

Analysis of Computational Thinking in Children's Literature for K-6 Students: Literature as a Non-Programming Unplugged Resource

Journal of Educational Computing
Research
0(0) 1–30

© The Author(s) 2021

Article reuse guidelines:

sagepub.com/journals-permissions

DOI: 10.1177/07356331211004048

journals.sagepub.com/home/jec



Evan David Ballard¹  and
Rachelle Haroldson¹ 

Abstract

As schools and districts across the United States adopt computer science standards and curriculum for K-12 computer science education, they look to integrate the foundational concepts of computational thinking (CT) into existing core subjects of elementary-age students. Research has shown the effectiveness of teaching CT elements (abstraction, generalization, decomposition, algorithmic thinking, debugging) using non-programming, unplugged approaches. These approaches address common barriers teachers face with lack of knowledge, familiarity, or technology tools. Picture books and graphic novels present an unexplored non-programming, unplugged resource for teachers to integrate computational thinking into their CT or CT-integrated lessons. This analysis examines 27 picture books and graphic novels published between 2015 and 2020 targeted to K-6 students for representation of computational thinking elements. Using the computational thinking curriculum framework for K-6, we identify the grade-level competencies of the CT elements featured in the books compared to the books' target age groups. We compare grade-level competencies to interest level to identify each CT element representation as “foundational,” “on-target,” or “advanced.” We conclude that literature

¹Teacher Education Department, University of Wisconsin-River Falls, River Falls, Wisconsin, United States

Corresponding Author:

Evan David Ballard, 2335 Blomquist Ave, White Bear Lake, MN 55110, United States.

Email: e.dave.ballard@gmail.com

offers teachers a non-programming unplugged resource to expose students to CT and enhance CT and CT-integrated lessons, while also personalizing learning based on CT readiness and interest level.

Keywords

computational thinking, computer science education, children's literature, elementary education, unplugged

As the field of computer science education has developed standards for K-12 students (Computer Science Teachers Association [CSTA], 2017; K-12 Computer Science Framework, 2016), states and districts across the United States have started to adopt plans that include computational thinking (CT) skills and concepts for elementary-age students (Code.org Advocacy Coalition, CSTA, & ECEP, 2019). These plans must include opportunities for teachers to learn how to integrate CT into curricula using familiar tools (Waterman et al., 2020), because teachers often hold incomplete preconceptions about CT (Cabrera, 2019). Unplugged activities are more accessible to teachers for integration into their instruction (Huang & Looi, 2021).

In this paper, we examine children's literature for representation of the elements of CT (abstraction, generalization, decomposition, algorithmic thinking, debugging) using the computational thinking curriculum framework for K-6 outlined by Angeli et al. (2016). We identify the grade-level competencies of the computational thinking elements featured in the books compared to the age groups for which the books were written. We present our findings as tables and figures that teachers can use to select texts to expose students to CT skills and concepts, enhance their CT and CT-integrated lessons, and personalize instruction for students based on their reading interest level and CT readiness.

Literature Review*Computational Thinking*

In her 2006 article "Computational Thinking", Jeannette Wing defined computational thinking as "solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science" (p. 33). Wing called on educators to recognize CT as an essential skill for all students in the 21st century due to the increasingly ubiquitous roles of

computing devices in all parts of modern life. Since then, CT has become an important term in education and has been included in non-computer science documents like the Next Generation Science Standards (NGSS Lead States, 2013), but definitions continue to vary. In fact, Denning (2017) argues that vague or inconsistent definitions and models have harmed computer science education by sowing confusion and diluting or misrepresenting computer science practices.

Palts and Pedaste (2020) reviewed papers on CT published since Wing's original 2006 article and grouped works into 6 major clusters, each centered on a key article that was used to build the model of CT. One of these clusters is based on a paper by Selby and Woollard (2013) that in turn drew from Barr and Stephenson (2011) and International Society for Technology in Education and CSTA (2011), but focused particularly on well-defined skills and de-emphasized data manipulation terms that were deemed too broad. The purposes of this model include "to define more narrowly, not more broadly . . . [and] to refine the definition to facilitate assessment" (Selby & Woollard, 2013, p. 1). The model distills CT into 5 subskills: abstraction, decomposition, algorithmic thinking, generalization, and evaluation. This steps-focused model was adopted by Angeli et al. (2016) to build a curriculum framework aimed at "engaging children in thinking and problem solving by developing a solution to a problem, automating the solution through algorithmic thinking, and generalizing this solution to new problems when common patterns are identified or recognized" (p. 50). The goal of this framework is "introducing students of a very young age to the thinking processes of computational thinking so they become competent to learn more advanced theoretical and practical topics of computer science in secondary education" (p. 50).

CT Integration

Guided by initiatives from organizations like the Computer Science Teachers Association (CSTA) and Code.org, the focus in CS education has expanded to include CS and CT across K-12 education (CSTA, 2017; Code.org Advocacy Coalition, CSTA, & ECEP, 2019). Schools and districts have turned their attention to how to address CT in their curriculum. The following sections outline teachers' conceptions of CT, teacher preparation and professional development efforts to develop CT knowledge, uncoupling of CT and programming, and non-programming approaches to teach, learn, and integrate CT.

Teacher Conceptions of CT and Preparing Teachers to Teach CT. Teacher preparation is key to CT implementation in curriculum. Cabrera (2019) found that in multiple studies, teachers, both in-service and pre-service, had unhelpful preconceptions about CT. Teachers equated technology and programming with CT and defined CT as general problem solving without naming any CT-specific elements or

practices. Teachers viewed CT as difficult to learn and unapproachable, partly because of the association with programming. In line with these findings, Barr and Stephenson (2011) called for providing teachers the necessary professional development and resources in order to integrate and embed CT across disciplines.

Preparation to teach and integrate CT begins with preservice teachers so they can develop ways of infusing CT in their practice. When elementary and secondary pre-service teachers were exposed to a CT module in a class on theories of learning and motivation, it improved their understanding of CT. Specifically, it strengthened the connection they made between problem solving and computational thinking as well as their understanding of ways to integrate CT across disciplines and without using a computer (Yadav et al., 2011).

Barriers to CT integration faced by in-service teachers, particularly elementary teachers, must be considered. Teachers cite challenges that range from lack of alignment with district priorities focused on reading and math to lack of resources in terms of instructional time, funds and practical examples of activities, and for this reason researchers have focused on an integrated approach to teaching CT (Israel et al., 2015; Hestness et al., 2018; Ketelhut et al., 2020). Yadav et al. (2016) recommend that elementary teachers receive support to develop their CT knowledge to authentically integrate CT tasks in their curricular context for successful implementation. Waterman et al. (2020) propose a framework for CT integration that exploits the content and CT overlap in core subjects like science and math to do “double duty.” They propose three levels of CT integration: exist, enhance, and extend. Integration at the “exist” level highlights aspects of CT already in the lesson. Integration at the “enhance” level involves adding tasks to make the connections between the disciplinary concept and computing content stronger. Integration at the “extend” level uses the disciplinary concepts as a way to explore CT applications.

When teachers experience professional development focusing on a CT integrated approach in core subjects (reading, science, mathematics), they are successful in implementing CT (Israel et al., 2015; Waterman et al., 2020). When the CT activities are unplugged they appear more accessible to teachers, which results in them integrating and using those activities with their students (Huang & Looi, 2021). Elementary teachers do not have the cognitive load for additional disciplines or unfamiliar tools, thus they require familiar tools (Waterman et al., 2020).

Even when elementary teachers come in with some conceptions of CT, professional development experiences shift their thinking from general broad definitions of CT as problem solving to more specific ideas that include types of logical thinking (like conditionals) and connections between algorithms to abstraction and generalization. These clearer understandings allowed teachers to see, then amplify the “seeds” of computational thinking in their STEM activities (Yadav et al., 2018).

Teaching Non-Programming CT. CT is at the core of computer science, and CT is often associated with programming. Researchers have shown that students as young as kindergarten learn CT skills through programming (Bers et al., 2014; Sullivan & Bers, 2016; Tran, 2019; Yin et al., 2020). Recently, researchers have proposed disconnecting CT from programming in order to focus on the process of problem-solving inherent in CT (Lu & Fletcher, 2009; Waterman et al., 2020; Brackmann et al., 2017; Hooshyar et al., 2020). This shift to an emphasis on CT skills apart from programming is evidenced by the change at the secondary level in the AP computer science courses. AP Computer Science A has been offered since the 1980's and emphasizes programming, currently in the Java programming language (College Board, 2021a). The big pedagogical shift happened in 2016 when the College Board introduced AP Computer Science Principles, an introductory course whose framework is guided by CT practices and a conceptual approach with the goal of exposing more students to CT in a way that places less emphasis on programming (College Board, 2021b).

Lu and Fletcher (2009) argue that students need to be exposed to and participate in CT separate from programming to develop the necessary CT skills. They suggest laying the foundation of CT with algorithmic processes, abstraction, and a common language of vocabulary and symbols, Computational Thinking Language (CTL), to “permeate the pedagogy.” They cite examples in mathematics and reading to illustrate the strategic introduction of CT vocabulary. Through repeated encounters of CT integrated across the curriculum teachers can show the ubiquity and importance of CT. While Caeli and Yadav (2020) argue for combining programming and non-programming approaches, they emphasize that non-programming or unplugged approaches develop deeper understanding of the problem-solving process.

Approaches Using Digital Technology. Non-programming CT activities can use technology. Hooshyar et al. (2020) introduced elementary students to an adaptive educational game (AutoThinking) that improved their CT skills (algorithmic thinking; pattern recognition and generalization; debugging; simulation) and taught CT concepts (sequence, conditional, loop). The adaptivity of the game personalized the learning for individual students and increased student interest, satisfaction, flow state, and tech acceptance. Waterman et al. (2020) integrated CT into the ecological system game *Oh! Deer* by adapting the existing activity for digital data collection and processing using spreadsheets. I. Lee et al. (2014) describe two different activities to integrate CT in the K-8 curriculum to bolster the practice approaches available to teachers; one using a pre-built computer model and simulation to learn about ecosystems and the other a web-based mobile tool for storing data to learn about velocity and acceleration.

Unplugged Approaches. Programming and non-programming CT activities can be “computer-less” or “unplugged,” meaning they do not require a computer or

device. Unplugged approaches allow teachers to integrate computer science into their classrooms regardless of the schools' access to hardware or the internet and regardless of the students' home access. Students miss out on opportunities that impact their grades, test scores, and college/career plans because of the digital divide, whereby students from different socio-economic backgrounds have different access to devices and internet services (Hampton et al., 2020).

Unplugged approaches, as well as the term, originated with 20 activities put together in Computer Science Unplugged (T. Bell et al., 1998) and evolved into the CS Unplugged website (CS Unplugged, 2021). These activities involve games, puzzles, stories, challenges, and sensory-rich experiences. Unplugged activities are low-cost, low-floor (accessible), and not material-intensive (V. R. Lee & Recker, 2018). Huang and Looi (2021) discuss how unplugged pedagogies foster CT development by complementing programming, integrating into other subjects, and supporting teaching and learning. They suggest that unplugged activities can serve as a "priming step" before programming with the aim to have students develop their understanding of algorithmic steps. In one study paper circuitry activities exposed students to algorithmic thinking, conditional logic, and debugging, to name a few, while using familiar materials and concepts to "provide multiple points of entry for students who are less familiar with computational thinking ideas." (V. R. Lee & Recker, 2018, p. 198). Brackmann et al. (2017) used existing unplugged resources (e.g., Hello Ruby) with upper elementary students that improved their CT skills.

Children's Literature as CT Resources. Picture books help abstract concepts become concrete. Massey (2015) suggests that, "In the hands of educators, picture books serve a much greater function than aesthetic reading; they can be a vehicle for the construction of knowledge and for solidifying concepts in a learning environment for older students" (p. 45). Whether as a resource to expose students to new material or enforce concepts already presented, literature has the potential to support CT integration. Furthermore, as elementary teachers work to integrate CT into existing curricula, they can take advantage of the overlap between CS standards and the Common Core State Standards For English Language Arts (CCSS-ELA) (National Governors Association Center for Best Practices & Council of Chief State School Officers, 2010).

While research has analyzed content and representation in literature for subjects like science (Kelly, 2018) and computer science (Haroldson & Ballard, 2021), no analysis has been done to look at the representation of computational thinking and its specific elements in children's books.

Study Context

As elementary teachers are more likely to integrate CT in their core subjects using accessible tools, literature has potential for CT integration. Picture books

and graphic novels have potential for non-programming CT integration as an unplugged resource that is low-cost and familiar. While simply reading texts demonstrating coding or computational thinking is not sufficient for students to learn CT, texts can be a valuable tool in instruction. The texts can serve as an important “priming” activity to prepare students to learn and practice CT (Huang & Looi, 2021). As Waterman et al. (2020) suggest, teachers can use this sort of non-programming unplugged activity to “enhance” a CT or CT-integrated lesson. To aid teachers in selecting texts to support their CT and CT-integrated instruction in this way, there is a need for an analysis reporting on CT in the children’s literature published in recent years.

To focus on picture books for elementary students, we sought a framework that emphasizes CT skills for these younger grades. The computational thinking curriculum framework for K-6 developed by Angeli et al. (2016) distills CT for these grades into 5 major elements: abstraction, generalization, decomposition, algorithmic thinking, and debugging. We adopted this framework for our analysis because it offers clear and concrete competency targets in each element for grades K-2, 3-4, and 5-6 that are written for compatibility with unplugged CT learning. Hereafter, the computational thinking curriculum framework for K-6 will be referred to as “the framework.” We distilled the purpose of this analysis into the guiding questions:

- What elements of computational thinking are represented in the books?
- What grade-level computational thinking competencies are represented in the books?

While many books about computer science and CT have been published for grades K-6 students, information about what is in these books needs to be made available for them to be easily incorporated into the elementary curriculum. This analysis focuses on reporting the CT elements represented in each book together with the grade-level interest range of the books and the grade-level CT competencies demonstrated. Teachers can use this information to select texts that are interesting and appropriate for their students to prime students for CT learning or enhance existing curriculum with CT elements. How teachers specifically use these books in their CT or CT-integrated lessons is outside of the scope of our analysis.

Methods

This analysis employed content analysis, a form of qualitative analysis, of 27 picture books and graphic novels targeting grades K-6 published between 2015 and mid-2020. Content analysis is often used in the fields of media studies, and children’s literature (P. Bell, 2001; Galda et al., 2000) and focuses on finding patterns in qualitative material with the data coming from text-based sources.

This reductive, sense-making effort aims to “identify core consistencies and meanings” (p. 453) that emerge as patterns or themes with patterns being more descriptive and themes being more categorical (Patton, 2002). At its core, content analysis investigates *what the text is about* (Galda et al., 2000). P. Bell (2001) also notes that the nature of content analysis is comparative, using categories from which relevant variables and values emerge. While Cohen et al. (2000) outline seven different purposes of content analysis, this analysis focuses on the purposes of examining content against standards and describing trends in the content.

Book Search and Selection Criteria

By 2015, there was national interest in the computer science education movement in the United States. This interest manifested in a number of high-profile initiatives, like the early 2016 White House announcement of the CSForAll initiative (Smith, 2016), and the release of the K-12 Computer Science Framework the same year (K-12 Computer Science Framework, 2016). Our analysis focuses on books published in the years from 2015 to 2020, during which this increased focus on CS education in grades K-12 was being realized.

For a systematic approach in finding relevant texts, subject headings for children’s books were confirmed in the Library of Congress. After searching broad terms, like “computer science,” that would encompass computational thinking in addition to the term, “computational thinking,” using the additional designator “juvenile literature” three standardized search terms emerged. These terms included: “computer science,” “computer programming,” and “computer scientists.” The phrase “computational thinking” was not found among the subject headers in the Library of Congress. For this reason, we moved forward in the search process using these three search terms that would include or be associated with computational thinking. While we cannot speak for certain why “computational thinking” is not found, we conjecture that the concept of computational thinking is still too new in the realm of children’s literature and as a result has not found its way into the official subject headers. Searches in the databases using the aforementioned search terms did indeed produce lists of books that included computational thinking.

We used two searchable and accessible databases found in the United States. Because our focus is on books for grades K-6, we used Novelist K-8, a database of children’s literature. We used WorldCat, a global library catalog, because it includes non-fiction and biographies. Table 1 shows the number of titles each database listed for a given search term. The WorldCat database searches are done within content criteria categories (e.g. “non-fiction,”) so these have also been listed in the table.

After the initial searches through the databases using the standardized search terms, we applied another round of criteria to narrow down the number of titles.

Table 1. Titles Found in Database by Search Term.

NoveList K-8		
	<i>Computer science</i>	109
	<i>Computer programming</i>	51
	<i>Computer scientists</i>	17
WorldCat		
Non-fiction	<i>Computer science</i>	558
	<i>Computer programming</i>	713
Biography	<i>Computer science</i>	65
	<i>Computer scientists</i>	58
Fiction	<i>Computer science</i>	341
	<i>Computer programming</i>	219

First, the interest age-range of the books, as provided by the publishers, needed to be in the K-6 grade band. If any part of the book's interest age-range was within grades K-6, the book was considered. Any texts completely outside the K-6 grade band (i.e., PreK, 7-12) were eliminated. Second, the book had to follow a narrative format, with characters actively solving a specific problem or small set of problems, which excluded any reference, how-to, and computer language specific books (e.g., Ruby, Python).

After these criteria were applied, we gathered together the books that remained for a first read-through to look for general evidence of computational thinking. This final step in the selection process ulted in narrowing down the list to the core 27 picture books and graphic novels (26 fiction and 1 non-fiction) used in the analysis (see Appendix for the complete list).

Text Analysis

First, we examined the text of 27 picture books and graphic novels against the elements of CT (abstraction, generalization, decomposition, algorithmic thinking, debugging) as laid out by the computational thinking curriculum framework for K-6 (Angeli et al., 2016). The text considered included any narration, character dialogue, and code or pseudocode represented in the images. Then, after identifying examples of the criteria present in each book, we looked for and described trends in the frequency of the elements of CT represented.

Analysis of the texts was based on close adherence to the language in the framework. The framework is prescriptive, focused on curriculum writing, while the analysis of the texts is descriptive, reporting the CT elements represented in each book. Because of this, minor ambiguities occasionally arose during the analysis. As each ambiguity arose, we each reviewed the framework, then jointly

developed guidelines that could be applied in the text in question and in the analysis of other texts (see Table 2).

An example of a minor ambiguity is in the algorithmic thinking section, where the grades 3–4 competency level specifies that the algorithm represented must iterate or loop, and the grades 5–6 competency level lists that the algorithm loops/iterates and includes conditionals, variables for storage and retrieval of information, and mathematical or logical expressions. After review of the framework, we added the specification that conditionals, variables, or advanced math or logic would demonstrate grades 5–6 competency even if no looping or iteration was represented. Not all algorithms need loops or iteration, so it would have harmed the utility of the analysis to insist on this aspect when other advanced elements of computational thinking were represented. One example of this is in *Power Coders: Day of the Gamer* (Vink & Gennari, 2019b), where there is no significant discussion of loops or iteration, but there is an involved discussion of conditionals for program flow.

Another ambiguity we encountered was in the abstraction section of the framework. The grades 3–4 range specifies “create a model/representation to solve a problem,” and the grades 5–6 range specifies “create a new model/representation to solve a problem” (Angeli et al., 2016, p. 50). We interpreted the key difference, the word “new,” to mean that students are expected to demonstrate the depth of understanding necessary to recreate the model in an appropriate context. If characters selected a model based on a clear description of its merits or a comparison with an alternative model, they demonstrated this depth of understanding. This put the CT competency represented in the book in the grades 5–6 range, whereas characters applying a known model without discussing its application, merits, or preferability compared to another model demonstrated competency in the grades 3–4 range. An example of this can be found in *Secret Coders (3): Secrets and Sequences* (Yang & Holmes, 2017a), in which the heroes escape imprisonment by using a small robot to lure away a robotic guard. They apply abstractions introduced earlier in the book, including conditionals and random numbers, and test multiple models in order to find a pattern of movement for the small robot that will capture the attention of the robot guard. They reason with their understanding of the specifics of how each model interacts with the environment (see Table 2).

To ensure consistent application of the analysis, we read each text a minimum of three times: once to become familiar with the text, once to flag possible examples of CT elements in the text, and once together to discuss each possible representation. Examining the text closely alongside the framework, a consensus was reached on each example of whether it clearly demonstrated the CT element, and if so, what grade-level competency was represented.

After cataloging representations of CT elements in the books, we identified which CT competency targets the book addressed in each element. This allowed us to highlight the competency grade-level range. If at least 50% of the book’s

Table 2. Computational Thinking Competency Targets by Grade Range.

Element/Skill	Grades K–2	Grades 3–4	Grades 5–6
Abstraction	With the use of external reference systems, create a model/representation* to solve a problem	Create a model/representation to solve a problem	Create a new model/representation to solve a problem
Application Notes	Characters were given a model or representation to use and they used it	Characters independently decided to use a model or representation	Characters invented their own abstraction or demonstrated the same depth of understanding that would be required to create the abstraction by deconstructing and discussing the merits of the abstraction
Generalization	Identify common patterns between older and newer problem-solving tasks, and use sequences of instructions previously employed, to solve a new problem	Remix and reuse (by extending if needed) resources that were previously created.	
Application Notes	Characters related a new problem to an old one	Characters went on to actually reuse, remix, or repurpose existing code or computational artifacts	
Decomposition	Break a complex task into a series of simpler subtasks	Break a complex task into simpler subtasks.	
Application Notes	Characters broke a problem down into smaller problems or tasks	Characters broke tasks down, then assembled the pieces into a functional whole	

(continued)

Table 2. Continued.

Element/Skill	Grades K–2	Grades 3–4	Grades 5–6
Algorithmic thinking	<p>Define a series of steps for a solution.</p> <p>Put instructions in the correct sequence.</p>	<p>Define a series of steps for a solution.</p> <p>Put instructions in the correct sequence.</p> <p>Repeat the sequence several times (iteration).</p>	<p>Define a series of steps for a solution.</p> <p>Put instructions in the correct sequence.</p> <p>Repeat the sequence several times (iteration).</p> <p>Make decisions based on conditions.</p> <p>Store, retrieve, and update variables.</p> <p>Formulate mathematical and logical expressions.</p> <p>Characters added conditionals, variables, or logical expressions to their sequence of steps (with or without repetition)</p>
Application Notes	<p>Characters defined steps and put them into an ordered sequence</p>	<p>Characters showed clear examples of repetition, looping, or iterating through a set</p>	
Debugging	<p>Recognize when instructions do not correspond to actions.</p> <p>Remove and fix errors.</p>		
Application Notes	<p>Characters engaged in troubleshooting or debugging unexpected behavior</p>		

Note. *model/representation = can be conceptual, mathematical, mechanical, textual, graphical, etc.
Adapted from Angeli et al. (2016).

interest grade-range was within its CT competency grade-range, then the average student who reads the book is likely to be within the grade-range in the framework, so we considered the text “on-target.” If there was not 50% overlap and an element presented itself at a grade band lower than that grade level interest band of the book, then the competence was considered “foundational.” If there was not 50% overlap and an element presented itself at a grade band higher than the grade level interest band of the book, then the competence was considered “advanced.”

Results

Figure 1 shows the number of books demonstrating each number of elements. Only one book demonstrated only one element, and 19 of the 27 books (70%) demonstrated 3 or more distinct elements. Six books demonstrated all 5 elements: *Hello Ruby: Adventures in Coding* (Liukas, 2015), *How to Code a Sandcastle* (Funk & Palacios, 2018), *Rox’s Secret Code* (Lecocq et al., 2018), *Secret Coders: Paths & Portals* (Yang & Holmes, 2016), *Secret Coders: Secrets and Sequences* (Yang & Holmes, 2017a), and *Secret Coders: Robots and Repeats* (Yang & Holmes, 2017b).

Figure 2 shows the frequency with which each CT element appeared in the books. Abstraction appeared the most frequently (23 books), and generalization appeared the least (10 books). The distribution among “foundational,” “target,” and “advanced” was most even in abstraction and algorithmic thinking because

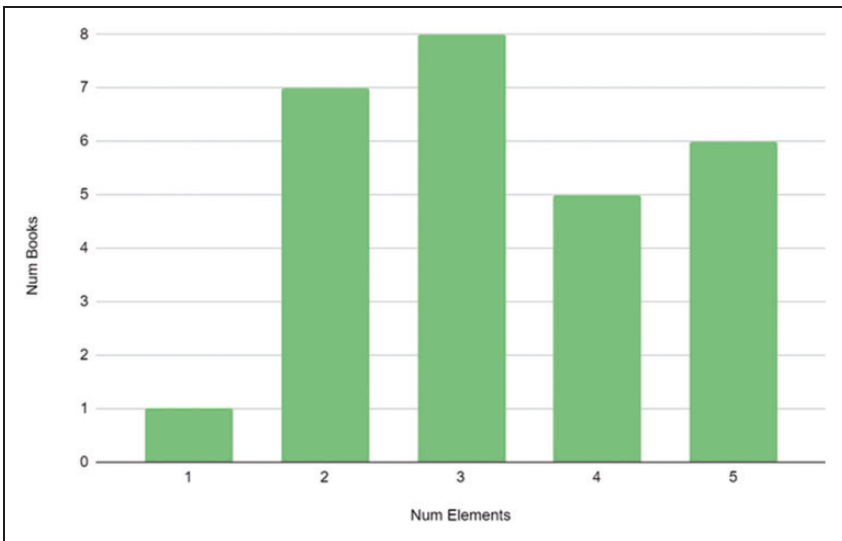


Figure 1. Number of Books Demonstrating a Given Number of Elements.

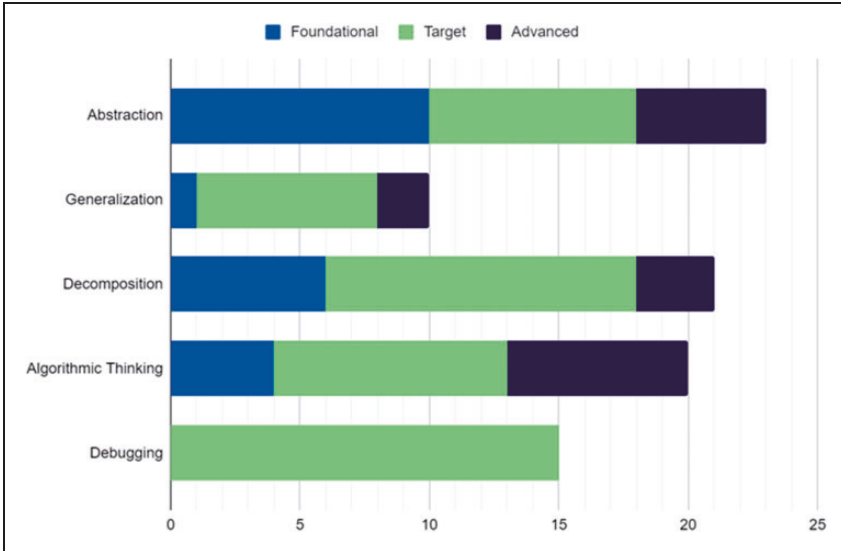


Figure 2. Number of Books Including Each Element.

these have the most granularity in the framework, having separate competency targets for grades K-2, 3-4, and 5-6. Generalization and decomposition are dominated by “on target” examples because the wide grades 3-6 competency band in each made it less likely that any particular group of students would be outside the CT competency grade-range. The framework includes only one set of competency targets for debugging, applying to grades K-6, so all examples are considered “on target.”

Figures 3 and 4 show the title of each book with its interest grade-range (the black bar) and the CT competency ranges (other-colored bars) represented in the book. The vertical gridlines separate grade levels. The books are arranged by increasing interest grade-range. A given color of bar will only appear if the corresponding CT element was represented in the book, and will stretch between the lowest grade-level and highest grade-level from the range specified in the framework (Angeli et al., 2016) as outlined in Table 2. Figure 3 shows the books with interest ranges beginning in grades K-2, and Figure 4 shows the books with interest ranges beginning in grades 3-6.

Abstraction

We found demonstrated examples of abstraction in 23 of our 27 books, more than any other element. Of those, 10 were foundational for the interest

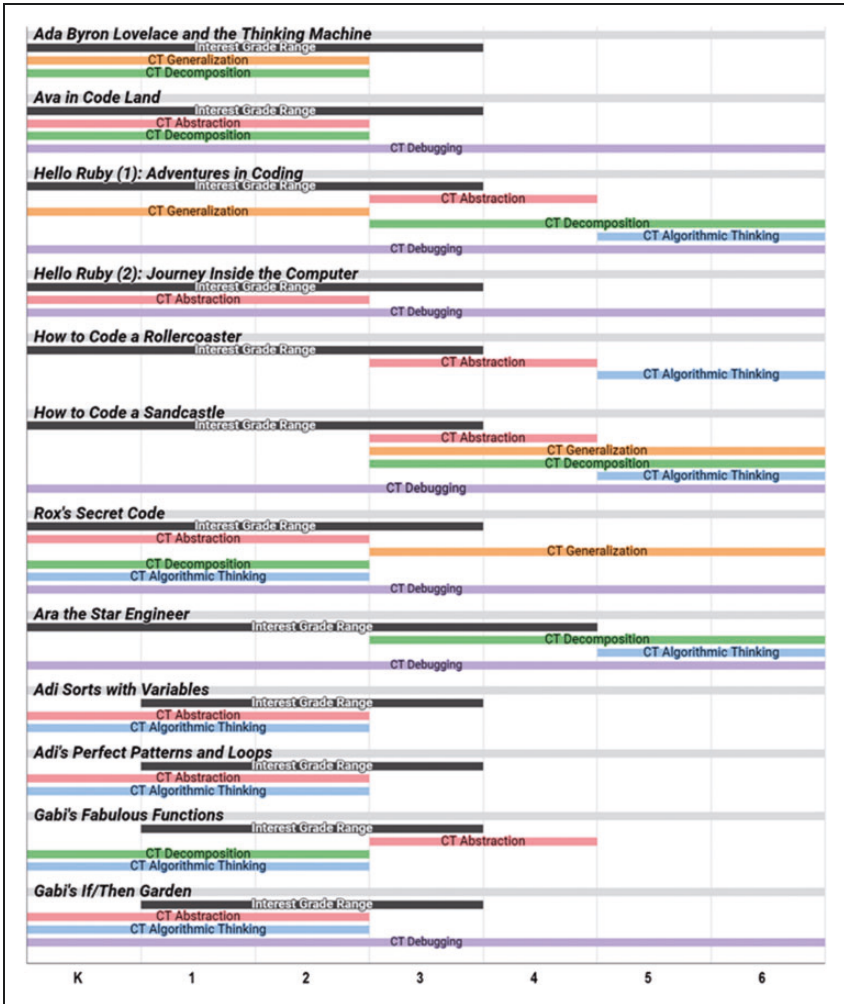


Figure 3. Interest and CT Competency Ranges (Interest Grade Range K-3).

grade-range, 8 aligned with grade-range targets, and 5 were advanced for the interest grade-range.

How to Code a Rollercoaster (Funk & Palacios, 2019) is an excellent example of a book that presents abstraction to younger readers. It carefully explains multiple models used in computer science, including variables, booleans, and conditionals, by demonstration. The main characters Pearl and Pascal make decisions based on conditions and track their progress as they enjoy a theme park, updating the reader at every step in terms of these abstractions. The book

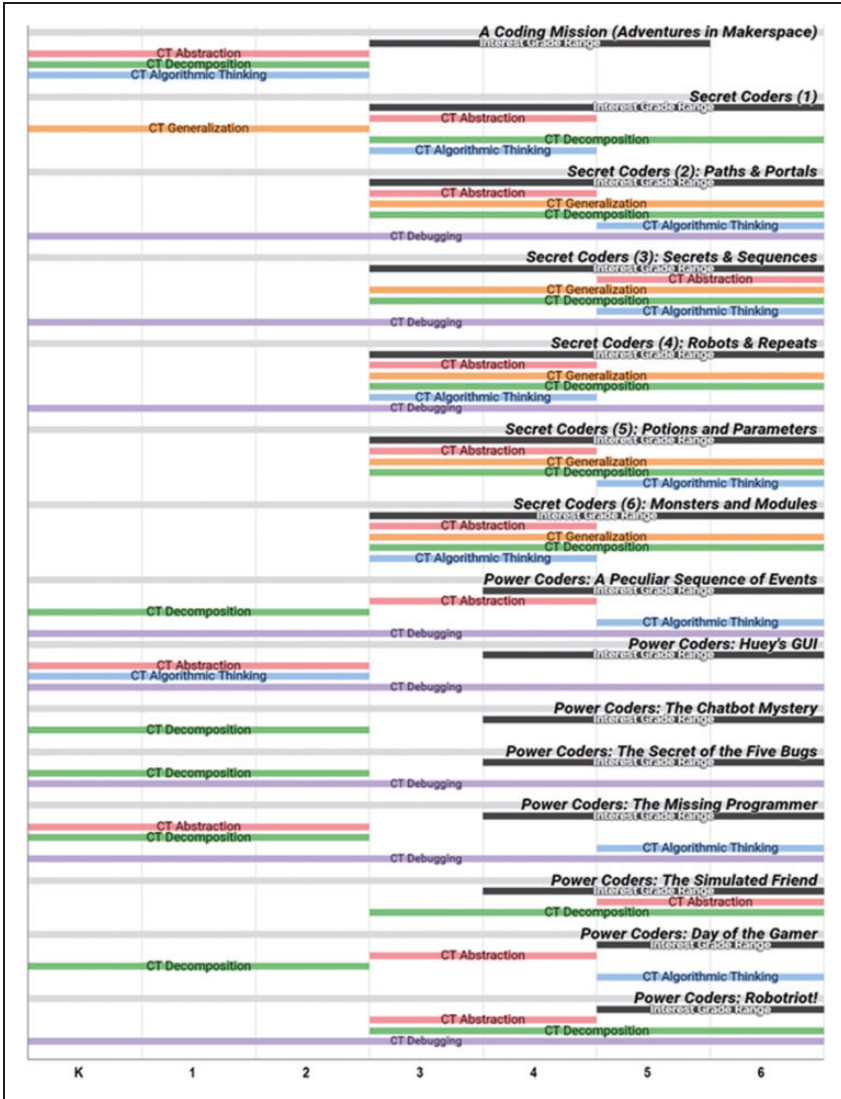


Figure 4. Interest and CT Competency Ranges (Interest Grade Range 3–6).

meets targets in the grades 3-4 band, as it not only demonstrates abstractions but discusses their use in detail.

Ava in Code Land (Hitchman et al., 2020) does something similar, showing a fantastic land of objects with attributes that can be read and edited (alongside

snippets of real Javascript code). This book meets targets in the grades K-2 band, with a concrete and limited demonstration of simple abstraction.

Secret Coders (Yang & Holmes, 2015), the first volume in the series, provides a strong example of abstraction at a foundational level; the book is written for grades 3-7 and the abstraction is in the 3-4 CT competency band. Eni realizes the strange birds around school have eyes that display binary numbers, a system made of only two numbers, 0 and 1. Eni teaches Hopper about binary using pennies to model when the eyes on the robot birds are open (column full of pennies) or closed (column empty).

Generalization

We found demonstrated examples of generalization in 10 of our 27 books, making it the least-represented element in our books. Of those, 2 were foundational for the interest grade-range, 6 aligned with grade-range targets, and 2 were advanced for the interest grade-range. The inclusion of a broader band for this CT practice in the framework (a 3-6 band) made it more likely for the CT grade-range target to overlap with the interest grade-range. This effect accounts for the large proportion of examples which were considered aligned with grade-range targets. For younger readers in grades K-3, *How to Code a Sandcastle* (Funk & Palacios, 2018) provides a straightforward advanced example of generalization at the 3-6 grade band when Pearl re-uses the code for her sandcastle to solve a new problem. She adds on steps for building a moat to create a sandcastle kingdom.

One clear example of generalization for older readers (grades 3-7) happens in *Secret Coders: Paths and Portals* (Yang & Holmes, 2016). Hopper and Eni reuse the OpenSesame program that has the turtle walk a hexagon to develop a program (now called HopperRocks) that has the turtle walk a hexagon with lines coming out of the corners. This text then demonstrates how algorithmic thinking and debugging work together, building on the generalized solution. They forget to have the turtle turn around at the end of the line and have to fix the program to include a 180 turn.

In *Secret Coders: Robots and Repeats* (Yang & Holmes, 2017b), Mr. Bee asks the Coders to rework the programs for DrawTriangle and DrawSquare to draw a pentagon. Josh realizes they need to adjust the number of sides in the Repeat line. Hopper and Eni determine how to calculate the change in angle for the turns in a pentagon. Mr. Bee then pushes them to “write a program that can draw a regular polygon with any number of sides” (Yang & Holmes, 2017b, p. 77). This demonstrates one of many examples of two elements interwoven together: the students apply generalization and abstraction to develop a program that accepts a parameter.

Decomposition

We found demonstrated examples of decomposition in 21 of our 27 books, tying for second most represented element with algorithmic thinking. Of those, 6 were foundational for the interest grade-range, 12 aligned with grade-range targets, and 3 were advanced for the interest grade-range. Like with generalization, a broader band of grades 3-6 made overlap with the interest grade-range more likely. This leads to a large proportion of examples aligning with grade-range targets. A good example of decomposition is presented in *Ara the Star Engineer* (Singh & Konak, 2018), in which Ara decides to count the stars in the universe and consults with several professional engineers to break the problem into smaller tasks, then assemble the smaller solutions they develop into a complete program for her robot. This places the book clearly in the grades 3-6 band for decomposition as outlined in the K-6 Framework.

Another example, in a book targeted at older students, is in *Power Coders: Robotriot!* (Roza & Gennari, 2020). In this book, the Power Coders assist another student who is building a battle robot for a competition. They examine each robot task in terms of individual commands to robot parts, then discuss the effects of the basic functions on the overall robot behavior (assembling the smaller pieces into a complete, functional program for the robot). Like in *Ara the Star Engineer*, the detailed breaking down of tasks and then assembly of simpler parts into a whole meets the requirements for the grade 3-6 band for decomposition.

In *Hello Ruby: Adventures in Coding* (Liukas, 2015), Ruby demonstrates decomposition in the grade 3-6 band several times in the book, notably when she plans her adventure in parts at the beginning of the day and when she helps a group of foxes develop a plan for gardening by starting with basic tasks.

Algorithmic Thinking

We found demonstrated examples of algorithmic thinking in 20 of our 27 books, tying for second most represented element with decomposition. Of those, four were foundational for the interest grade-range, nine aligned with grade-range targets, and seven were advanced for the interest grade-range. For example, in *Gabi's Fabulous Functions* (Karanja & Whitehouse, 2019c), Adi and Gabi apply the steps of recipes to understand functions. They have to define and order the steps to gather the ingredients, berries, yogurt, and granola (input) and assemble them together (function) to produce the finished parfait (output). Then in *Adi's Perfect Patterns and Loops* (Karanja & Whitehouse, 2019b), Adi and Gabi learn about the repeating routes the mail carrier and bus driver follow. After they put together the sequence of steps, they apply the concept of loops to their train's route (move to station, drop off, pick up, repeat).

The Power Coders, in *Power Coders: A Peculiar Sequence of Events* (Vink & Gennari, 2019a), realize the time loop they are in that involves one of their classmates getting bullied at the end of class is just like a coding “while” loop. They demonstrate algorithmic thinking by ordering the events leading up to the bullying, and they then figure out the exit condition of the loop in order to stop both the loop and the bullying.

In *Power Coders: The Missing Programmer* (Bowen & Gennari, 2019b), the Power Coders and Ms. Jones use a flowchart to assist them with the mystery. The flowchart is a logical expression that helps them understand the data inputs, variables, and decisions for the program the missing programmer left them. The flowchart also helps the group determine where the error is (a line of code is missing), a clear example of the interaction connection of the elements of algorithmic thinking and debugging.

Debugging

We found demonstrated examples of debugging in 15 of our 27 books, making it the second least-frequently occurring element in the books. As the framework defines only one broad (K-6) band for debugging, all examples overlapped with interest grade-ranges enough to be considered aligned with grade-range targets. Because of the wide competency band for this element, debugging bands cover the entire range in Figures 3 and 4.

An example of debugging comes from *Power Coders: Huey’s GUI* (Bowen & Gennari, 2019a). Grace looks at the code for ordering cupcakes and finds a problem with the code; rather than getting the cupcake flavors they ordered, everyone gets carrot cake.

For younger readers, *Rox’s Secret Code* (Lecocq et al., 2018) shows the debugging process when Rox’s chore robot goes berserk and begins destroying the city in an effort to organize everything in its path, including cars, buildings, and people. Rox discusses the root of the behavior with a friend and formulates a plan, first devising a way to distract the robot (based on her knowledge of its behavior) and then patching its code with the code from another robot. Rox demonstrates the thought process of debugging and the attitude of perseverance that students need to fix problems in their own programs.

Hello Ruby: Journey Inside the Computer (Liukas, 2017) uses basic debugging as its entire premise, following Ruby as she explores the interior of her computer and rules out reasons her mouse is not working.

Discussion

In this analysis, 27 children’s books and graphic novels were analyzed for elements of CT as outlined in the curriculum framework for K-6 (Angeli et al., 2016). We sought to answer the guiding questions: “What elements of CT are

represented in the books?” and “what grade-level CT competencies are represented in the books?” From this analysis it became apparent that the books represented CT at three different levels relative to the interest grade-ranges of the books (foundational, on-target, advanced). We found a variety of CT elements and competency grade-ranges for books at every interest level. We argue these books provide opportunities for priming students for learning CT and enhancing curriculum with CT elements, as well as opportunities for differentiated learning based on student interests and readiness in CT. Table 3 shows our findings of which CT elements are included at what competency level in each text (the table is arranged by interest level for easy selection of appropriate texts for a given group of students).

CT Integration With Literature: Non-Programming Unplugged Resource

The results of this analysis show that children’s literature demonstrates elements of CT at varying competency levels. Books present students with a non-programming unplugged resource to expose them to the ideas, vocabulary, symbols, and skills of CT as they build that foundation (Lu & Fletcher, 2009). Teachers can choose how to integrate instruction with elements of CT according to the three levels of integration: exist, enhance, and extend (Waterman et al., 2020). Whether teachers work within the disciplines they are already teaching to make existing lessons more CT-infused lessons or develop new CT lessons, children’s books offer examples, scenarios, and problems related to CT for use in the classroom.

Teachers are more likely to integrate CT activities if they are accessible and use familiar tools, thus lessening the cognitive load (Huang & Looi, 2021; Waterman et al., 2020). Given the priority of reading as a core subject in the elementary grades, teachers are used to weaving books into their lessons. Teachers do not have to take on more cognitive load with books because they represent a familiar tool. In this way teachers can instead focus on whether to integrate books with an existing lesson to highlight an aspect of CT or enhance a lesson by adding in books as another way to show connections. In addition, some of the book series like *Secret Coders*, *Hello Ruby*, and *Adi/Gabi* (see Appendix) offer additional activities at the end of the story that teachers can integrate to enhance or extend their lessons. If students want to take the books home, they can do these unplugged exercises with their families without the need for a computer.

Personalizing CT Learning

Accommodating the different ways students learn is referred to as differentiation (Tomlinson & Allan, 2000). Teachers personalize the learning for students by differentiating the content students will learn (instruction), the activities

Table 3. Computational Thinking Competencies by Book.

Book Title	Interest		Abstraction	Generalization	Decomposition	Algorithmic Thinking	Debugging
	Grade	Range					
Ada Byron Lovelace and the Thinking Machine (Wallmark & Chu, 2015)	0-3	0-3	Tgt ✓	Tgt ✓	Tgt ✓		Tgt ✓
Ava in Code Land (Hitchman, Cullen, & Martin, 2020)	0-3	0-3	Tgt ✓		Tgt ✓		Tgt ✓
Hello Ruby: Adventures in Coding (1) (Liukas, 2015)	0-3	0-3	Adv ↑	Tgt ✓	Adv ↑	Adv ↑	Tgt ✓
Hello Ruby: Journey Inside the Computer (2) (Liukas, 2017)	0-3	0-3	Tgt ✓				Tgt ✓
How to Code a Rollercoaster (Funk & Palacios, 2019)	0-3	0-3	Adv ↑			Adv ↑	
How to Code a Sandcastle (Funk & Palacios, 2018)	0-3	0-3	Adv ↑	Adv ↑	Adv ↑	Adv ↑	Tgt ✓
Rox's Secret Code (Lecocq, Archambault, von Innerebner, & Dengo, 2018)	0-3	0-3	Tgt ✓	Adv ↑	Tgt ✓	Tgt ✓	Tgt ✓
Ara the Star Engineer (Singh & Konak, 2018)	0-4				Adv ↑	Adv ↑	Tgt ✓
Adi Sorts with Variables (Karanja & Whitehouse, 2019a)	1-3		Tgt ✓			Tgt ✓	
Adi's Perfect Patterns and Loops (Karanja & Whitehouse, 2019b)	1-3		Tgt ✓			Tgt ✓	
Gabi's Fabulous Functions (Karanja & Whitehouse, 2019c)	1-3		Adv ↑		Tgt ✓	Tgt ✓	
Gabi's If/Then Garden (Karanja & Whitehouse, 2019d)	1-3		Tgt ✓			Tgt ✓	Tgt ✓
A Coding Mission (Adventures in Makerspace) (Miller, Hoena, & Brown, 2019)	3-5		Fnd ↓		Fnd ↓	Fnd ↓	
Secret Coders (1) (Yang & Holmes, 2015)	3-7	3-7	Fnd ↓	Fnd ↓	Tgt ✓	Fnd ↓	Tgt ✓
		3-7	Fnd ↓	Tgt ✓	Tgt ✓	Adv ↑	Tgt ✓

(continued)

Table 3. Continued.

Book Title	Interest Grade Range	Algorithmic Thinking					
		Abstraction	Generalization	Decomposition	Algorithmic Thinking	Debugging	
Secret Coders (2): Paths & Portals (Yang & Holmes, 2016)	3-7	Adv ↑	Tgt ✓	Tgt ✓	Adv ↑	Tgt ✓	
Secret Coders (3): Secrets & Sequences (Yang & Holmes, 2017a)	3-7	Fnd ↓	Tgt ✓	Tgt ✓	Fnd ↓	Tgt ✓	
Secret Coders (4): Robots & Repeats (Yang & Holmes, 2017b)	3-7	Fnd ↓	Tgt ✓	Tgt ✓	Adv ↑		
Secret Coders (5): Potions & Parameters (Yang & Holmes, 2018a)	3-6	Tgt ✓	Tgt ✓	Tgt ✓	Tgt ✓		
Secret Coders (6): Monsters & Modules (Yang & Holmes, 2018b)	4-6	Fnd ↓	Fnd ↓	Fnd ↓	Tgt ✓	Tgt ✓	
Power Coders: A Peculiar Sequence of Events (Vink & Gennari, 2019a)	4-6	Fnd ↓			Fnd ↓	Tgt ✓	
Power Coders: Huey's GUI (Bowen & Gennari, 2019a)	4-6					Tgt ✓	
Power Coders: The Chatbot Mystery (Bowen & Gennari, 2019a)	4-6			Fnd ↓		Tgt ✓	
Power Coders: The Secret of the Five Bugs (McKay & Gennari, 2019b)	4-6			Fnd ↓		Tgt ✓	
Power Coders: The Missing Programmer (Bowen & Gennari, 2019b)	4-6	Fnd ↓		Fnd ↓	Tgt ✓	Tgt ✓	
Power Coders: The Simulated Friend (Vink & Gennari, 2019c)	4-6	Tgt ✓		Tgt ✓			
Power Coders: Day of the Gamer (Vink & Gennari, 2019b)	5-8	Fnd ↓		Fnd ↓	Tgt ✓		
Power Coders: Robotriot! (Roza & Gennari, 2020)	5-8	Fnd ↓		Tgt ✓		Tgt ✓	

Fnd = Foundational, Tgt = Target, Adv = Advanced.

students will engage in (curriculum), or the products they produce to show what they know (assessment) (Sousa & Tomlinson, 2018). Personalization of the learning for students has shown to be effective in improving the CT skills of students in non-programming approaches (Hooshyar et al., 2020) and is a factor attributed to student success with computing experiences (Israel et al., 2015). Viewing literature as another non-programming approach to exposing students to CT, with its varying reading levels and representation of CT competencies, offers support for different learners.

Differentiation happens according to readiness and student interest (Doubet & Hockett, 2018). We argue that because the books in this analysis have a range of interest levels and CT competencies, teachers can differentiate the books to meet the needs of their students based on reading level and CT competency. They can personalize the books to students' interests and capture their motivation using the interest ranges for the books and their overall storylines and characters. Books like *Rox's Secret Code* (Lecocq et al., 2018) and *Hello Ruby: Adventures in Coding* (Liukas, 2015) will draw emerging readers into their stories with their colorful images and persistent, problem solving characters.

Teachers can personalize the books based on the students' incoming CT knowledge using the information about whether the grade level competencies are foundational, on target, or advanced. For younger readers who may have had less exposure to CT, using books with evidence of CT competencies that are foundational will help to prime students' understanding as well as increase their knowledge of key vocabulary, both considered strategies for building CT skills before programming (Huang & Looi, 2021; Lu & Fletcher, 2009). For students who are more confident in their CT skills, teachers can direct them towards a series like *Secret Coders* (see Appendix) which represents CT competencies on target with a sprinkle of advanced competencies.

Teachers can engage in tiering, a strategy for differentiation, that involves creating different levels within a task based on students' readiness. Doubet and Hockett (2018) explain that teachers "can differentiate content by pulling resources that vary in complexity" (p. 209). With literature covering elements of CT at varying levels of competency, this is certainly possible. Students can read books based on their CT readiness in addition to their reading interest. Engaging in this level of personalization meets the needs of the learners. Knowing that readiness and interest change over time, teachers can use the books fluidly, adjusting as needed.

As an example, an emerging reader in 1st grade who has been exposed to computers and Scratch Jr. would do well with the *How to Code* books (see Appendix). They are targeted for the interests of K-3 and full of advanced examples in all elements of CT. Additionally, they use models that students can relate to: building sandcastles and using maps. A 5th grader with a passion for graphic novels and a very recent exposure to computer science would do well

to start with the *Power Coders* series (see Appendix). The books are often on target with the CT elements and incorporate clear examples with accompanying pictures and diagrams to support deeper learning. Teachers of younger students looking to read out loud and incorporate whole class discussions to develop skills about predicting what happens next and identifying the process of problem solving should look to *Ava in Code Land* (Hitchman et al. 2020) and *Ara the Star Engineer* (Singh & Konak, 2018).

For teachers looking to provide a more extensive coverage of CT, *Rox's Secret Code* (Lecocq et al., 2018), *Hello Ruby: Adventures in Coding* (Liukas, 2015), and *How to Code a Sandcastle* (Funk & Palacios, 2018) include examples of all five elements of CT at target and advanced levels. For older readers, books 2, 3, and 4 from the *Secret Coders* series provide full coverage of all areas of CT. In this way, teachers can select which elements they want to highlight with options to highlight all of them. Teachers can use one book to teach many elements or use multiple books to focus on one element.

Limitations

This analysis made use of the computational thinking curriculum framework for K-6 developed by Angeli et al. (2016), and so the scope of the analysis is limited by the scope of the framework itself. The framework uses varied granularity in its treatment of the CT elements (for example, it lists the same debugging benchmarks for grades K-6), and in adopting the framework this analysis included the same variance in the width of age-ranges. In addition, the framework is focused on curriculum writing and so is prescriptive, but the analysis of existing texts is necessarily descriptive, so a small amount of interpretation and adaptation of the framework was necessary, as detailed in Table 2.

This analysis examined how the available children's literature represents CT elements and competencies as outlined in the framework. The scope of the guiding questions does not extend to specific instructional design or integration of these texts into lessons. The books by themselves cannot be expected to teach CT to students, but they can be used to help "prime" students (Huang & Looi, 2021) and help teachers enhance their CT or CT-integrated instruction (Waterman et al., 2020).

The framework lists competencies for grades K-6. To expand this research to the entire K-12 range, a framework that includes grades 7-12 would be required. In addition, a survey of books for middle and high school students would require more work in categorizing and characterizing these texts. Books written for grades 7-12 students are very different from the short-form picture books for grades K-6 in length, density and presentation of content, and narrative style. At the very least, the study would need to adapt to the fact that substantially greater length and density imply that these books would often incorporate many or all of the CT elements.

To list texts' interest grade-ranges, we relied on published recommendations from the publishers. When raw age instead of grade-range was used, we converted using the standard United States age-ranges. While some databases of books publish age-range recommendations, we found that they were inconsistent (sometimes recommending books from the same series to widely different audiences), whereas publishers showed general consistency. However, we must acknowledge that there are certainly at least minor differences between publishers in how they determine age-range recommendations.

Conclusion

Teaching CT is foundational to computer science education and a key component in CS standards and non-CS standards across the United States. More and more teachers are being asked to teach CT with universities integrating CT in their teacher preparation programs. Successful implementation of CT involves strategies that support teachers, weaving CT into their existing curricula to enhance what they are already teaching, preferably using non-programming unplugged approaches as those are perceived as accessible and familiar. The books from this analysis offer a non-programming unplugged resource, and one that is familiar for elementary teachers. In addition, children's literature affords opportunities for teachers to personalize the learning for students based on CT readiness and reading level.

Appendix

References for Children's Books

- Bowen, L. A., & Gennari, J. (2019a). *Power coders: Huey's GUI*. PowerKids Press.
- Bowen, L. A., & Gennari, J. (2019b). *Power coders: The missing programmer*. PowerKids Press.
- Funk, J., & Palacios, S. (2018). *How to code a sandcastle*. Viking.
- Funk, J., & Palacios, S. (2019). *How to code a rollercoaster*. Viking.
- Hitchman, J., Cullen, G., & Martin, L. (2020). *Ava in code land*. Feiwel and Friends.
- Karanja, C., & Whitehouse, B. (2019a). *Adi sorts with variables*. Picture Window Books.
- Karanja, C., & Whitehouse, B. (2019b). *Adi's perfect patterns and loops*. Picture Window Books.
- Karanja, C., & Whitehouse, B. (2019c). *Gabi's fabulous functions*. Picture Window Books.
- Karanja, C., & Whitehouse, B. (2019d). *Gabi's if/then garden*. Picture Window Books.

Lecocq, M., Archambault, N., von Innerebner, J., & Dengo, R. (2018). *Rox's secret code*. POW!.

Liukas, L. (2015). *Hello Ruby: Adventures in coding*. Feiwei and Friends.

Liukas, L. (2017). *Hello Ruby: Journey inside the computer*. Feiwei and Friends.

McKay, C. R., & Gennari, J. (2019a). *Power coders: The chatbot mystery*. PowerKids Press.

McKay, C. R., & Gennari, J. (2019b). *Power coders: The secret of the five bugs*. PowerKids Press.

Miller, S. M., Hoena, B., & Brown, A. (2019). *A coding mission (Adventures in Makerspace)*. Stone Arch Books

Roza, G., & Gennari, J. (2020). *Power coders: Robotriot!* PowerKids Press.

Singh, K., & Konak, I. (2018). *Ara the star engineer*. Page Two Books.

Vink, A., & Gennari, J. (2019a). *Power coders: A peculiar sequence of events*. PowerKids Press.

Vink, A., & Gennari, J. (2019b). *Power coders: Day of the gamer*. PowerKids Press.

Vink, A., & Gennari, J. (2019c). *Power coders: The simulated friend*. PowerKids Press.

Wallmark, L., & Chu, A. (2015). *Ada Byron Lovelace and the thinking machine*. Creston Books.

Yang, G. L., & Holmes, M. (2015). *Secret coders*. First Second.

Yang, G. L., & Holmes, M. (2016). *Secret coders: Paths & portals*. First Second.

Yang, G. L., & Holmes, M. (2017a). *Secret coders: Secrets & sequences*. First Second.

Yang, G. L., & Holmes, M. (2017b). *Secret coders: Robots & repeats*. First Second.

Yang, G. L., & Holmes, M. (2018a). *Secret coders: Potions & parameters*. First Second.

Yang, G. L., & Holmes, M. (2018b). *Secret coders: Monsters & modules*. First Second.


Declaration of Conflicting Interests


The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The authors disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by the National Science Foundation under Award #1660135 and the National Science Foundation under Award #1439768. Special thanks to Dr. Jaime Ballard and Dr. Earl Blodgett, who provided feedback on earlier drafts.

ORCID iDs

Evan David Ballard  <https://orcid.org/0000-0002-5554-8626>

Rachelle Haroldson  <https://orcid.org/0000-0002-5375-846X>

References

- Code.org Advocacy Coalition, CSTA, & ECEP. (2019). *2019 State of Computer Science Education*. https://advocacy.code.org/2019_state_of_cs.pdf
- Angeli, C., Voogt, J., Fluck, A., Webb, M., Cox, M., Malyn-Smith, J., & Zagami, J. (2016). A K-6 computational thinking curriculum framework: Implications for teacher knowledge. *Educational Technology & Society, 19*(3), 47–57.
- Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? *ACM Inroads, 2*(1), 48–54.
- Bell, P. (2001). Content analysis of visual images. In T. V. Leeuwen, & C. Jewitt (Eds.), *Handbook of visual analysis* (pp. 10–34). SAGE Publications.
- Bell, T., Witten, I., & Fellows, M. (1998). *Computer science unplugged: Off-line activities and games for all ages*. Computer Science Unplugged. <http://jmvidal.cse.sc.edu/library/bell98a.pdf>
- Bers, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers & Education, 72*, 145–157.
- Brackmann, C. P., Román-González, M., Robles, G., Moreno-Léon, J., Casali, A., Barone, D. (2017). Development of computational thinking skills through unplugged activities in primary school. In *Proceedings of 12th Workshop in Primary and Secondary Computing Education* (pp. 65–72). ACM. <https://doi.org/10.1145/3137065.3137069>
- Cabrera, L. (2019). Teacher preconceptions of computational thinking: A systematic literature review. *Journal of Technology and Teacher Education, 27*(3), 305–333.
- Caeli, E. N., & Yadav, A. (2020). Unplugged approaches to computational thinking: A historical perspective. *TechTrends, 64*(1), 29–36. <https://doi.org/10.1007/s11528-019-00410-5>
- Cohen, L., Manion, L., & Morrison, K. (2000). *Research methods in education*. 5th ed. RoutledgeFalmer.
- College Board. (2021a, February 21). *AP central: AP computer science A*. <https://apcentral.collegeboard.org/courses/ap-computer-science-a/course>
- College Board. (2021b, January 5). *AP central: AP computer science principles*. <https://apcentral.collegeboard.org/courses/ap-computer-science-principles?course=ap-computer-science-principles>
- Computer Science Teachers Association. (2017). *CSTA K-12 Computer Science Standards, Revised 2017*. <http://www.csteachers.org/standards>.
- CS Unplugged. (2021). *Computer science without a computer*. <https://csunplugged.org/en/>
- Denning, P. J. (2017). Remaining trouble spots with computational thinking. *Communications of the ACM, 60*(6), 33–39.
- Doubet, K. J., & Hockett, J. A. (2018). *Differentiation in the elementary grades: Strategies to engage and equip all learners*. ASCD.

- Galda, L., Ash, G. E., & Cullinan, B. E. (2000). Research on children's literature. In M. L. Kamil, P. B. Mosenthal, P. D. Pearson, & R. Barr (Eds.), *Handbook of reading research: Volume III* (pp. 351–381). Erlbaum.
- Hampton, K. N., Fernandez, L., Robertson, C. T., & Bauer, J. M. (2020). *Broadband and student performance gaps*. In H. James, & B. Mary (Eds.), Quello Center, Michigan State University. <https://doi.org/10.25335/BZGY-3V91>
- Haroldson, R. & Ballard, D. (2021). Alignment and representation in computer science: an analysis of picture books and graphic novels for K-8 students. *Computer Science Education*, 31(1), 4–29. <https://doi.org/10.1080/08993408.2020.1779520>.
- Hestness, E., Ketelhut, J. D., McGinnis, R., & Plane, J. (2018). Professional knowledge building within an elementary teacher professional development experience on computational thinking in science education. *Journal of Technology and Teacher Education*, 26(3), 411–435.
- Hooshyar, D., Pedaste, M., Yang, Y., Malva, L., Hwang, G., Wang, M., Lim, H., & Delev, D. (2020). From gaming to computational thinking: An adaptive educational computer game-based learning approach. *Journal of Educational Computing Research*, 1–27. <https://doi.org/10.1177/0735633120965919>
- Huang, W., & Looi, C. K. (2021). A critical review of literature on “unplugged” pedagogies in K-12 computer science and computational thinking education. *Computer Science Education*, 31(1), 83–111. <https://doi.org/10.1080/08993408.2020.1789411>
- International Society for Technology in Education & Computer Science Teachers Association. (2011). *Operational definition of computational thinking for K–12 education*. National Science Foundation.
- Israel, M., Pearson, J. N., Tapia, T., Wherfel, Q. M., & Reese, G. (2015). Supporting all learners in school-wide computational thinking: A cross-case qualitative analysis. *Computers & Education*, 82, 263–279.
- K-12 Computer Science Framework. (2016). *K-12 Computer Science Framework*. <http://www.k12cs.org>
- Kelly, L. B. (2018). An analysis of award-winning science trade books for children: Who are the scientists, and what is science? *Journal of Research in Science Teaching*, 55(8), 1188–1210. <https://doi.org/10.1002/tea.21447>
- Ketelhut, D. J., Mills, K., Hestness, E., Cabrera, L., Plane, J., & McGinnis, J. R. (2020). Teacher change following a professional development experience in integrating computational thinking into elementary science. *Journal of Science Education and Technology*, 29(1), 174–187. <https://doi.org/10.1007/s10956-019-09798-4>
- Lee, I., Martin, F., & Apone, K. (2014). Integrating computational thinking across the K-8 curriculum. *ACM Inroads*, 5(4), 64–71.
- Lee, V. R., & Recker, M. (2018). Paper circuits: A tangible, low threshold, low cost entry to computational thinking. *TechTrends*, 62(2), 197–203. <https://doi.org/10.1007/s11528-017-0248-3>
- Lu, J. J., & Fletcher, G. H. (2009). Thinking about computational thinking. In *Proceedings of the 40th ACM technical symposium on computer science education* (pp. 260–264). ACM.
- Massey, S. R. (2015). The multidimensionality of children's picture books for upper grades. *English Journal*, 104(5), 45–58.

- National Governors Association Center for Best Practices, Council of Chief State School Officers. (2010). *Common core state standards English language arts*. National Governors Association Center for Best Practices, Council of Chief State School Officers.
- NGSS Lead States. (2013). *Next generation science standards: For states, by states*. The National Academies Press.
- Palts, T., & Pedaste, M. (2020). A model for developing computational thinking skills. *Informatics in Education, 19*(1), 113–128.
- Patton, M. Q. (2002). *Qualitative research and evaluation methods* (3rd ed.). SAGE Publications.
- Selby, C., & Woollard, J. (2013). *Computational thinking: The developing definition*. <https://eprints.soton.ac.uk/356481/>
- Smith, M. (2016, January 30). *Computer science for all*. The White House Blog. <https://obamawhitehouse.archives.gov/blog/2016/01/30/computer-science-all>
- Sousa, D. A., & Tomlinson, C. A. (2018). *Differentiation and the brain: How neuroscience supports the learner-friendly classroom* (2nd ed.). Solution Tree Press.
- Sullivan, A., & Bers, M. U. (2016). Robotics in the early childhood classroom: Learning outcomes from an 8-week robotics curriculum in pre-kindergarten through second grade. *International Journal of Technology and Design Education, 26*(1), 3–20. <https://doi.org/10.1007/s10798-015-9304-5>
- Tomlinson, C. A., & Allan, S. D. (2000). *Leadership for differentiating schools and classrooms*. ASCD.
- Tran, Y. (2019). Computational thinking equity in elementary classrooms: What third-grade students know and can do. *Journal of Educational Computing Research, 57*(1), 3–31.
- Waterman, K. P., Goldsmith, L., & Pasquale, M. (2020). Integrating computational thinking into elementary science curriculum: An examination of activities that support students' computational thinking in the service of disciplinary learning. *Journal of Science Education and Technology, 29*(1), 53–64. <https://doi.org/10.1007/s10956-019-09801-y>
- Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35.
- Yadav, A., Hong, H., & Stephenson, C. (2016). Computational thinking for all: Pedagogical approaches to embedding 21st century problem solving in K-12 classrooms. *TechTrends, 60*(6), 565–568.
- Yadav, A., Krist, C., Good, J., & Caeli, E. N. (2018). Computational thinking in elementary classrooms: Measuring teacher understanding of computational ideas for teaching science. *Computer Science Education, 28*(4), 371–400. <https://doi.org/10.1080/08993408.2018.1560550>
- Yadav, A., Zhou, N., Mayfield, C., Hambruch, S., & Korb, J. T. (2011). *Introducing computational thinking in education courses* [Paper presentation]. The 42nd ACM Technical Symposium on computer science education (pp. 465–470). ACM.
- Yin, Y., Hadad, R., Tang, X., & Lin, Q. (2020). Improving and assessing computational thinking in maker activities: The integration with physics and engineering learning. *Journal of Science Education and Technology, 29*(2), 189–214. <https://doi.org/10.1007/s10956-019-09794-8>

Author Biographies

Evan Dave Ballard is a software developer specializing in data science and a physics and computer science teacher. He has taught courses in Java and Python and a summer course in general computer science. He is currently working on his Master of Science in Education at the University of Wisconsin-River Falls. He has previously published on computer science practices in children's literature with Dr. Rachelle Haroldson. He read many of the books discussed with his two young children.

Rachelle Haroldson is a clinical associate professor at the University of Wisconsin at River Falls and the Master Teacher of the STEMteach program. For the past five and a half years, she has taught, supervised, mentored, and advised six cohorts of graduates studying to become STEM secondary teachers. She actively integrates computer science into the one-year program with her teacher candidates, pushing all of them, regardless of content area, to see themselves as CS teachers and advocates. She has completed the Bootstrap Workshop, the College Board–endorsed UTeach Computer Science Principles Institute, and the Computer Science Education Certificate through the College of St. Scholastica. In the last four years she has presented or co-presented with graduate students on computer science–related topics at eleven difference conferences across the Midwest and nationally.