classroom processes, student academic adjustment, and teacher well-being: A synthesis of 40 years of research. *Review of Educational*

Research, 86(4), 981–1015. https://doi.org/10.3102/00346543156268O1.

Corresponding Author Contact Information:

Author name: Susan German

Department: School of Information Science and Learning Technologies

University, Country: University of Missouri, Columbia, United States

Email: susangermansciencteacher@gmail.com

Please Cite: German, S. (2026). Exploring Fifth-Grade Teachers' Integration of Computational Thinking, Instruction Strategies, and Efficacy Following Professional Development. *Journal of Research in Science, Mathematics and Technology Education, 9*(SI), 31- 59. https://doi.org/10.31756/jrsmte.912SI

Conflict of Interest: The author declares no conflict of interest.

Publisher's Note: All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

Data Availability Statement: Data available on request from the corresponding author.

Ethics Statement: Your Annual Exempt Form to project entitled 5th Grade Teacher Self-Efficacy Regarding Computational Thinking was reviewed and approved by the MU Institutional Review Board according to the terms and conditions described below:
IRB Project Number 2099651
IRB Review Number 443591
Initial Application Approval Date December 22, 2023
Approval Date of this Review October 25, 2024
IRB Expiration Date December 22, 2025
Level of Review Exempt
Project Status Active - Exempt
Risk Level Minimal Risk
HIPAA Category No HIPAA

Author Contributions: The author did all the work.

Received: February 11, 2026 ▪ Accepted: June 07, 2026