Computational Thinking and Self-Regulated Learning as Drivers of Programming Achievement: Evidence from a Systematic Review

Lokman Zulkifley[1], Mohd Hishamuddin Abdul Rahman[1,*], Nor Hasbiah Ubaidullah[2]

[1]Department of Computer Science and Digital Technology, Faculty of Computing and Meta-Technology, Universiti Pendidikan Sultan Idris, 35800 Tanjong Malim, Perak, Malaysia

[2]Department of Software Engineering and Smart Technology, Faculty of Computing and Meta-Technology, Universiti Pendidikan Sultan Idris, 35800 Tanjong Malim, Perak, Malaysia

[*]Corresponding Author

DOI: https://doi.org/10.47772/IJRISS.2026.100500791

Received: 18 May 2026; Accepted: 23 May 2026; Published: 13 June 2026

ABSTRACT

The increasing demand for stronger programming skills has created additional challenges for educators who must also foster students' technical understanding and metacognitive development. This systematic review examines how integrating Computational Thinking (CT) with Self Regulated Learning (SRL) operates as a pedagogical strategy within programming instruction. Based on findings from fifteen research studies published from 2020 to 2025, the review compiles evidence on instructional frameworks, technological supports, and learning outcomes associated with CT and SRL integration. Findings across the studies indicate that programming instruction that integrates SRL practices such as goal setting, progress monitoring, and reflective thinking can promote greater learner autonomy, deeper conceptual understanding, and stronger problem solving abilities. Technology supported learning environments, including adaptive tutoring systems and coding dashboards, contribute to this process by offering timely feedback that encourages sustained metacognitive engagement throughout programming activities. Overall, the review provides several recommendations that can assist educators in creating meaningful and engaging learning environments. Effective integration of CT and SRL has the potential to strengthen both cognitive and metacognitive development among students learning to program.

Keywords - Computational Thinking, Instructional Frameworks, Self-regulated Learning, and Programming Education

INTRODUCTION

Programming is increasingly recognised as a core competence that enables creativity, innovation, and effective problem solving across educational and professional domains, in addition to being an essential technical skill (Liu et al., 2024; Sobral, 2021). As programming becomes embedded in more areas of study and work, the need to strengthen learners' abilities in this field has gained importance. Despite this recognition, programming remains cognitively demanding for many learners. Beginners often struggle to grasp abstract concepts, apply systematic problem solving strategies, and sustain motivation when dealing with complex and iterative tasks (Lahtinen et al., 2022; Ubaidullah et al., 2021).

To address these challenges, two educational perspectives, CT and SRL, have attracted considerable academic attention. CT is described as a cognitive process in which learners break down problems and design sequential algorithmic steps that can be interpreted by a computer (Adomi et al., 2025; Rao & Bhagat, 2024).). Elements such as abstraction, decomposition, pattern recognition, and algorithm construction form the foundation of CT and are directly relevant to programming pedagogy (Ubaidullah et al., 2021; Bocconi et al., 2022).

SRL focuses on a learner's capacity to regulate cognitive, emotional, and behavioural processes in order to reach academic goals (Kitsantas et al., 2025; Robles Mucho et al., 2026). Strategies like establishing goals, monitoring one's progress, and engaging in reflective evaluation are particularly important in programming tasks, where learners must maintain focus and continuously refine their work (Bannert et al., 2021).

Although CT and SRL each demonstrate positive effects when implemented individually, studies examining their combined influence remain limited. Most research positions CT as a set of competencies and SRL as a learning support, yet evidence on how these two constructs interact to enhance programming performance is still developing (Li et al., 2025; Curi et al., 2025). With growing emphasis on learner autonomy and personalised instruction, examining the complementary roles of CT and SRL is both timely and necessary.

Drawing from empirical studies published between 2020 and 2025, this systematic review aims to address this gap by synthesising how CT and SRL are conceptualised, implemented, and assessed in programming education. The discussion is structured around the key research questions that guide the review. The following are the main research questions:

RQ1. In what ways does CT improve student learning in programming education?

RQ2. How can SRL aid in the development of programming skills?

RQ3. What results have been documented from the integration of CT and SRL techniques in programming education?

RQ4. What are the gaps in the literature that currently exists on the integration of CT and SLR in programming education?

The review's conclusions are intended to guide the creation of research-based teaching strategies that help students in programming settings build their CT and SLR skills.

LITERATURE REVIEW

Programming literacy is increasingly recognised as an essential competence across scientific, technical, and educational settings. Learners are now expected to write, analyse, and debug code effectively, yet many face substantial cognitive demands when engaging with programming tasks (Tariqnet al., 2025; Pérez Marín et al., 2022). Challenges in grasping fundamental programming ideas, developing algorithmic reasoning, and fixing errors frequently lead to learner frustration and higher withdrawal rates in beginner level courses (Liu et al., 2024; Lahtinen et al., 2021).

These challenges have motivated researchers to explore instructional strategies that enhance both cognitive skills and the metacognitive abilities required for sustained engagement and deeper learning.

Developing Programming Skills: Opportunities and Difficulties

Programming is widely regarded as a central competence for the twenty first century because it supports creativity, critical thinking, and digital literacy (Sobral, 2021). Yet many novices struggle with interpreting syntax, understanding abstract structures, designing algorithms, and breaking down complex tasks into manageable components (Lahtinen et al., 2022; Robins et al., 2003). These obstacles can diminish confidence and lead to withdrawal from computing courses. Consequently, there is growing emphasis on approaches that cultivate higher order thinking and effective self management. Within this evolving landscape, CT and SRL have emerged as important frameworks for strengthening programming education.

CT in the Teaching of Programming

CT is commonly described as a structured and strategic way of approaching problems. Adomi et al. (2025) identifies several core components, including algorithmic thinking, decomposition, pattern recognition, and abstraction. Rao and Bhagat (2024) also highlight debugging as a crucial element of this reasoning process. Liu

et al. (2024) argue that these practices form the conceptual foundation needed to produce efficient and well organised code. Although rooted in computer science, CT is now recognised as a transferable competence across disciplines and education levels (Bocconi et al., 2022; Korte et al., 2025).

CT encourages methodical and imaginative engagement with coding tasks. Liu et al. (2024) note that it helps learners move beyond trial and error by promoting deliberate planning and structured analysis. Empirical work by Kong and Wang (2024) and Curi et al. (2025) shows that explicit CT instruction enhances problem analysis and programming logic. More recent studies confirm its relevance in STEM contexts that require learners to model systems and interpret complex structures (Mills et al., 2024; Tariq et al., 2025). However, educators still face challenges such as insufficient resources and limited professional development for CT based instruction (Sunday et al., 2024).

Despite these limitations, research consistently reports positive outcomes. Mills et al. (2024) found that embedding CT into curriculum design improved reasoning and problem solving abilities. Choi and Choi (2024) show that visual programming platforms such as Code dot org support the development of CT and also increase learner motivation. However, these initiatives require strong institutional support and targeted teacher preparation for long term sustainability (Sunday et al., 2024). Collectively, these findings highlight CT as a key foundation for preparing learners for technology centred careers.

SRL as a Catalyst for Programming Success

SRL provides a complementary perspective by focusing on how learners guide their cognitive, motivational, and behavioural processes. According to Kitsantas et al. (2025), Zimmerman's SRL framework describes learners as active participants who manage their own learning processes through goal setting, strategic planning, continuous self-monitoring, and reflective evaluation of their performance. In programming contexts, this means preparing strategies for complex tasks, tracking progress during coding, and evaluating approaches during problem solving. Robles Mucho et al. (2026) describes SRL as consisting of forethought, performance, and reflection phases.

A wide body of research links strong SRL abilities to improved performance, persistence, and deeper understanding. Programming tasks often involve uncertainty and repeated failure, and SRL helps learners cope more effectively with these challenges. Kong and Liu (2023) reported that using a performance-based assessment system enhanced primary pupils' abilities in setting goals, organising their time, and evaluating their own learning.

Watanabe et al. (2025) demonstrate that TrackThinkDashboard supported learners in monitoring coding behaviours and reflecting on their progress. Li and Ma (2025) describe CodeRunner Agent, an artificial intelligence supported tool that provides strategy oriented feedback to improve planning and monitoring. These studies indicate that SRL encourages autonomy and resilience in programming education.

CT and SRL Integration in Programming

While CT and SRL represent different theoretical domains, they complement each other effectively in programming contexts. CT equips learners with the cognitive skills needed to interpret problems and design workable solutions, whereas SRL contributes the metacognitive processes that help students plan their actions, keep track of their progress, and adjust their approaches as needed (Li et al., 2025).

Yang et al. (2020) found that planning prompts, reflective tools, and think aloud activities enhanced metacognitive awareness and improved performance on tasks involving algorithm design. Druga (2025) reports that the Cognimates platform, which integrates artificial intelligence, robotics, and creative building tasks, supported the development of both CT and SRL abilities. Choi and Choi (2024) show that activities involving Code dot org, robotics, and artificial intelligence strengthened reflective design thinking, demonstrating growth in both computational and metacognitive domains.

Across the reviewed studies, the evidence shows that programming instruction is most effective when learners are guided in how to think computationally and how to regulate their thinking. CT provides structural reasoning

for analysing programming problems, while SRL cultivates adaptable and reflective learners capable of applying their skills across varied contexts. Studies published from 2020 to 2025, summarised in Table 1, offer a foundation for developing integrated instructional models that promote both CT and SRL.

Table 1: Review of Significant CT and SRLResearch In Programming Education Conducted During 2020-2025.

Year	Authors	Methodology	Population / Sample	Key Findings
2020	Yang et al.	Conducted an experimental investigation.	Involved primary level learners in Spain.	Integrating CT with metacognitive techniques in Spanish primary school pupils enhanced autonomy and computational problem-solving skills.
2023	Kong and Liu	Adopted a quasi-experimental method combined with learning platform analytics.	Focused on schoolchildren using an SRL supported learning system.	The platform enhanced programming performance and self-regulation (target setting, monitoring).
2024	Mills et al.	Performed a systematic synthesis of existing literature	Reviewed more than sixty studies covering learners from kindergarten to pre-university (K-16).	Integrating CT improves critical thinking, yet there are no standardised methods for assessment.
2024	Choi and Choi	Employed a quantitative pre-test–post-test research design.	Worked with primary school pupils in South Korea.	Block-based coding enhanced CT abilities, motivation, and achievement in elementary school pupils in South Korea.
2024	Sunday et al.	Integrated a systematic review with a design-based research approach.	Examined teaching and learning practices among K-12 STEM teachers and students.	Co-design pedagogy was suggested as a successful methodology when obstacles to CT were identified.
2025	Watanabe et al.	Used a design based research framework.	Involved undergraduate students in Japan.	Students were able to visualise and consider SRL techniques while working on programming challenges.
2025	Li and Ma	Experimental design with the use of AI supported tools	Both secondary schools and universities.	AI tools CodeRunner Agent increased SRL for both high school and college students by providing tailored feedback and assistance with strategic planning.
2025	Druga	Qualitative + platform development.	Children aged 7-13 using Cognimates.	AI design, coding, and exploration serve to cultivate CT and SRL in a fun, imaginative setting.
2025	Li et al.	Quantitative survey.	487 high school students in China	SRL links programming confidence to CT, and cognitive style affects how strongly students benefit.

METHODOLGY

This review brought together scholarly publications released between 2020 and 2025 that examined how CT) and SRL are integrated into programming education. A systematic literature review (SLR) approach was used, and the procedures were guided by the reporting standards outlined in PRISMA to ensure clarity, consistency, and replicability of the review process (Page et al., 2021).

Inclusion and Exclusion Standards

To ensure that only high-quality and relevant research was considered, the study employed a clearly defined set of selection rules. These criteria, summarised in Table 2, were used to filter studies during the screening process.

Table 2: *S*ummary of Inclusion and Exclusion Criteria Used in the Review

Criteria	Included Studies	Excluded Studies
Publication Window	Research published between January 2020 and May 2025, reflecting current trends and developments.	Any publications dated before 2020.
Language of Publication	Studies written in English, allowing consistent interpretation.	Work published in languages other than English.
Type of Source	Scholarly journal articles that have undergone peer review.	Non-refereed materials, including theses, conference abstracts, grey literature, editorials.
Research Focus	Studies that address the intersection of CT and/or SRL within programming or computing education.	Research that does not relate to CT, SRL, programming, or computer science instruction.
Educational Setting	Investigations carried out in school classrooms, university courses, or teacher-education programmes.	Studies conducted in business, industrial, or informal/non-academic training contexts.

Data Sources and Search Strategy

Searches were carried out across several major scholarly databases namely *Scopus, Web of Science, ERIC, the ACM Digital Library,* and *IEEE Xplore*. A customised Boolean query was used to capture studies linking CT, SRL, and programming-related education:

("computational thinking" OR CT) AND ("self-regulated learning" OR SRL) AND ("programming instruction" OR "coding practices" OR "computer science teaching")

This search strategy initially returned 214 records, all of which were transferred into Zotero to remove duplicates and to organise the citations for further screening.

Screening and Selection Procedures

The selection of studies followed the four stages outlined in the PRISMA framework:

1. Identification

Total records located through database searches (n = 214):
Scopus (n = 72)
Web of Science (n = 48)
ERIC (n = 35)

ACM Digital Library (*n* = 29)
IEEE Xplore (*n* = 30)

2. Screening

Duplicate records removed (*n* = 25)
Records after deduplication (*n* = 189)
Titles and abstracts screened (*n* = 189)
Records excluded at this stage (*n* = 144)

3. Eligibility

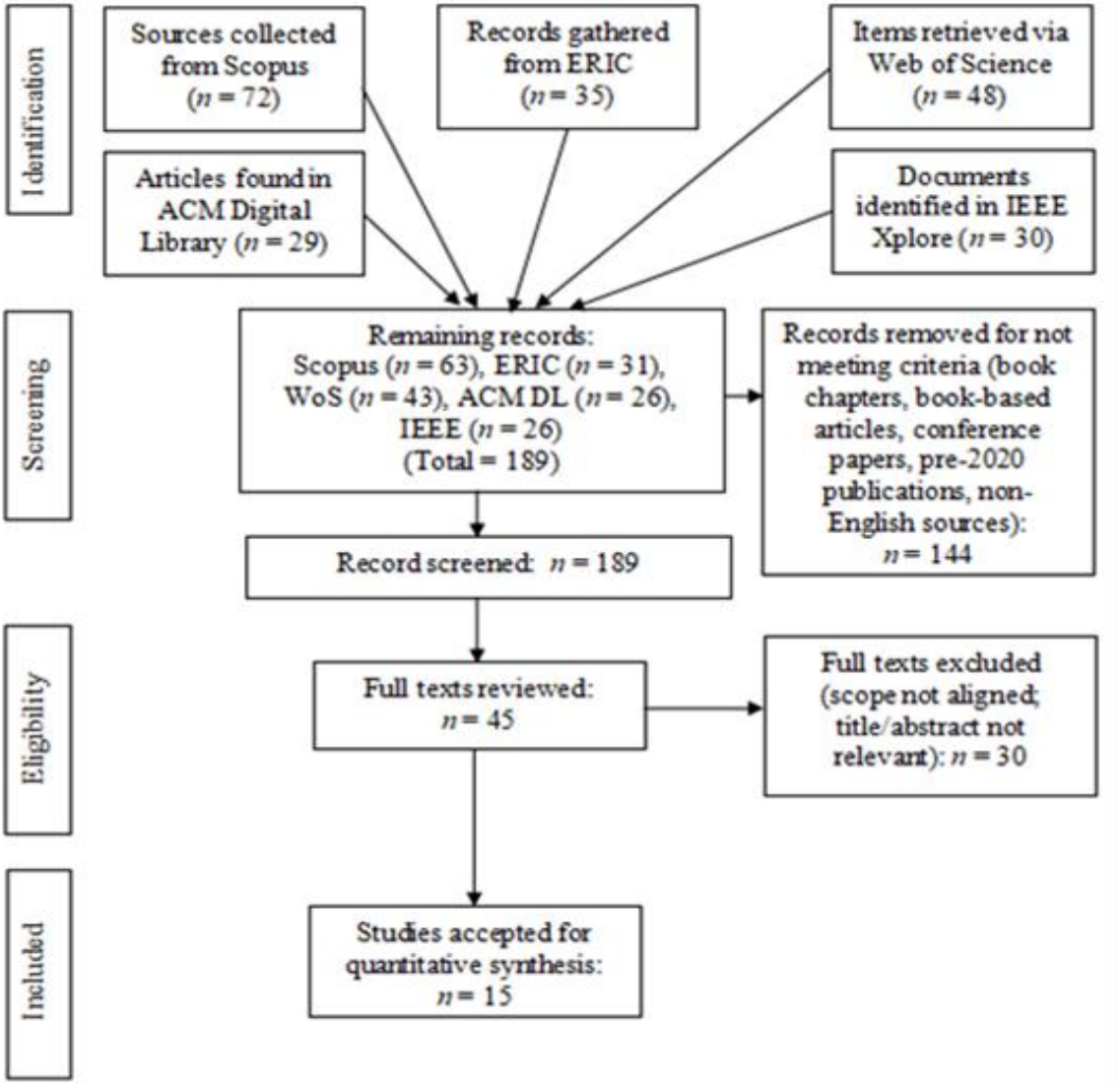
Full-text articles reviewed (*n* = 45)
Full-text exclusions (with documented reasons) (*n* = 30)

4. Included

Studies meeting all criteria and included in the qualitative synthesis (*n* = 15)

Figure 1 provides an overview of the screening and selection stages using the PRISMA flow diagram.

Figure 1: Flowchart Illustrating the Search Strategy Used in this Study



Data Extraction and Analysis

Relevant data were systematically gathered from the selected studies using a predefined coding framework. Key variables included the year of publication, names of the authors, participant information, educational settings, and the nature of the intervention, encompassing instructional tools, digital platforms, and teaching methodologies. Additionally, the integration of CT and SRL, study limitations, and measured outcomes were recorded.

The process of thematic synthesis was carried out in three distinct stages. Initially, findings were coded to capture core ideas and concepts. These codes were then grouped to form descriptive themes. Finally, analytical themes were constructed to reflect deeper insights aligned with the study's research questions. This rigorous approach enabled the identification of recurring patterns and significant gaps within the existing body of literature.

FINDINGS

Analysis of the fifteen peer-reviewed studies produced four central themes that closely address the research questions, namely (i) strategies for integrating CT in programming instruction; (ii) approaches that enhance learning through SRL; (iii) combined CT and SRL learning experiences; and (iv) current limitations and future needs in CT and SRL research.

Theme 1: Integrating CT within programming instruction

Sub-theme 1.1: Learning modules and curriculum infused with CT

Studies published between 2020 and 2025 consistently show increasing emphasis on embedding CT as part of programming instruction. Instead of treating CT as an adjunct, researchers highlight its role as a cognitive toolkit, including decomposition, abstraction, algorithmic thinking, and debugging, which are essential for managing programming complexity. Curriculum designs incorporating CT intentionally align these concepts with coding skills rather than presenting them separately. For example, Liu et al. (2024) reported the use of scaffolds, structured problem-solving activities, and explicit CT terminology in K–12 programming units. Similarly, Tareq and Yusof (2024) found that STEM-based modules integrating CT improved both conceptual knowledge and coding proficiency.

A recurring point across these studies is that CT integration is most effective when it aligns tightly with course outcomes and instructional sequences (Mills et al., 2024). Many researchers also underline the value of blending SRL components into CT-oriented teaching. Yang et al. (2020), Li et al. (2025), and Ubaidullah et al. (2021) show that deliberate use of strategies such as planning, progress monitoring, and reflective questioning helps learners navigate CT processes more effectively. These designs draw on scaffolding approaches such as reflective journals, guiding prompts, planning tools, and think-aloud exercises, all of which contribute to strengthening both CT skills and SRL behaviours.

Sub-theme 1.2: CT focused pedagogies

Problem-based learning (PBL), project-based learning, design thinking, and collaborative coding increasingly appear in programming instruction because these methods naturally cultivate CT. Adorni et al. (2024) introduced the FADE-CTP framework, which calibrates task complexity with student reasoning processes. According to Tivka and Tambouris (2021), beginners' logical reasoning improved when they engaged in CT-specific strategies such as flowcharting and algorithm design exercises. These pedagogical orientations consistently foster iterative thinking, experimentation, and evidence-based reasoning, which are central to CT.

Sub-theme 1.3: Scaffolding and guided practice for CT tasks

The reviewed studies also emphasise the role of scaffolding in enabling learners, particularly novices, to manage the abstract nature of programming. Guided CT tasks help students master complex ideas such as recursion, iteration, and conditional logic. Zhang et al. (2024) demonstrated that structured, step-by-step CT tasks enhanced comprehension and retention, while Ubaidullah et al. (2021) highlighted the importance of timely teacher

feedback and guided questioning in helping students regulate their cognitive processes during algorithmic problem-solving.

Across this theme, numerous studies (for example, Choi & Choi, 2024; Mills et al., 2024; Yang et al., 2020) reinforce that CT integration strengthens students' motivation, logical reasoning, and engagement. Effective curriculum design, hands-on activities, and alignment of CT objectives with programming tasks are essential in cultivating computational problem-solvers.

Theme 2: Learning enhancement through SRL strategies

Theme 2 concerns how SRL approaches support the acquisition of programming and CT competencies through planning, monitoring, reflection, and self-assessment.

Sub-theme 2.1: Use of metacognitive strategies

A large portion of the literature highlights metacognition, including goal setting, self-monitoring, and reflection, as central to programming success. Kong and Liu (2023) proposed a metacognitive framework that strengthened students' ability to identify errors and understand underlying programming concepts. Ubaidullah et al. (2021) similarly demonstrated that self-monitoring and reflective engagement support the development of strong problem-solving skills. Metacognitive prompts embedded into coding tasks help cultivate awareness and strategic control over learning.

Sub-theme 2.2: Self-evaluation and goal-setting

Goal-directed behaviour consistently correlates with improved programming outcomes. Yang et al. (2020) showed that systematic goal-setting within programming modules enhanced learner engagement and achievement. Choi and Choi (2024) reported similar findings, noting that students who articulated and tracked their CT related goals achieved better learning results. Encouraging students to evaluate their progress nurtures intrinsic motivation and sustained persistence.

Sub-theme 2.3: SRL supporting environments and tools

Technological environments that provide progress tracking, guided feedback, and learning analytics are widely used to support SRL. Liu et al. (2024) demonstrated that digital portfolios and feedback loops allowed students to monitor CT related progress in programming activities. Romero (2024) described online systems enabling learners to advance at their own pace while receiving continuous monitoring and support. Collectively, these environments help improve resilience (Ubaidullah et al., 2021; Arakawa et al., 2021), conceptual understanding (Kong & Liu, 2023), and motivation (Choi & Choi, 2024).

Theme 3: Synergies between CT and SRL in programming education

This theme explores how CT and SRL complement one another to enrich programming instruction and reinforce active, learner-centred engagement.

Sub-theme 3.1: Merging CT processes with SRL strategies

Instructional models that explicitly combine CT practices, such as decomposition, abstraction, and debugging, with SRL processes, including planning, monitoring, and reflecting, are becoming more prominent. Many of these approaches adopt project-based, inquiry-driven, or gamified learning structures. For instance, Kong and Liu (2023) showed that their metacognitive CT framework improved students' understanding of problem-solving strategies and their ability to manage time and effort during programming tasks.

Sub-theme 3.2: Technology-Supported CT and SRL Integration

Digital tools such as Scratch, Code.org, LMS dashboards, and feedback-rich learning platforms enhance the integration of CT and SRL by making learning progress visible and actionable. These tools offer automated feedback, visualisations, and interactive coding environments that guide students through problem breakdown,

planning, iteration, and evaluation. Liu et al. (2024) highlighted how such tools enable teachers to identify and support SRL behaviours in real time.

Sub-theme 3.3: Cognitive and Behavioural Gains

Studies provide strong evidence that when CT and SRL are combined, students demonstrate notable improvements in algorithmic reasoning, problem-solving, and overall CT. Ubaidullah et al. (2021) found that learners in CT and SRL supported environments outperformed peers in computational logic and self-directed learning. Students also approached programming tasks more systematically and with greater conceptual depth.

Sub-theme 3.4: Emotional and Motivational Outcomes

The integration also contributes to learners' emotional and motivational development. Choi and Choi (2024) identified clear links between SRL-infused CT tasks and heightened motivation, confidence, and engagement. Sunday et al. (2024) similarly found that co-created CT and SRL learning experiences improved participation and sustained involvement in STEM contexts. Learners using block-based systems with SRL scaffolds reported strengthened self-efficacy and improved CT competencies.

Across the studies, CT and SRL integration has been shown to strengthen problem-solving and CT performance, increase learner autonomy and perseverance, enhance programming accuracy and algorithmic development, and boost motivation and engagement.

Theme 4: Challenges and future directions in CT and SRL integration

Sub-theme 4.1: Fragmented frameworks

Few studies provide holistic frameworks that unify CT and SRL. Many focus on one domain independently. Work by Kong and Liu (2023) and Romero (2024) is among the few attempts to address this gap. More systematic design and evaluation are needed.

Sub-theme 4.2: Limited longitudinal and experimental research

Most studies use cross-sectional or exploratory approaches. There is insufficient evidence on long-term effects or sustained behavioural change resulting from CT and SRL integration. Longitudinal and experimental research designs are recommended.

Sub-theme 4.3: Inconsistent assessments

Research varies widely in how CT and SRL are assessed. Some rely on surveys and others on specific CT tasks. Lack of unified assessment frameworks limits comparability. More comprehensive, validated tools are needed.

Sub-theme 4.4: Underrepresentation of diverse learners

Most studies originate from high-income educational settings. Learners with disabilities, low-resource contexts, and culturally diverse groups remain underexamined. Future research must address inclusion and equity.

Sub-theme 4.5: Limited Focus on Teachers

Teacher roles in CT and SRL integration are rarely addressed. More studies should explore teacher training, instructional design, and teacher-student co-design of scaffolds and adaptive supports.

Sub-theme 4.6: Limited use of advanced technologies

Although AI, learning analytics, and intelligent tutoring show potential, their application remains limited. It is recommended that subsequent research analyse the potential of these technologies to support CT and SRL tasks through adaptive prompts and real-time feedback.

Final Reflection and Way Forward

The literature consistently demonstrates that integrating CT with SRL produces meaningful improvements in cognitive, behavioural, and motivational aspects of programming learning. Successful interventions share three common elements: (i) explicit SRL principles embedded within CT tasks, (ii) technology-enabled environments that provide actionable feedback, and (iii) authentic, real-world programming projects that promote independence. This synthesis provides a grounding framework for future instructional design and research aimed at nurturing self-regulated computational thinkers. The main ideas and corroborating research in CT and SRL integration are compiled in Table 3.

Table 3: Main Themes And Representative Studies Supporting CT and SRL Integration

Research Question	Theme	Sub-Themes	Description	Key References (2020–2025)
RQ1: In what ways does CT improve student learning in programming education?	Theme 1: Integrating CT within programming instruction.	Learning modules and curriculum infused with CT, CT focused pedagogies and scaffolding and guided practice for CT tasks.	Various instructional strategies have been employed to embed CT into education, such as integrating it into mathematics and science curricula, using platforms like Scratch and Code.org, and adopting game-based or project-based learning approaches.	Mills et al. (2024); Liu et al. (2024); Choi & Choi (2024); Voon et al. (2022); Sunday et al. (2024)
RQ2: How can SRL aid in the development of programming skills?	Theme 2: Learning enhancement through SRL strategies.	Use of metacognitive strategies, self-evaluation and goal-setting, and SRL-supporting environments and tools.	SRL supports programming skill development by guiding learners through phases of planning, monitoring, and reflection. Instructional frameworks designed around SRL foster independent learning and enhance problem-solving capabilities.	Kong & Liu, (2023); Ubaidullah et al. (2021); Tivka & Tambouris (2021); Yang et al. (2020)
RQ3: What results have been documented from the integration of CT and SRL techniques in programming education?	Theme 3: Synergies Between CT and SRL in programming education	Merging CT processes with SRL strategies, technology-supported CT and SRL integration, cognitive and behavioural gains, and emotional and motivational outcomes.	Integrated approaches that embed SRL within CT-related activities highlight the importance of structured guidance. Digital learning environments have also been shown to promote SRL behaviours while	Tariq et al. (2025); Romero (2024); Adorni et al. (2024); Zhang et al. (2024); Merino-Armero et al. (2022)

Research Question	Theme	Sub-Themes	Description	Key References (2020–2025)
			facilitating CT-based problem-solving, often resulting in improved learner engagement, persistence, and academic performance.	
RQ4: What are the gaps in the literature that currently exists on the integration of CT and SLR in programming education?	Theme 4: Challenges and future directions in CT and SRL integration.	Fragmented frameworks, limited longitudinal and experimental research, inconsistent assessments, underrepresentation of diverse learners, limited focus on teachers, and limited use of advanced technologies.	Despite these promising findings, the intersection of CT and SRL remains an emerging field. Current research is limited by a lack of longitudinal studies, insufficient empirical evidence on SRL's specific influence on CT development, and a narrow focus on diverse learner demographics.	Liu et al. (2024); Tariq et al. (2025); Romero (2024); Mills et al. (2024); van der Lubbe et al. (2025)

DISCUSSION

Fifteen academic studies from 2020-2025 were analysed to understand how CT and SRL are incorporated into programming education. These studies were drawn from established academic databases including IEEE Xplore, Web of Science, ERIC, ACM Digital Library, and Scopus, which ensures that the review is grounded in credible sources. This discussion explains how CT and SRL function in instructional environments, how they shape learner development in programming, and which areas of research remain insufficiently explored. The structure reflects the themes and research questions derived from the systematic review.

Theme 1: Integrating CT within programming instruction

RQ1: In what ways does CT improve student learning in programming education

Across the reviewed studies, CT is presented as a core pedagogical priority. Educators adopt various strategies to embed CT into programming tasks. Common approaches include block based tools such as Scratch and Blockly, robotics materials including Arduino and LEGO Mindstorms, and unplugged activities designed to promote algorithmic reasoning without digital devices (Yang et al., 2020; Mills et al., 2024). These resources reduce early difficulty and encourage active exploration. Block based environments shift attention away from syntax and toward structure, sequencing, and logical flow. Robotics provides an immediate connection between written code and observable behaviour, making abstract reasoning more concrete and strengthening iterative thinking skills needed for debugging.

Approaches grounded in inquiry, including Problem Based Learning and Project Based Learning, appear frequently across CT research. These models require learners to investigate authentic problems, design solutions, build working programs, and justify their thinking. Such learning processes mirror professional programming practice and reinforce collaboration, adaptability, and systematic reasoning (Yang et al., 2020).

Teacher readiness is consistently highlighted as essential for successful CT instruction. Liu et al. (2024) emphasise that teachers must know how to scaffold tasks, encourage analytical thinking, and guide problem solving without taking over students' decisions. Platforms such as Code dot org provide structured content and professional development opportunities that support this work (Choi & Choi, 2024).

Real world tasks also contribute strongly to CT development. Activities involving weather modelling, budgeting tools, or scientific simulation require learners to apply CT principles meaningfully. These experiences strengthen decomposition, pattern recognition, and algorithmic reasoning (Voon et al., 2022; Merino-Armero et al., 2022). Gamified learning platforms and competitions further motivate learners and provide supportive settings in which to practice these skills.

Across the studies, there is a clear consensus that successful CT instruction requires deliberate curriculum design, engaging learning environments, and structured scaffolds that enable learners to progress from superficial coding toward deeper computational reasoning.

Theme 2: Learning enhancement through SRL strategies

RQ2: How can SRL aid the development of programming skills?

SRL appears throughout the reviewed studies as a crucial factor in programming success. Programming requires learners to manage complexity, analyse errors, make strategic choices, and revise their work. SRL provides skills such as planning, setting goals, monitoring progress, and reflecting on performance, which help learners navigate these cognitive demands (Kong & Liu, 2023; Ubaidullah et al., 2021).

Studies by Romero (2024) and Choi and Choi (2024) show that learners who use SRL strategies demonstrate increased persistence, confidence, and willingness to attempt alternative solutions. Digital dashboards and learning analytics strengthen these behaviours by allowing learners to observe patterns, identify weaknesses, and receive reminders to reflect and adjust.

Reflective journals serve as an effective SRL tool. Recording challenges and reasoning processes encourages metacognitive awareness and helps learners refine their strategies. Robles Mucho et al. (2026) and Tivka and Tombouris (2021) report that structured reflection improves conceptual understanding and adaptability.

Environments that emphasise metacognition, such as predicting debugging outcomes or evaluating solution paths, support programming fluency. Adaptive feedback and analytics provide targeted support that addresses misconceptions and strengthens problem solving.

Zhang et al. (2024) highlight intelligent tutoring systems that personalise SRL development by analysing learner behaviour and adjusting feedback accordingly. These systems reduce cognitive load, support autonomy, and are especially effective in online and self paced environments.

Overall, SRL enhances independence, persistence, and strategic thinking, making it a powerful complement to CT based instruction.

Theme 3: Synergistic integration of CT and SRL in programming

RQ3: What results have been documented from the integration of CT and SRL techniques in programming education?

Recent technological advances allow CT and SRL to be integrated into unified instructional ecosystems. Several studies describe systems in which CT tasks are embedded within cycles of planning, performance, monitoring, and reflection (Adorni et al., 2024; Zhang et al., 2024).

Adorni et al. (2024) describe a model that includes metacognitive prompts throughout the programming process. Learners begin by identifying goals, receive automated support during task execution, and complete structured reflection activities. This approach results in improved accuracy, motivation, and self belief.

Adaptive learning environments described by Romero (2024) and Tariq et al. (2025) personalise instruction by analysing both CT abilities and SRL behaviours. These systems adjust task difficulty and feedback timing. Learners within such systems demonstrate stronger time management and improved problem solving.

However, some gaps remain. Yang et al. (2020) note that many robotics based CT activities omit explicit SRL support. Although learners complete tasks successfully, they may not develop transferable metacognitive skills. Integrating SRL scaffolds within robotics can strengthen long term learning outcomes.

Overall, the literature suggests that CT and SRL should be integrated rather than taught separately. Planning should inform coding choices, monitoring should guide revisions, and reflection should consolidate understanding. When woven together, these processes lead to deeper engagement and greater ownership of learning.

Theme 4: Challenges and future directions in CT and SRL integration

RQ4: What are the gaps in the literature that currently exists on the integration of CT and SLR in programming education?

Despite increased interest, research on CT and SRL integration contains several notable gaps. Longitudinal research is limited, with most studies reporting short term improvements without examining long term development (van der Lubbe et al., 2025). Cultural and geographic diversity is also limited, as most studies come from Western contexts (van der Lubbe et al., 2025). Underrepresented groups, including rural learners, students with disabilities, and those from low income backgrounds, are seldom included (Sunday et al., 2024). Higher education and informal learning contexts receive little attention, and teacher preparation remains insufficient (Tivka & Tombouris, 2021).

Future work should emphasise inclusive design, scalable instructional models, and assessments that capture both CT and SRL processes. Emerging artificial intelligence tools also offer promising opportunities for personalised, real time support.

CONCLUSION

This review synthesised insights from fifteen peer-reviewed studies published between 2020 and 2025 to explore how CT and SRL have been integrated into programming education. Four primary themes were identified: the approaches used to incorporate CT, the role of SRL as a cognitive support, the joint application of CT and SRL in instructional contexts, and prevailing research gaps that signal areas for further inquiry.

The findings suggest an increasing trend in the use of innovative pedagogical tools such as robotics, block-based programming environments, and authentic problem-solving tasks to embed CT concepts like abstraction, pattern recognition, and algorithmic thinking. SRL, in turn, plays a foundational role in supporting students' ability to self-manage their learning through planning, monitoring, and reflection.

When implemented in a coordinated manner, CT and SRL foster not only cognitive growth but also learner autonomy, engagement, and resilience. However, the current body of literature remains fragmented, with few longitudinal studies and limited attention to inclusive or cross-contextual instructional models. There remains a strong need for future studies to create reliable assessment instruments, gather long-term evidence on learner development, and design teaching models that can adapt to diverse educational settings and student groups. In conclusion, the integration of CT and SRL represents a promising pedagogical paradigm that can help students build the competencies required to succeed in a rapidly changing digital landscape.

As educational systems continue to adapt to global shifts in technology and workforce demands, researchers and practitioners must work collaboratively to ensure that programming education is both technically rigorous and metacognitively enriching.

AUTHOR CONTRIBUTIONS

Lokman Zulkifley: Conceptualization, data curation, original draft preparation. Mohd Hishamuddin Abdul Rahman: Supervision, writing- reviewing and editing. Nor Hasbiah Ubaidullah: writing- reviewing and editing.

ACKNOWLEDGEMENTS

The authors would like to thank the Faculty of Computing and Meta-Technology and Institute of Graduate Study, Universiti Pendidikan Sultan Idris, Tanjong Malim, Perak, Malaysia for supporting this study.

REFERENCES

1. Adorni, G., Piatti, A., Bumbacher, E., Negrini, L., Mondada, F., Assaf, D., ... & Gambardella, L. M. (2025). FADE-CTP: A Framework for the Analysis and Design of Educational Computational Thinking Problems. *Technology, Knowledge and Learning*, *30*(2), 1073-1148. https://arxiv.org/abs/2403.19475
2. Arakawa, K., Hao, Q., Greer, T., Ding, L., Hundhausen, C. D., & Peterson, A. (2021). In situ identification of student self-regulated learning struggles in programming assignments. In *Proceedings of the 52nd ACM technical symposium on computer science education* (pp. 467-473).
3. Bocconi, S., Chioccariello, A., Kampylis, P., Dagienė, V., Wastiau, P., Engelhardt, K., Earp, J., Horvath, M.A., Jasutė, E., Malagoli, C., Masiulionytė-Dagienė, V. & Stupurienė, G. (2022). *Reviewing Computational Thinking in Compulsory Education*. Publications Office of the European Union, Luxembourg. https://data.europa.eu/doi/10.2760/126955
4. Choi, W. C., & Choi, I. C. (2024). The Influence and Relationship between Computational Thinking, Learning Motivation, Attitude, and Achievement of Code. org in K-12 Programming Education. *arXiv preprint arXiv:2412.14180*. https://arxiv.org/abs/2412.14180
5. Curi, M. E., Gerosa, A., Viera, M., & Carboni, A. (2025). A review of computational thinking interventions in upper elementary education. *Computers and Education Open*, 100252.
6. Druga, S. (2025). Coding isn't dead, but how it's taught needs to change, says Google DeepMind research scientist. *Business Insider*. https://www.businessinsider.com/google-deepmind-research-scientist-coding-not-dead-ai-education-2025-5
7. Kitsantas, A., Bembenutty, H., Cleary, T. J., Alexander, P. A., DiBenedetto, M. K., Kaplan, A., ... & Schunk, D. H. (2025). Barry J. Zimmerman's Enduring Legacy: The Inspiring Fusion of Self-Regulated Learning Theory, Practice, and Mentorship. *Educational Psychology Review*, *37*(3), 78.
8. Kong, S. C., & Liu, B. (2023). Supporting the self-regulated learning of primary school students with a performance-based assessment platform for programming education. *Journal of Educational Computing Research*, *61*(5), 977-1007.
9. https://journals.sagepub.com/eprint/FSCBDFVJJQVHF6JC8JNS/full
10. Kong, S. C., & Wang, Y. Q. (2024). Dynamic interplays between self-regulated learning and computational thinking in primary school students through animations and worksheets. *Computers & Education*, *220*, 105126.
11. Korte, S. M., Körkkö, M., & Føreland, L. R. (2025). Developing computational thinking skills in higher education through peer reflection on robotics and programming exercises with Bee-Bots, Lego Mindstorms EV3 and Minecraft Education. *Learning, Culture and Social Interaction*, *55*, 100947.
12. Li, H., & Ma, B. (2025). Design of AI-Powered Tool for Self-Regulation Support in Programming Education. *arXiv preprint arXiv:2504.03068*.
13. Li, Q., Jiang, Q., Liang, J. C., Xiong, W., & Zhao, W. (2025). Roles of programming self-efficacy, cognitive styles, and self-regulated learning strategies on computational thinking in computer programming. *Humanities and Social Sciences Communications*, *12*(1), 1-12.
14. Liu, Z., Gearty, Z., Richard, E., et al. (2024). Bringing computational thinking into classrooms: A systematic review on supporting teachers in integrating computational thinking into K-12 classrooms. *International Journal of STEM Education, 11*, 51. https://doi.org/10.1186/s40594-024-00510-6.
15. Merino-Armero, J. M., González-Calero, J. A., & Cozar-Gutierrez, R. (2022). Computational thinking in K-12 education. An insight through meta-analysis. *Journal of Research on Technology in Education*, *54*(3), 410-437. https://doi.org/10.1080/15391523.2020.1870250.

16. Mills, K. A., Cope, J., Scholes, L., & Rowe, L. (2025). Coding and computational thinking across the curriculum: A review of educational outcomes. *Review of Educational Research*, *95*(3), 581-618. https://doi.org/10.3102/00346543241241327.
17. Page, M. J., McKenzie, J. E., Bossuyt, P. M., Boutron, I., Hoffmann, T. C., Mulrow, C. D., ... & Moher, D. (2021). The PRISMA 2020 statement: an updated guideline for reporting systematic reviews. *bmj*, *372*. https://doi.org/10.1136/bmj.n71.
18. Rao, T. S. S., & Bhagat, K. K. (2024). Computational thinking for the digital age: a systematic review of tools, pedagogical strategies, and assessment practices. *Educational technology research and development*, *72*(4), 1893-1924.
19. Robles Mucho, J. L., Andrade-Girón, D. C., Benites-Tirado, V. R., Quispe-Maquera, N. B., Ayala-Jara, C., Tito Chura, H. E., ... & Barboza, J. J. (2026). Open learner models and pedagogical strategies in higher education: a meta-synthesis of approaches to self-regulated learning. In *Frontiers in Education,* 10:1760183. doi: 10.3389/feduc.2025.1760183
20. Romero, M. (2024). Lifelong learning challenges in the era of artificial intelligence: A computational thinking perspective. *arXiv preprint arXiv:2405.19837*. https://arxiv.org/abs/2405.19837
21. SOBRAL, S. R. (2021). Bloom's Taxonomy to IMPROVE TEACHING-LEARNING IN INTRODUCTION TO PROGRAmming. *International Journal of Information and Education Technology, 11*(3),148-153. HTTPS://DOI.ORG/10.18178/ijiet.2021.11.3.1504
22. Sunday, A. O., Agbo, F. J., & Suhonen, J. (2025). Co-design pedagogy for computational thinking education in K-12: A systematic literature review. *Technology, knowledge and learning*, *30*(1), 63-118. https://link.springer.com/article/10.1007/s10758-024-09765-y
23. Tareq, Z. A., & Yusof, R. J. R. (2024). Modeling a problem-solving approach through computational thinking for teaching programming. *IEEE Transactions on Education*, *67*(2), 282-291.
24. Tariq, R., Aponte Babines, B. M., Ramirez, J., Alvarez-Icaza, I., & Naseer, F. (2025). Computational thinking in STEM education: current state-of-the-art and future research directions. *Frontiers in Computer Science*, *6*, 1480404. https://doi.org/10.3389/fcomp.2024.1480404
25. Ubaidullah, N. H., Mohamed, Z., Hamid, J., Sulaiman, S., & Yussof, R. L. (2021). Improving novice students' computational thinking skills by problem-solving and metacognitive techniques. *International Journal of Learning, Teaching and Educational Research*, *20*(6), 88-108.
26. van der Lubbe, L. M., van Borkulo, S. P., & Jeuring, J. T. (2025). A Systematic Literature Review of Computer Science MOOCs for K-12 education. *arXiv preprint arXiv:2501.18986*. https://arxiv.org/abs/2501.18986
27. Voon, X. P., Wong, S. L., & Wong, L. H. (2022). Developing computational thinking competencies through constructivist argumentation learning: A problem-solving perspective. *International Journal of Information and Education Technology, 12*(6), 529-539. https://doi.org/10.18178/ijiet.2022.12.6.1650
28. Watanabe, K., Matsuda, Y., Nakamura, Y., Arakawa, Y., & Ishimaru, S. (2025). TrackThinkDashboard: Understanding Student Self-Regulated Learning in Programming Study. *International Journal of Activity and Behavior Computing*, *2025*(1), 1-17. https://doi.org/10.48550/arXiv.2503.19460
29. Yang, K., Liu, X., & Chen, G. (2020). The influence of robots on students' computational thinking: A literature review. *International Journal of Information and Education Technology*, *10*(8), 627-631.
30. Zhang, X., Aivaloglou, F., & Specht, M. (2024). A systematic umbrella review on computational thinking assessment in higher education. *European Journal of STEM Education, 9*(1), 02. https://doi.org/10.20897/ejsteme/14175.