Computational Thinking Pedagogical Framework + for Early Learners

Safia A. Malallah, Kansas State University, safia@ksu.edu
Lior Shamir, Kansas State University, lshamir@ksu.edu
William Henry Hsu, Kansas State University, bhsu@ksu.edu
Joshua Levi Weese, Kansas State University, weeser@ksu.edu
Salah Alfailakawi, Kansas State University, Bosloh@ksu.edu

Abstract: Pedagogy provides a solid foundation for educators to design effective teaching and learning experiences. However, very few resources address computational thinking (CT) pedagogical experiences for that prepare early learners to become problem solvers in the computer science and engineering domains, skills that are necessary to meet future industry requirements. To address this gap, this paper proposes a framework and models to help educators identify available CT experiences to incorporate them into their lessons. The framework includes nine pedagogical experiences: (1) Unplugged, (2) Tinkering, (3) Making, (4) Remixing, (5) Robotics+, (6) Engineering, (7) Coding, (8) Dataying, and (9) Artificial Intelligence (AI).

Introduction

The growth of computational careers worldwide means that students of all ages, including children in early childhood, must be consistently exposed to various problem-solving approaches and be creators, not consumers, of technology [1] to meet future industry requirements [2]. More meaningful ways exist to train students' computational thinking (CT) abilities as the computer science (CS) domain keeps evolving and generating new important buzzwords, such as artificial intelligence (AI) and data science [3]. However, educators who lack CS backgrounds often not aware of the available educational CT experiences, and the current resources are very limited. Although appropriate CT selection can enhances educational goals because it enables increased comprehension of new concepts in various disciplines, including language, math, music, and art, educators must be aware of how CT can be utilized to determine appropriate applications for K–12 students [4].

This paper outlines the Computational Thinking Pedagogical Framework Plus (CTPF+) to present various CT pedagogical experiences suitable for early childhood development. The framework includes nine pedagogical experiences that cognitively train CT skills: (1) Unplugged, (2) Tinkering, (3) Making, (4) Remixing, (5) Robotics +, (6) Engineering, (7) Coding, (8) Dataying, and (9) AI. This work also proposes the Foundation-to-Creation model to complement the framework and holistically justify the needed foundation, as well as the CT-HOT thinking process to help educators form CT-related questions. By utilizing the proposed framework and models, educators can offer their students meaningful, diverse CT learning experiences

Background

Child's Developmental Milestones and Computational Thinking

Successful CT growth primarily correlates with a child's developmental milestones since CT activities require unique cognitive skills. For example, video games require understanding of analogies, processing speed, and deductive reasoning [5], while online search engine usage requires recall memory, spelling, and Boolean logic [6]. Typing on a keyboard requires motor skills, visual skills, and cognitive ability [7], while technological communication requires speech and language skills [8]. Fine motor skills are necessary to control technology [9], while gross motor skills and whole-body interaction can improve somatosensory experience [10]. In

addition, social-emotional skills impact repeated trial-and-error activities, leading either to a user's frustration and failure or enhanced group work and reliable communication [11] [12].

Positive Technological Development

Positive Technological Development (PTD) is a theory-based framework considered one of the best to promote positive behaviors around children, presented in Figure 1. It integrates technology to maximize its positive impact on human flourishing and well-being. It is comprised of three levels with standards for positive outcomes, called assets. This level includes the six Cs: caring, connection, contribution, competence, confidence, and character. Another level is behaviors, whose purpose is to raise the positive behavior around technology. This level has six small Cs: communication, collaboration, community building, content creation, creativity, and choice of conduct. The third level is classroom practice that involves designing and implementing educational experiences that align with the principles of the first two levels beside early childhood principles [1].

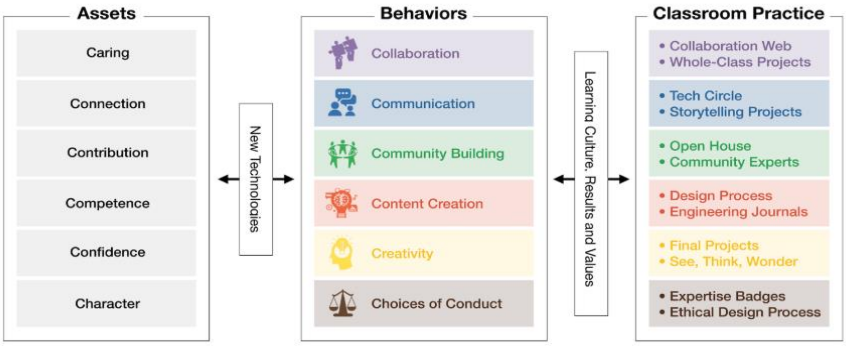


Figure 1. PTD [1]

Higher-Order-Thinking

Higher Order Thinking (HOT) is an approach that goes beyond simple recall or understanding of information by asking questions that involve analyzing, synthesizing, and evaluating information to reach a conclusion or decision. The concept of HOT has a long history, dating back to the work of educational philosophers such as John Dewey and Jean Piaget in the early 20th century [13].

Method

Design

A conceptual framework is a structured approach to organizing and understanding complex ideas using a visual representation to define topic elements and their relationships [14]. This research combined the conceptual framework approach from methods in previous research [15] [16] to develop the CTPF+ framework. The following six stages were used to develop the framework:

Steps 1
Literature

Step 2
Find CT Elements

Step 3
Find CT Pedagogical Experience

Steps 4-5
Connect CT Elements & CT Experience

Step 6
Developing CTPF+

Figure 2. Design Steps for the Dataying Framework

1. ***Literature Reviews***: Literature reviews start with researching the database and identifying relevant studies, screening them to remove duplicates; eliminating the ones that do not match search criteria; and checking that they target early childhood. Starting with recent literature reviews, we examined the reference section and included items that contain pedagogy experiences or tools. The purpose of this step was to understand the status of CT in early childhood.
2. ***Identify Early Childhood Computational Thinking*:** This step examines literature reviews and identify CT concepts that prove it is suitable for early childhood. This stage included clustering, grouping, and eliminating redundant concepts.
3. ***Identify Pedagogical Experience with Computational Thinking***: This step aimed to identify a pedagogical experience used to train CT concepts from literature reviews.
4. ***Connection of the Elements***: This stage identified relationships between CT concepts and pedagogical experiences. Additionally, it examined their binding.
5. **Fill Missing Gaps**: This step identified needed knowledge to enhance utilization of CTPF+. Several models were constructed to fill the missing gaps
6. ***Developing the Framework***: This final step and it put all the elements together.

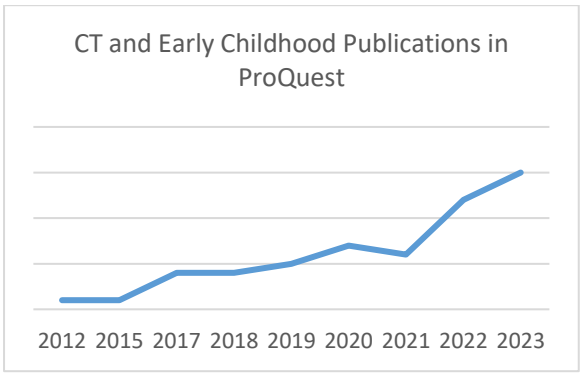
Keywords, Database, and Criteria

Searching was performed twice for this research. The first round searched the keywords "early childhood || preschooler" + "Computational Thinking" + "review." The exclusion criteria were if the research was not for children ages 4 to 7 or if the paper subject is not literature reviews published in the past 3 years. The results were filtered by reviewing their abstracts and titles. The second round searched the same keywords as the first, except it omitted "review." The exclusion criteria were the same as for round 1 along with papers that did not include pedagogy experiences or CT concepts for those ages 4 to 7. The results were filtered by looking at their abstracts, titles, and content. The sources used are English conference papers and proceedings; government and official websites; and scholarly journals. ProQuest was the search engine used for both rounds.

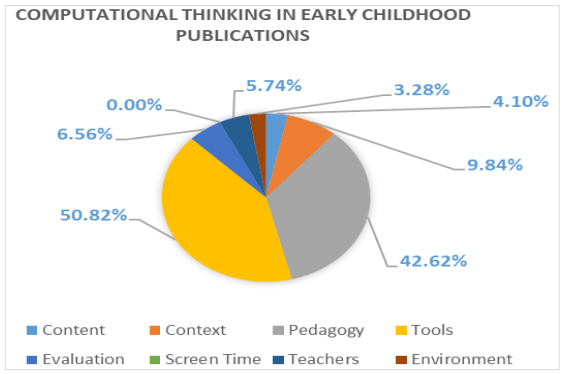
Results

Step 1: Literature Reviews

Analysis of the literature reviews resulted in 142 papers. The original search identified 65 documents; the rest were discovered by reviewing the references of the original 65. Figure 3.a presents chronological order of the papers that met our search criteria. Although we did not specify any time frames, we did not come across any results before 2012. MAXQDA tool was used to analyze the literature reviews. Figure 3.b. illustrates the percentages of the number of papers categorized by their main focus.



a. CT and Early Childhood Publications Per Year



b. CT and Early Childhood Publications Per Category

Figure 3. Summary of the Literature Reviews

Four main papers investigated literature reviews about CT in early childhood. First, Yue Zeng et al. analyzed 42 studies based on the framework developed by Brennan and Resnick. They identified CT as concepts (i.e., control flow/structures, representation, and hardware/software); as a practice (i.e., algorithmic design, pattern recognition, abstraction, debugging, decomposition, iteration, and generalizing); and as a perspective (i.e., expressing and creating, connecting, perseverance, and choices of conduct) [17].

Second, Jiahong Su et al. also performed a systematic review of studies from 2010 to 2022, examining 26 research studied that analyzed CT integration into early childhood through various approaches such as programming and coding activities; robotics and engineering; game design; and storytelling. This paper highlights the common terms associated with CT, concepts, and skills that have been mentioned multiple times in the selected studies. These terms include "sequencing, conditionals/control structures, iterations/loops, testing and debugging, pattern recognition, algorithms, modularity representation, and problem-solving" [18].

The third paper by Zhang et al. was a systematic review of learning CT using Scratch in kindergarten through ninth grade. It utilized a framework developed by Brennan and Resnick to assess the effectiveness of different pedagogical approaches to CT education through Scratch. They identified that Brennan and Resnick's framework can be delivered to this age range. Furthermore, Zhang identified additional CT skills: input/output; code reading; interpretation and communication; utilization of multimodal media; predictive thinking; and human-computer interaction. Kindergarten through ninth grade is a wide age group; it is not clear which work for kindergarten through second grade [19].

The fourth paper was published by the Committee on Enhancing Science and Engineering in Prekindergarten through Fifth Grade. It supported developing an appropriate technique to include CT for young children. The integration is often achieved through the STEM-based engineering design (unplugged with a sheet or plugged into tangible technology); open-ended applications such as ScratchJr; level-up applications such as Code.org curriculum; plugged activities; robotics; unplugged activities; and using C as context to teach other subjects like art and music. The committee also contributed suggestions to use CT as a tool for teaching students through fifth grade by presenting multiple representations of it. The representations should include a range of examples, such as unplugged activities, real-life scenarios, everyday language, pseudocode, flowcharts, code tracing/tracking charts and tables, and coding languages. The idea is to start with more tangible concepts and gradually progress to more abstract ones, ensuring that the representations are developmentally appropriate approximations of the coding processes [20].

Yadav et al. focus on the incorporation of CT into K–12 education. The authors review various pedagogical approaches for teaching CT, including coding activities, game design, and robotics. They argued that CT should be integrated into the existing curriculum rather than taught as a standalone subject and provided examples of how this can be done across multiple subject areas [21]. Also, Rehmat et al. focused on exploring effective instructional strategies for teaching young learners CT. The authors highlighted the importance of developing CT skills in early education and provided an overview of key CT concepts and skills. It was suggested to use questioning and modeling techniques to aid students in understanding the robot's movements and associated CT competencies [22]. Additionally, Saxena et al. did an exploratory study investigating the effectiveness of unplugged and plugged activities in promoting CT skills in early childhood education. The authors developed CT activities that are both unplugged (offline) and plugged (using technology) and evaluated their effectiveness in promoting CT skills among young learners. The CT skills studied were sequencing, algorithm, procedure, and pattern recognition. The study found that both types of activities are effective in promoting CT skills, but unplugged activities are more effective in promoting problem-solving and logical

thinking skills. The authors concluded that combining unplugged and plugged activities can successfully promote CT in early childhood education [23]. Hodges et al. established purposeful analogous representations of coding processes to evaluate the suitability of these representations and their translation in a study of second grade students using a computing device [24]. Aggarwal et al. compared physical manipulatives to develop code within the Kodu curriculum to students who did not use the manipulatives. The students were divided into two groups. One used flashcards and tiles to enter code into the Kodu Game Lab. The other used just paper and pencil representations before entering code into the Kodu Game Lab. The authors suggested that manipulatives may have diminishing returns and should be scaffolded carefully [25].

Certain areas of focus were not covered by the literature reviews and are important for this paper. Morgan et al. discussed using engineering to inspire and develop CT skills in young children and create the (STEM + C) model. The study involved 85 preschool children who participated in a series of design activities involving problem-solving, collaboration, and simple coding tools. The results showed that the activities effectively promoted CT skills such as decomposition, pattern recognition, and abstraction. The proposed CT in their work is data collection, data analysis, data representation, problem decomposition, abstraction, algorithms and procedures, automation, simulation, and parallelization [26]. Also important is the AHA! Island project. The educational program focuses on training CT concepts using engineering through a series of cartoons and hands-on activities to reach low-income families and their children ages 4 to 5. It is supported by the National Science Foundation. The program is designed to be engaging and interactive by teaching children about concepts such as abstraction, algorithmic thinking, pattern recognition, problem deconstruction, design process, debugging process, and logical reasoning [27].

Another important area of focus is Integrated-CT-organization developed pathways to include CT in grades K–12 through other non-STEM disciplines. It includes some data science elements such as data practice and data analysis [20]. Donna Kotsopoulos proposed that "unplugged, tinkering, making, and remixing are effective pedagogical experiences to train CT for young children in her Framework (CTPF)" [28]. NAEYC considered "making, tinkering, and engineering as three important overlapping concepts to teach STEM" [29]. Williams et al. designed an early childhood AI curriculum that depends on building, training, and programming a social robot. Williams delivered three AI concepts—knowledge-based systems, supervised machine learning, and generative—through introducing the HOT technique. Table 1 presents HOT questions using three systems: rock-paper-scissors for KBS, food classification for the SML, and music remix for generative AI systems [30].

Table 1. HOT Questions Using Different Systems

Knowledge-Based Systems	Supervised Machine Learning	Generative Music
Rock-Paper-Scissors activity	Healthy and Unhealthy food game	Remixing music
1. We teach the robot the normal rules. Then, Sally plays rock and the robot plays paper. Who does the robot think has won? Sally or the robot? (Robot) 2. Sally plays paper five times. What does the robot think she will play next? Rock, paper, or scissors? (Paper) 3. The robot thinks that Sally will play paper next. What will the robot play so that it can beat Sally? Rock, paper, or scissors? (Scissors) 4. We changed the rules so that they are all opposite rules (paper beats scissors). Sally plays scissors and the robot plays paper. Who does the robot think has won? Sally or the robot? (Robot)	1. The robot does not know anything about foods. You put strawberries and tomatoes into the good group. Which group will the robot think chocolate goes? The good group or the bad group? (Good) 2. What food does the robot think is most like a tomato? Strawberry, banana, or milk? (Strawberry) 3. You put ice cream in the good category and bananas in the bad category. What category will the robot put corn? The good category or the bad category? (Bad)Generative Music (GM)	1. Priya asks the robot to play back with the bars in the middle. Does the robot play the same song or a different song? (Same) 2. Priya asks the robot to play back with the bars to the right. Does the robot play the same song or a different song? (Different) 3. Does the robot's song have to have some of the same notes as the input? (Yes)

Step2: Computational Thinking Elements for Early Childhood

Based on the literature review, Table 2 presents the 36 identified CT elements (i.e., concepts, practice, perspective) suitable for children ages 4–7 years old. Except for skills 1, 2, 7, 8, 34–36, which were derived from the literature reviews in round 2, the rest of the CT skills were identified from the four literature reviews in round 1.

Table 2. Suitable CT Elements

	Concepts/Practices/ perspectives	Resources		Concepts/ Practices/ perspectives	Resources
1	Logical Thinking and reasoning	(AHA! Island project)	19	Predictive thinking	(Zhang et al)
2	problem deconstructions	(AHA! Island project)	20	Code communication (read, understand, communicate)	(Zhang et al)
3	Control flow/ control structures	(Bers et al	21	connecting	(Zhang et al)
4	make decision	(Bers et al)	22	Expressing	(Zhang et al)
5	scientific methods	(Bers et al)	23	series of actions logically	(Pugnali et al). (García-Valc et al). (Relkin et al). (Gerosa et al.)
6	Design process	(Bers et al). (AHA! Island project)	24	sequencing	(Pugnali et al). (Kazakoff et al). (Bers et al). (Pugnali et al). (García-Valc et al). (Relkin et al). (Saxena et al.). (Gerosa et al.)
7	parallelism	(STEM + C)	25	Debugging and troubleshooting	(Rehmat rt al). (Bers et al). (Pugnali et al). (García-Valc et al). (Relkin et al). (Gerosa et al.). (Bers et al). (AHA! Island project). (Yadav et al.). (Zhang et al)
8	automation	(STEM + C)	26	Decomposition, Modularity.	(Rehmat rt al). (Relkin et al). (Relkin et al). (Bers et al). (STEM + C). (Rich et al.). (Yadav et al.).
9	simulation	(STEM + C). (Lee et al)	27	Abstraction	(Rehmat rt al). (AHA! Island project). (STEM + C). (Lee et al). (Yadav et al.)
10	spatial concepts	(Gerosa et al.)	28	Algorithm and procedure	(Rehmat rt al). (Saxena et al.). (Bers et al). (AHA! Island project). (STEM + C). (Lee et al). (Zhang et al)
11	questions	(Rehmat rt al). (Zhang et al)	29	Pattern recognition	(Rehmat rt al). (Saxena et al.). (Relkin et al). (AHA! Island project). (Lee et al). (Yadav et al.).
12	Project representation	(Relkin et al).	30	sorting	(Papadakis et al.). (Bers et al). (Lee et al)
13	cause and effect	(Relkin et al).	31	conditionals	(Pugnali et al). (Bers et al)
14	enhancement	(Relkin et al).	32	loop/repeats	(Pugnali et al). (Pugnali et al)
15	Identifying problems	(Relkin et al).	33	hardware/ software	(Relkin et al). (Gerosa et al.). (Bers et al). (AHA! Island project). (Bers et al 2).
16	Event	(Zhang et al)	34	Data collection	(STEM + C).
17	Variable	(Zhang et al)	35	Data analysis	(STEM + C).
18	being iterative & incremental	(Zhang et al)	36	data representation	(STEM + C). (Bers et al 2).

During analysis, similar elements, such as decomposition and modularity, were merged, while other elements, such as connection and expiration, were removed. The elements were then clustered by their nature into five groups: building solutions process, solution types, technology literacy, foundation for CT, and CT skills (Figure 4).

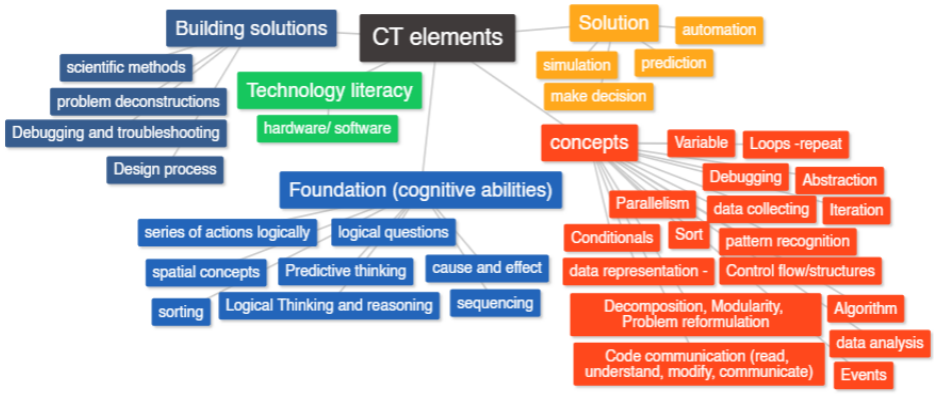


Figure 4 . Clustering of CT Elements

Step3: Computational Thinking Pedagogical Experiences

The results of the previous steps identified two ways to teach with CT (CT as a context and CT as a content), in addition to 10 pedagogical experiences: unplugged, tinkering, making, mixing, coding, engineering, data science, AI, robotics, and devices. Devices and robotics carry the same natures—hardware —so their handling should be similar. Therefore, they were grouped into robotics+ to include any computing devices that consist of hardware and software to carry out user tasks.

Several relationships can be identified between the elements. One relationship can be based on their ability to build a solution. Some problems (i.e., engineering) require creating physical solutions such as building a bridge using macaroni and marshmallows. Another type creates digital solutions like creating an app that can count using ScratchJr. A third type uses a mix (i.e., robotics) to control a code that makes a robot dance. A fourth type (i.e., data science)

is based on collecting and analyzing data for insights and decision making. Even AI is part of this category. Its primary purpose is to create smart models to solve a problem. Yet, more tools are needed to allow children to build their open-ended solutions using AI. Thus, engineering, coding, data science, and AI elements are bound under one category: problem-solving.

This group of roles also can be labeled CT as content. The content itself is a direct use of the CT. This infers the category of CT as a context. CT is a medium to of attaining educational goals by enabling a deeper comprehension of new concepts in various disciplines—including language, math, music, art, and others—through the integration of CT principles. Each problem-solving element can be practiced using an unplugged activity such as a coding card, tinkering, exploring scratch, and "making" by using a recipe (using exact instructions) for baking a cake. This denotes the included relationships; everything can be done using almost everything. Thus, the framework would include them as overlapping.

Young students need to start with more concrete experiences and progress toward more abstract ones. Progress happens by transitioning between representations. Start with unplugging; then tinkering to see how tools work and their capabilities; making to develop a good quality product by following an adequate example; and then problem-solving to build their solutions with any roles they like. Remixing is the outlier as it requires grasping the skills to avoid being overwhelmed by the number of new challenges that might arise.

Progress does not always create solutions; it also can be the consumption of knowledge. For example, it could be information consumption apps that teach concepts, like Aha Island, or coding games that engage a child in level progress with pre-expected answers and no creativity, such as the code.org curriculum. The framework also considers them.

Step4: Connect Pedagogy Experiences and Computational Thinking

It is recommended to propose CT skills depending on the nature of the problem. In other words, which thinking hat does the child need to wear: engineer, coder, or data scientist? Creating coding projects requires full coding skills such as events and variables. The following CT sets are a recommendation depending on the roles. Note that skills can be combined.

- Coding and robotics: loops, events, conditionals, parallelism, control flow/structures, code communication (i.e., read, understand, modify, communicate), variable.
- Data science: data collecting, data representation, data analysis.
- The rest are recommended for all kinds of tasks, including making, tinkering, or a combination.

Step5: Fill Missing Gaps

Figure 5 presents CT Foundation-to-Creation model in a way that connect CT elements with pedagogical experience using the findings from the previous steps, children's developmental milestones, and PTD. The model has three levels: foundation, CT skills, and solutions. Level one works as a foundation for the model. For a child to be ready to build a solution using CT, the child must have skills in the necessary categories. Eight categories from the children's developmental milestones can be used with plugged and unplugged activities. For example, to use the sorting CT skills over materials, the logical thinking foundation is employed. Additionally, the child needs to understand the problem. So first, the child needs to understand the ability to know that the materials will be moved and the order will be changed. Then they must identify which relationships are between the elements. If it is a number, the child must realize they use a number qualitative relationship (i.e., bigger, smaller, and equal). If it is language and literacy, the child needs to understand letter logic (i.e., the order of the alphabet). These skills are logical ordering, grouping, and transferring to a domain.

The second level of the model is CT skills, which build from the foundation. For example, creating coding projects like a game requires coding the robotic skills. Because the

nature of the robot and CT are similar, a child can use loops and events variables. However, those are not part of the engineering of the robot. The same can be said with data science—creating data-driven solutions using skills to collect the data, represent the data, and find insight into the analysis. Making, tinkering, and mixed methods can be used for multiple other types of projects.

The third level of the model is creating solutions. Almost all roles and solution-building require understanding the problem; preparing and planning; building; checking the solutions; and sharing the solutions. The five proposed steps were recommended with the expectation that the foundation necessary for each has been attained. For example, understand the problem by breaking down complex problems into smaller parts, creating algorithms to solve the smaller problems, and planning to save time and resources. PTD frameworks should be integrated with these steps to maximize the positive impact while using CT to build solutions

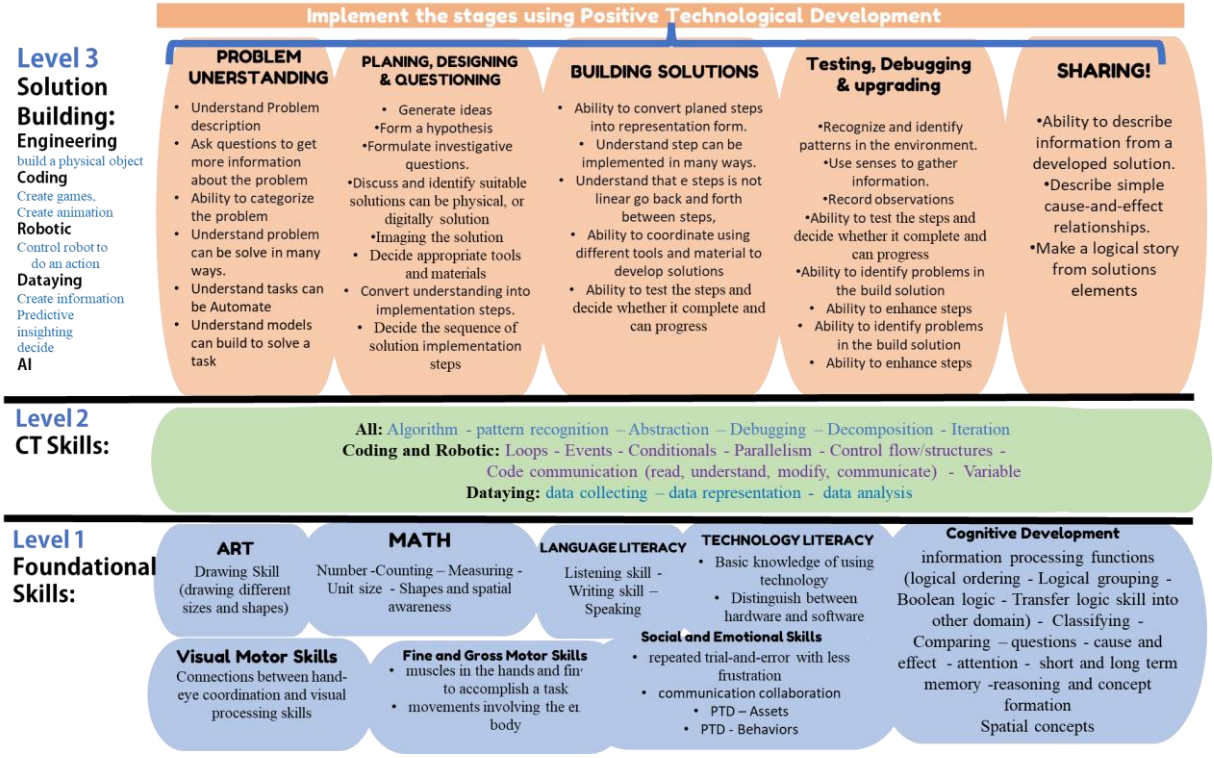


Figure 5. CT-Foundation-to-Creation Model

Early childhood educators play an essential role in enhancing the CT experience and increasing student understanding of CT. Teachers need to determine HOT question types through solution building stage (Figure 6). An example of a question for the thinking step could be "I wonder what would happen if…?" An example of a testing question could be "Can you show me how to use it...?," and a question for the self-reflection step could be "What was the most interesting thing you learned here?" An example question for the improvement step could be "What might you do differently next time?"

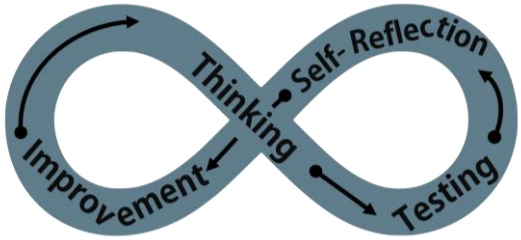


Figure 6. CT-Thinking Process

Step 6: Developing the Computational Thinking Pedagogical Framework Plus (CTPF+)

The CTPF+ has nine main pedagogical experiences that can be implemented using CT as a context or content. Together with the CT Foundation-to-Creation model, they can be used as a holistic picture to choose the type of CT experience and its expected solutions. This gives educators and the child a clear option of the available choices. As shown in Figure 7, the hierarchy constructed to allow for expansion to add new computational experiences in the problem-solving circle after confirming its suitability for early childhood education (e.g., coding was not suitable until ScratchJr and block programming were developed).

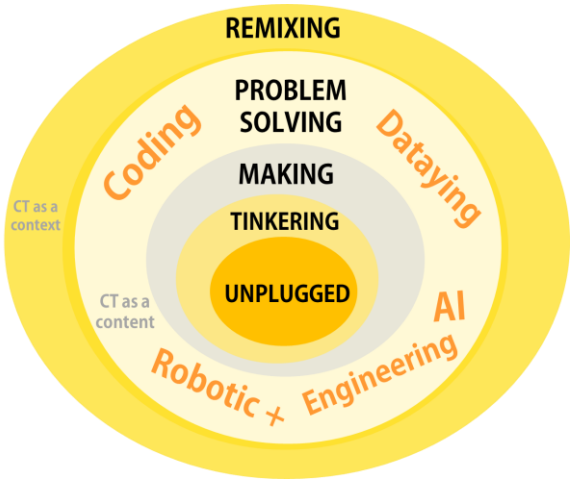


Figure 7. CTPF+ Framework

Engineering in early childhood is using materials to construct a physical object to solve a problem or fulfill a need or desire. The integration of CT in engineering design can help build more systematic and efficient engineering solutions. Several curricula can be found to train CT using engineering. CT can be used in many different stages of the engineering design process, including problem identification, ideation, solution implementation, and evaluation.

Robotic+ uses robotics and electronic tools with codes to control the robot to solve a problem or fulfill a need or desire. Teaching CT using Robotic+ has become an increasingly popular method to teach STEM to young children, particularly because it does not always require a screen. CT skills are the combination of CT in coding and engineering.

Artificial intelligence in early childhood makes a machine perform tasks—such as decision-making—by learning from humans. There are limited AI tools and curriculum research, so we cannot generalize how AI can be used with CT for early childhood. But referring to Williams's work, students can learn algorithm, pattern, and abstraction concepts within the three key AI concepts.

Coding is the process of creating instructions that a computer can understand and execute specific to a task or application being developed. Children can learn CT through coding activities, where CT is originally derived from the coding computer science field. Figure 8 shows coding development stages to train to code. The first stage is the pre-operational stage, which focuses on coding thinking skills using unplugged activities such as exact instruction games. The second stage is the concrete implementation stage, which emphasizes operational skills for coding and robotics to build projects using progressive coding language. It starts with tangible for 3-year-olds, moves to graphical coding for 4-year-olds, and progresses to block and hybrids. The final stage is the abstract stage, which requires advanced abstract thinking skills and the ability to replace concrete ideas.

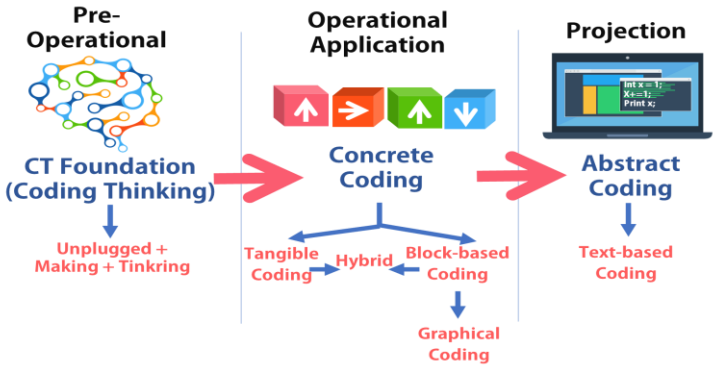


Figure 8. CT Coding Development Stages

Dataying is an original term intended to be a buzzword for early childhood data science. It is the process of solving a simple data-decision problem by collecting a small data sample and then analyzing it for insights to make a decision in a way that is suitable for early childhood education. Currently, no customized tools exist to support dataying for early childhood. This can be used using the unplugged activities. Figure 9 outlines six stages to introduce the foundation of data science thinking for children who have never been exposed to data-driven solutions [31].

Figure 9. Dataying Framework [31]

Unplugged with Computational Thinking activities refer to educational activities that do not require a screen but rather use physical materials that challenge children to think computationally. Unplugged activities can be done both with and without technology. Activities without technology include storytelling, worksheets, card games, and real-life tasks. Activities with technology use tangible tools and include computer science picture books, watching cartoons, graphic novels, and tangible coding, where physical objects are used to represent commands in a sequence, teaching children how sequences of instructions can be used to train CT.

Computational Thinking Tinkering is the act of playing with and exploring materials through trial and error to discover how things work and how to create things from them. By experimenting, individuals can better understand their capabilities, a fundamental skill for building solutions. Often, learning happens unintentionally as children put materials together, test what happens, and try again. CT tinkering can be applied to a wide range of challenges, from fixing a ScratchJr code to placing a listening sensor on a KIBO robot.

Making is hands-on experimentation going through steps to build a project that does not require solving a problem. It is done because it is fun and includes tasks like constructing a desk from reading the instructions without having a plan for what they will do with the desk. An example of CT making is decomposition skills to understand the needed material to make this object.

Pedagogical remixing is the process of taking pieces of the pedagogical experience and using them together to solve a problem. The parts can be combined, modified, or transformed to create something new. Children should be encouraged to use materials in ways they might not have thought of before. For example, they could use blocks, robots, and scratch apps to build animals that move and make a sound.

CTPF+ Integration

The CTPF+ can be integrated into the early childhood curriculum by complementing the "classroom practices" of the PTD framework - early childhood pedagogy and principles as a teaching method using age-appropriate techniques. To begin, teachers should observe the lesson goals and then select a CT pedagogical experience suitable or preferable for the lessons from CTPF+. Next, identify CT concepts relevant to the pedagogical experience from level-2 in the CT-Foundation-to-Creation Model. After identifying the relevant CT concepts, teachers can identify the needed foundation for their students from level-1. This will help ensure the students have the necessary knowledge to understand and apply the CT concepts and skills. Lastly, redesign the lesson using the CT activities to deliver the lesson outcomes. For example, suppose an engineering lesson is being taught on building a bridge, and the students need more foundation over the comparing skill. In that case, the lesson implementation should focus on the comparison, such as which materials are stronger, by doing testing and experiments on strengths and weaknesses. The students will use pattern recognition, abstraction, and debugging to train the foundation.

If technology is chosen as a medium for teaching CT, it is recommended that teachers be aware of the best practices for using technology around children. This will help ensure that the technology used is safe, age-appropriate, and aligns with the curriculum's learning goals [32].

Limitations and Future Work

The CTPF+ frameworks based on the systematic review collected from ProQuest. Therefore, works that can provide different insight into this research may have been missed. Also, most literature reviews build their work on Brennan and Resnick, which can lead to bias as it influences all the author’s views. Other limitations are the limited work for data science, and AI infers the need to have more research to influence the judgments, and the inclusion of CT for early childhood. As a future work, the models and framework developed could be branched into several qualitative research studies for validation. Additionally, AI inclusion for early childhood learning could be studied.

Acknowledgements

This work was funded by the National Science Foundation (NSF) with Grant No DRL GEGI008182. However, the authors alone are responsible for the opinions expressed in this work and do not reflect the views of the NSF.

References

[1] A. Strawhacker and M. U. Bers, "Promoting positive technological development in a Kindergarten makerspace: A qualitative case study," *European Journal of STEM Education,* vol. 3, no. 3, p. 9, 2018.

[2] B. Vittrup, S. Snider, K. K. Rose, and J. Rippy, "Parental perceptions of the role of media and technology in their young children's lives," *Journal of Early Childhood Research*, vol. 14, no. 1, pp. 43-54, 2016.
[3] A. Smith and J. Anderson, "AI, Robotics, and the Future of Jobs," *Pew Research Center*, vol. 6, p. 51, 2014.
[4] L. Caparrotta, "Digital technology is here to stay and the psychoanalytic community should grapple with it," *Psychoanalytic Psychotherapy*, vol. 27, no. 4, pp. 296-305, 2013.
[5] A. Hisam, S. F. Mashhadi, M. Faheem, M. Sohail, B. Ikhlaq, and I. Iqbal, "Does playing video games effect cognitive abilities in Pakistani children?," *Pakistan journal of medical sciences*, vol. 34, no. 6, p. 1507, 2018.
[6] H. Hutchinson, A. Druin, B. B. Bederson, K. Reuter, A. Rose, and A. C. Weeks, "How do I find blue books about dogs? The errors and frustrations of young digital library users," *Proceedings of HCII 2005*, pp. 22-27, 2005.
[7] Indigo. "Unlocking Abilities: Keys to Developing Touchscreen Skills." https://www.indigosolutions.org.au/docs/default-source/unlocking-abilities/touchscreen-resources/unlocking-abilities-keys-to-developing-touchscreen-skills.pdf?sfvrsn=b28a3ef5_8 (accessed 2022).
[8] J. H. Danovitch, "Growing up with Google: How children's understanding and use of internet-based devices relates to cognitive development," *Human Behavior and Emerging Technologies*, vol. 1, no. 2, pp. 81-90, 2019.
[9] K. E. Wohlwend, "One screen, many fingers: Young children's collaborative literacy play with digital puppetry apps and touchscreen technologies," *Theory Into Practice*, vol. 54, no. 2, pp. 154-162, 2015.
[10] Z. Ren and J. Wu, "The effect of virtual reality games on the gross motor skills of children with cerebral palsy: A meta-analysis of randomized controlled trials," *International journal of environmental research and public health*, vol. 16, no. 20, p. 3885, 2019.
[11] M. Toeters, M. ten Bhömer, E. Bottenberg, O. Tomico, and G. Brinks, "Research through design: a way to drive innovative solutions in the field of smart textiles," in *Advances in Science and Technology*, 2013, vol. 80: Trans Tech Publ, pp. 112-117.
[12] Z.-J. Zhong, "The effects of collective MMORPG (Massively Multiplayer Online Role-Playing Games) play on gamers' online and offline social capital," *Computers in human behavior*, vol. 27, no. 6, pp. 2352-2363, 2011.
[13] M. H. Hopson, R. L. Simms, and G. A. Knezek, "Using a technology-enriched environment to improve higher-order thinking skills," *Journal of Research on Technology in education*, vol. 34, no. 2, pp. 109-119, 2001.
[14] S. Bas and G. Tegan. "What Is a Conceptual Framework? | Tips & Examples." https://www.scribbr.com/methodology/conceptual-framework/ (accessed.
[15] C. Carroll, M. Patterson, S. Wood, A. Booth, J. Rick, and S. Balain, "A conceptual framework for implementation fidelity," *Implementation science*, vol. 2, pp. 1-9, 2007.
[16] S. A. Malallah, "Developing computational thinking best practices for early childhood education in Kuwait and United States," Kansas State University, 2022.
[17] Y. Zeng, W. Yang, and A. Bautista, "Computational thinking in early childhood education: Reviewing the literature and redeveloping the three-dimensional framework," *Educational Research Review*, p. 100520, 2023.
[18] J. Su and W. Yang, "A Systematic Review of Integrating Computational Thinking in Early Childhood Education," *Computers and Education Open*, p. 100122, 2023.
[19] L. Zhang and J. Nouri, "A systematic review of learning computational thinking through Scratch in K-9," *Computers & Education*, vol. 141, p. 103607, 2019.

[20] Tamara J. Moore, "The Integration of Computational Thinking in Early Childhood and Elementary Education," ed: Committee on Enhancing Science and Engineering in Prekindergarten through Fifth Grade, 2020.
[21] A. Yadav, H. Hong, and C. Stephenson, "Computational thinking for all: Pedagogical approaches to embedding 21st century problem solving in K-12 classrooms," *TechTrends*, vol. 60, pp. 565-568, 2016.
[22] A. P. Rehmat, H. Ehsan, and M. E. Cardella, "Instructional strategies to promote computational thinking for young learners," *Journal of Digital Learning in Teacher Education*, vol. 36, no. 1, pp. 46-62, 2020.
[23] A. Saxena, C. K. Lo, K. F. Hew, and G. K. W. Wong, "Designing unplugged and plugged activities to cultivate computational thinking: An exploratory study in early childhood education," *The Asia-Pacific Education Researcher*, vol. 29, no. 1, pp. 55-66, 2020.
[24] C. B. Hodges, S. Moore, B. B. Lockee, T. Trust, and M. A. Bond, "The difference between emergency remote teaching and online learning," 2020.
[25] A. Aggarwal, C. Gardner-McCune, and D. S. Touretzky, "Evaluating the effect of using physical manipulatives to foster computational thinking in elementary school," in *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, 2017, pp. 9-14.
[26] M. M. Hynes *et al.*, "Inspiring Computational Thinking in Young Children's Engineering Design Activities (Fundamental)," in *2016 ASEE Annual Conference & Exposition*, 2016.
[27] "AHA! Island." WGBH Educational Foundation. ahaisland.org (accessed.
[28] D. Kotsopoulos *et al.*, "A pedagogical framework for computational thinking," *Digital Experiences in Mathematics Education*, vol. 3, no. 2, pp. 154-171, 2017.
[29] NAEYC. "What You Need to Know About Tinkering, Making, and Engineering." https://www.naeyc.org/sites/default/files/globally-shared/downloads/PDFs/resources/pubs/sample_what_you_need_to_know_about_tinkering_making_and_engineering.pdf (accessed 2022).
[30] J. Su and W. Yang, "Artificial intelligence in early childhood education: A scoping review," *Computers and Education: Artificial Intelligence*, p. 100049, 2022.
[31] S. Malallah, J. Weese, L. Shamer, and W. Hsu, "Data Science (Dataying) for Early Childhood," presented at the ASEE Annual Conference, 2023.
[32] S. Malallah, J. Weese, and K. Alsalmi, "The "besTech" Technology Practice Framework for Early Childhood Education," presented at the ASEE Annual Conference, 2023.