ISSN 2324-7878 (online)
palrap.org

PaLA College & Research Division (CRD)

Pennsylvania Libraries: Research & Practice

Research

Rethinking Computational Thinking for Public Libraries' Youth Programs

Challenges and Recommendations

Marissa Guidara

Marissa Guidara is District Library Youth Services Consultant at Reading Public Library, marissa.guidara@gmail.com

Computational thinking has become a popular and important concept in education throughout the nation. Public libraries, with their technology services and their role as an informal learning space, have been tagged as an ideal place for computational thinking learning for children. However, the literature and research surrounding computational thinking is often vague and even misleading, presenting differing visions of what computational thinking is, what it should look like in practice, and how it might be evaluated for effectiveness. As a result, youth services librarians face many challenges in their attempts to understand, design, and evaluate computational thinking programs for their libraries. This paper explores the issues inherent in current computational thinking research and discusses the challenges they represent in designing and facilitating youth computational thinking programs in public libraries, as well as presents recommendations for best practices.

Introduction

Computational thinking has become a popular term in recent years, a buzzword that just as easily finds its way into parenting magazine articles as it does academic research. It is being touted as a new "literacy" (Snelling, 2018) and as a "fundamental skill used by everyone in the world by the middle of the 21st century…as fundamental as reading writing and arithmetic. The third pillar of the scientific method" (Wing, 2016).

The notion of computational thinking as a fundamental skill for the future has not gone unnoticed by governments, educators, and big businesses who are looking for ways to prepare children for the technology and economy of the future. In 2016, President Obama announced the "Computer Science for All" initiative, which "identified STEAM (science, technology, engineering, and math) learning and computer science as national priorities for all age groups" (Prato, 2017, p. 19). A year later, Google announced $500,000 worth of grants for libraries interested in participating in its Libraries Ready to Code initiative (Braun & Visser, 2017).

Despite the call to arms for computational thinking education, schools are not flying the banner. According to Prato (2017), "nine in ten parents want their children to study computer science, but only one in four schools teach computer programming" (p. 21). It is public libraries and youth services librarians who are being asked to fill this gap because of their role as informal learning spaces (Martin, 2017), the free access they offer to technology and internet, and in turn, their already established commitment to bridge the digital divide (Braun & Visser, 2017). With so much money and attention being thrown in the direction of computational thinking, it is no wonder that libraries are attempting to take up the call to promote and facilitate these types of programs.

However, while many youth services librarians begin to develop computational thinking programs for their patrons, several major challenges arise with the very concept of computational thinking. While research and literature, library blogs and workshops, enthusiastically tell youth services librarians they should offer this, there is little consensus on what "this" is or what it looks like. Computational thinking literature "is at an early stage of maturity, and is far from either explaining what [computational thinking] is, or how to teach and assess this skill" (Kalelioglu, Gulbahar, & Kukul, 2016, p. 583).

As a whole, the body of computational thinking research contains varying definitions of computational thinking, makes claims about the benefits of computational thinking that have yet to be substantiated, and provides inconsistent visions of what computational thinking programs should look like. Basic questions like, "What is computational thinking?" and "What does it look like in practice?" do not have easy answers. This paper examines the challenges facing youth services librarians as a result of these issues inherent in computational thinking research and recommends best practices that consider the unique space of the public library for moving forward.

Definition(s): What is computational thinking?

Despite the popularity and excitement that surround computational thinking programs in public libraries, it is surprising to note that "there is not one unanimous definition of computational thinking" (Angeli, et al., 2016, p. 49). Similarly, Brennen and Resnick (2012) note that "there is little agreement on what a definition for computational thinking might encompass" (p. 2). Even the recently released Libraries Ready to Code web collection (2018) states that "there is no one single list of the specific concepts, practices, and dispositions included in [computational thinking]."

The very first challenge youth services librarians face when implementing computational thinking programs is immediate: What exactly is computational thinking? If there is little agreement on a definition and no single list of concepts to practice, what exactly are librarians meant to be teaching or introducing to their young patrons?

Computational thinking has been described as

- "defining, understanding, and solving problems; reasoning at multiple levels of abstraction; understanding and applying automation; and understanding the dimensions of scale" (Allan, et al., 2010, p. 1);
- "a creative process that includes understanding concepts, processes, and perspectives of a designer" (Brennan & Resnick, 2012, p. 2);
- "thought processes involved in formulating problems so their solutions can be represented as computational steps and algorithms" (Aho as cited in Tedre & Denning, 2016, p. 120);
- "an underlying set of skills foundational to computer science though also transferable to broader applications" (Braun & Visser, 2017, p. 8);
- "the way that computer scientists think, the manner in which they reason" (Riley & Hunt as cited in Kaleioglu, Gulbahar, & Kukul, 2016, p. 585);
- "a set of thinking skills that may not result in computer programming (Syslo & Kwiatkowska as cited in Kaleioglu, Gulbahar, & Kukul, 2016, p. 585);

- and "solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science" (Wing, 2006, p. 33).

In these examples alone, computational thinking is somehow simultaneously a thought process, a design process, a set of skills, a problem-solving method, and an application for automation, and may not necessarily result in computer programming at all. Many words used are computer science jargon such as *algorithm* or *abstraction*.

These definitions represent different visions of computational thinking and the computer science jargon used can be very daunting to anyone outside of the computer science field, including youth services librarians tasked with facilitating computational thinking for children. Tedre and Denning (2016) note this is a challenge for educators saying that the "multiple [computational thinking] visions, although inspiring and ambitious, do not agree on what exactly should be taught about [computational thinking], how to assess whether students have learned [computational thinking], and who are the main beneficiates of [computational thinking]" (p.120).

Library-Centered Definitions

Initially, the library profession had begun to address this problem of vague and various definitions for youth services librarians. In Phase II of Libraries Ready to Code, an initiative created from a partnership between Google and the American Library Association, library and information science professors were tasked with creating a new curriculum for master of library science students with a focus on computational thinking as a literacy. This would help close the skills gap that many youth services librarians point to as a reason for their reluctance to facilitate computational thinking programs (Braun & Visser, 2017). In doing so, professors created a definition that defined it within the context of a library:

> "[Computational thinking] within the context of libraries: youth learn through opportunities to uncover everyday personal or community problems and develop strategies to solve those problems. Through these problem-definition, data gathering, and problem-solving steps, they are mentored and coached by library staff, peers, and community members, and participate in activities that lead to an understanding of their community and the world around them and the ability to analyze, synthesize, produce, and organize information. Through these activities, they build decomposition, pattern recognition, algorithms, abstraction, and automation skills" (Drouillard, 2017).

This was a good attempt to situate this concept into the realm of libraries. However, by the time the Libraries Ready to Code resource collection was released in June 2018, this definition was gone, replaced with a short description of computational thinking and links to outside sources: to the classic 2006 Wing article, a Google-sponsored definition, a "leadership guide," and a teacher association definition (Libraries Ready to Code, 2018), none of them attempting to define computational thinking in terms of a public library. This is a huge misstep. Creating a definition that is sound, specific, and aligns with public library standards and practices is necessary to overcome the obstacles of a vague definition.

As Denning (2017) would argue, this vagueness is the point. Denning believes that "in attempts to appeal to other fields besides [computer science], [researchers] offered vague and confusing definitions of computational thinking…the claims that it benefits everyone beyond computational designers are as yet unsubstantiated" (Denning, 2017, p. 33). Computational thinking was purposely positioned to become a catchall term that seemingly applied to any field or situation in an effort to make it more appealing outside of the computational science field.

Origins of Modern Computational Thinking

Despite their differences, most definitions harken back to or were influenced by Wing's pivotal 2006 article on computational thinking. Wing's essay is considered an origin point for educators and librarians because she

manages to accomplish two things: She situates computational thinking outside the computer science field, and situates computational thinking outside of computers themselves.

First, Wing (2006) claims that "computational thinking is a fundamental skill for everyone, not just for computer scientists. To reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability" (p. 33). That notion, that computational thinking, and therefore coding and computer science, are fundamental to everyone was novel in 2006 and is what myriad researchers and educators have pointed towards when justifying the need for computational thinking in classrooms, public libraries, and other youth programs. Later, she added that "someone with the ability to use computation effectively would have an edge over someone without" (Wing, 2016). This is the origin of computational thinking as a new literacy that is fundamental to everyone and the impetus for many educators and librarians outside of the computer science field to take up the task of teaching it.

Second, Wing goes so far as to explain that computational thinking is a tool used in everyday life, removing the notion of computers or computational models completely:

> "When your daughter goes to school in the morning, she puts in her backpack the things she needs for the day; that's prefetching and caching…At what point do you stop renting skis and buy yourself a pair: that's online algorithms. Which line do you stand in at the supermarket?; that's performance modeling for multi-server systems" (Wing, 2006, p. 34).

Wing uses computational thinking outside the bounds of computer science or even computer use, claiming it as a fundamental skill for everyday life. It is little wonder why that belief is so appealing to educators and librarians. Wanting to be supportive of computational thinking but lacking computer science knowledge, librarians can use Wing's "everyday life" approach to create computational thinking programs, developing programs around everyday challenges like packing a bag. However, is this computational thinking?

Not all researchers think so. The claims that computational thinking benefits fields outside of computer science or in everyday life are unsubstantiated (Denning, 2017). Denning asks, "Is it really true that any sequence of steps is an algorithm? That procedures of daily life are algorithms?" (Denning, 2017, p. 34).

Some researchers in the computer science field are uncomfortable with a computational thinking definition that removes computational models. To some, computational thinking requires the computational model or automation. The absence of any mention of computational models in Wing's definition is considered a "mistake… we engage with abstraction decomposition, data representation, and so forth, in order to get a model to accomplish certain work" (Denning, 2017, p. 36). The point of computational thinking is specifically to work with these models.

Similarly, Guzdial (2015) recounts pushback against computational thinking at a European workshop. "The educational psychologists thought it was unbelievable that learning computing would in any way impact the way that people think or problem-solve in everyday life. Didn't we believe that once about Latin?" (Guzdial, 2015). This comparison between computational thinking and the old outdated practice of teaching children Latin is a testament to how unconvinced some researchers and educators are as to the claims that it can be valuable in everyday life or in fields outside of computer science.

The Forgotten History of Computational Thinking

To further complicate matters, the myriad researchers and educators who credit Wing with coining the definition of computational thinking are incorrect. Wing's article and definition certainly stirred a renewed interest in computational thinking in 2006, but the term itself dates back to 1980 when Papret used the term in his book *Mindstorms: Children, Computers, and Powerful Ideas* (which the LEGO coding tool is named after), to describe a "mental skill children develop from practicing programming" (Denning, 2017, p.35).

In fact, the mainstream push to develop computational thinking programs in educational institutions already happened in the 1980s and 1990s. Much of this research is largely forgotten among the current cultural conversation and research. And perhaps with good reason—many researchers didn't find evidence that teaching computational thinking to students was particularly successful (Kafai, 2016). These early attempts at computational thinking in classrooms "did not deliver on promises. By the mid-1990s most schools turned away from programming" (Kafai, 2016, p. 26). A lack of historical context forces educators, youth services librarians, and researchers themselves to reinvent the wheel and could lead to watered-down versions of these preceding programs (Tedre & Denning, 2016).

Programming: Is this computational thinking?

The various visions of and definitions for computational thinking in libraries means there isn't one "right" way to offer computational thinking skills. The lack of direction mixed with Wing's "everyday life" approach has resulted in an "anything goes" style of programming. From creating games that solve social ills (Martin, 2017) to simply making guacamole with your family (Libraries Ready to Code, 2018), youth services librarians are creating programs based on their own interpretations of computational thinking definitions and their skillsets (or lack thereof). This had led to a heavy reliance on specialized tools and websites (Braun & Visser, 2017, p. 18), as well as the popularity of "unplugged" computational thinking programs, both of which have researchers questioning their validity as computational thinking programs.

Reliance on Tools

Youth services librarians who lack the skill set, but still want to facilitate computational thinking, have heavily relied on popular block coding websites and robotics toys that promise to do the computational thinking facilitating for them (Braun & Visser, 2017).

Websites such as Hour of Code and Scratch use simple block code, a visual programming language, where children drag and drop puzzle pieces into the desired order (an algorithm) to move a character across the screen. They are considered easy for both children to use and librarians to learn, but some researchers believe they oversimplify "coding and thus using these tools detracts from the ability of learning to grasp complex programming concepts" (Braun & Visser, 2017, p. 35). These visual programming languages bear little resemblance to the programming languages used in professional settings.

Other popular tools are robots, such as Bee-bots and Cubettos. Robots serve as a visual expression of computational thinking (Prato, 2017), have an obvious coolness factor (Braun & Visser, 2017), and as Duill (2012) describes, feed the "humanoid robotic fantasies children derive from entertainment media" (p. 96).

Some of these robots use an app which helps to demystify computational thinking and coding skills by blending the digital and physical world (Braun & Visser, 2017, p. 23), helping children see how their computational thinking efforts affect objects in front of them. Dash and Dot robots, for instance, use a block code app so children are working on a screen (the digital world) and seeing their robot move in front of them in the real world.

Some robots are considered "tangible tech" such as Bee-bots and Cubettos, which do not use a computer or app to code; no screen is necessary. Instead, the robot features a series of buttons or manipulating pieces that are put into an algorithm. A child presses a button, and the robot completes the sequence in the code.

While these tools make life easier for youth services librarians and may help attract children to programs, the focus on these tools can be problematic, according to Braun and Visser (2017). This focus "shows that library staff are less likely to acknowledge the learning accomplished through these activities than the tools that are used in the learning" (Braun & Visser, 2017, p. 18). The novelty of the toys takes precedence over learning outcomes.

Similarly, Duill (2012) notes that toys such as the Bee-Bots are "cognitively a 'black box' with buttons…their main use is in teaching left/right turns and distance estimation, which are more simply and economically incorporated in pencil and paper mazes" (p. 96). Duill believes that because such tools are closed to creation, they offer very little computational thinking value.

Both Duill (2012) and Braun and Visser (2017) call for a more constructionist approach, where children create an object or project. However, they fail to recognize that the need for such tools in the first place is that they offer librarians an easy way to introduce computational thinking skills. Criticism that such programs are mere gateways (Braun & Visser, 2017) place youth services librarians in a dilemma with their programming: They are encouraged to use these tools because they don't know enough about computational thinking (Prato, 2017), but the critics claim the programs aren't enough (Duill 2012; Braun & Visser, 2017; Denning, 2017).

Unplugged Programs

Unplugged activities take their lead from Wing's (2006) notion that computational thinking can be used in everyday life situations. These programs include no technology or computers and often focus on pattern recognition, solving problems through challenges, or creating step-by-step instructions. They serve as a "low-stakes entry point for library staff just learning how to integrate computational thinking activities" (Braun & Visser, p. 18). Most of these unplugged activities take the form of algorithm design. For instance, the Libraries Ready to Code website (2018) describes a computational thinking program where families are challenged to "create their own recipe for guacamole…they are finding patterns in the recipe and abstracting ideas to craft their own variations of the original. The step-by-step recipes they create are similar in flow to the algorithms a computer follows" (Haines, 2018).

This everyday life algorithm is reminiscent of Wing's (2006) assertion that computational thinking could be used to pack a bag or choose a line at the grocery store. However, Denning (2017) asserts that a computational model is necessary for an algorithm. An algorithm "is not any sequence of steps, but a series of steps that control some abstract machine or computational model without requiring human judgement. Computational thinking includes designing the model, not just the steps to control it" (Denning, 2017, p. 33). An everyday life algorithm, such as making guacamole, requires human judgement, which has never been considered a part of an algorithmic step.

Are these two program examples, a reliance on a tool such as a black-box robot and making an algorithm out of a recipe, computational thinking? Yes and no, depending on which research one subscribes to. A definitive answer is impossible without a widely agreed-upon definition to draw from. While library blogs and workshops extol the ease of use of such tools, the academic literature criticizes them for not being effective or impactful enough, creating a major conflict and challenge for youth services librarians.

Evaluations: Is this working?

As we've seen, the vague and varied definitions of computational thinking have led to challenges in developing computational thinking programs for public libraries. It is little surprise then to find that these same issues have left librarians unsure of how to assess whether or not their programs have been successful (Denning, 2017). To complicate matters further, the public library's role as an informal learning environment also acts as a barrier towards traditional modes of assessment.

The literature on computational thinking offers very little in reference to assessments. Benitti found that out of more than 70 papers on computational thinking, only 10 provided measurements of computational thinking in programs (cited in Toh, Causo, Tzuo, Chen & Yeo, 2016, p. 148).

Skill vs. Knowledge

First, the various definitions of computational thinking refer to it as both a "concept" and a "skill." These terms are not interchangeable. How a librarian would evaluate a concept or knowledge is much different than how one might evaluate a skill. For example, Sullivan & Bers (2015), used a one-on-one style assessment on students in preschool through second grade who received training with a KIWI robot. Working with a researcher, they were tested for skill recall. Researchers asked, "What part should I use if I want my robot to turn its light on?" and the child would receive a point for correctly identifying five different robot parts, including three sensors, a light output, and the motor. It was reported that the youngest students "struggled with mastery of the robotic parts" (p. 17) and were not successful on this assessment. Does failing to learn this specific skill of pressing a sensor mean that the children did not practice or learn computational thinking concepts, such as abstraction or decomposition? The assessment isn't able to test for that.

Likewise, evaluations that test for knowledge can give equally dubious results. In Brennan & Resnick (2012), measurement was conducted via product-based assessments. Researchers looked at Scratch coding projects created by children ages 8-17 and asked questions about their experiences. In one example, an interviewer asked a student, "How does this work?" The student was unable to explain any part of it, revealing that he had seen a similar project on a website and copied the code. While this student lacks the knowledge to explain how his project works, it could still be argued that he used computational thinking skills by altering the block in the project that he had taken from the internet.

Traditional Library Models of Assessment

Ironically, the informal learning setting, one of the very elements that makes public libraries such an ideal computational thinking environment (Braun & Visser, 2017), also makes it difficult to evaluate and assess computational thinking programs for success. Youth services librarians do not grade, and are not beholden to education standards or tests. Children can practice computational thinking "without the pressure of failing…you are not being graded in any way, so it is a safe environment for [patrons] to experiment" (Suramaniam cited in Snelling, 2018). But this also means youth services librarians do not have tools to evaluate whether or not their programs are effectively facilitating computational thinking.

In fact, many library programs are not assessed for effectiveness. Many public libraries used output-based assessments, relying on participation numbers to determine whether or not a program was successful (Matthews, 2010). In recent years, many libraries have moved towards an outcome-based assessment, relying on surveys and focus groups that ask participants to comment on the content or the goals of the program itself (Cole, Walter, & Mitnick, 2013). In the former method, 50 children at a program may be deemed a great success, but it would be unclear whether or not the program was effective in relaying any educational elements. In the latter, only 5 kids may come to a program, but if one child commented that the program had value or that he met his personal goal for the program, then that program could be considered a great success even though the participation was low.

Considering the lack of much of a prior history of program evaluation, the assessment limitations of an informal learning environment, and the fact that researchers are unsure of how to assess computational thinking, youth services librarians again face another obstacle of determining whether their programming is successful in facilitating computational thinking.

Recommendations

A key element in facilitating successful and effective youth computational thinking programs in public libraries is framing them squarely within the context of the public library itself. While there may not be one widely

accepted definition, method of execution, or assessment strategy (and any that might be created would most likely be done for the benefit of schools, and not the unique setting of a public library), an authoritative body such as the American Library Association, the Association of Library Services to Children, or the Libraries Ready to Code initiative could address these issues by creating their own definition framed around public libraries as a base on which youth services librarians could confidently build their programming. The fact that the Libraries Ready to Code initiative shied away from interpreting a computational thinking definition in favor of pointing to several others, none of which mention public libraries, is a big misstep.

Further, public libraries' foray into computational thinking should play to the libraries' strengths. It is the unique aspects of a public library that make it such an ideal place for computational thinking programs (Braun & Visser, 2017, p. 2). This includes its existing technology efforts, informal learning environments, its collections, and its already-successful programs.

Technology and The Digital Divide

A definition of computational thinking for public libraries must include the mention of a computational model (Denning, 2017). Technology should be utilized when possible in programs. First, if computational thinking is understood to be an "underlying set of skills" (Braun & Visser, 2017, p. 4) or foundation for coding, robotics, and computer science, then it can be believed that practicing with computational models and tools should also mean practicing computational thinking. By creating a definition that focuses on these skills, the thinking concepts behind computational thinking *may* naturally follow.

Furthermore, while the "everyday life" approach makes things easier for youth services librarians, it fails to tackle the digital divide, which is one of the other reasons public libraries were identified as an ideal place for computational thinking (Braun & Visser, 2017). Public libraries are no strangers to the term "digital divide," which describes the unequal access to computer and technology (Reich & Ito, 2017). Teaching computational thinking, introducing children to the tools and programs used, and teaching the skills necessary for computer science tackles the digital divide by offering equitable access to the tools, ideas, and concepts necessary to create technology, ensuring that all voices, especially those who are underrepresented in the computer science fields, can be heard.

Despite the criticism that tech tools like robotic toys and coding game websites are not effective at teaching computational thinking (Braun & Visser, 2017; Duill, 2012), youth services librarians should make use of them, especially as they themselves are learning how to facilitate these new types of programs. It also introduces these technology pieces to children who, in the case of underserved children including those who are underrepresented in computer science fields, may not see them elsewhere, acting as another deterrent to the digital divide.

Utilizing technology and computers in computational thinking programs, of course, means youth services librarians will have to learn to become more comfortable with technology, coding, and robotics. Libraries Ready to Code Phase II made some great efforts in this area by designing a curriculum for computational thinking that would be taught to prospective librarians during their master of library science studies. This way, computational thinking would be embedded into librarianship itself, establishing it as an essential concept or skill for the profession. Hopefully, such curriculum will not only provide the "what" and "why" of computational thinking, which, as can be seen from current research, can get tricky, but also provide training on the basic fundamentals of computational thinking and introduce practical skills such as coding with program languages. Youth services librarians do not need to be persuaded to teach patrons computational thinking by providing numerous reasons on why it is important; they simply need training on all of the computer science concepts that computational thinking should be leading towards.

Similarly, national and state library associations should consider offering similar workshops for youth services librarians that focus on practical technology skills that can be used to facilitate computational thinking

programs. Youth services librarians will also need the support of their individual libraries, who must find the time and budget necessary for librarians to learn, design, and facilitate these new types of programs.

Outcome Based Assessments

As an informal learning environment, public libraries have often struggled with creating assessments for an environment where experimenting and failing are celebrated (Snelling, 2018). As libraries move towards outcome-based evaluations in other areas, these same types of evaluations may help create successful stories to share about programming. Outcome-based evaluations offer individual assessments by encouraging children to make goals, and help libraries determine the impact of their programming by collecting stories and anecdotes about the value of the program.

Building on Library Strengths

While much of the research regarding computational thinking focuses on the notion of problem solving, research fails to think about how to initiate and inspire solutions to these problems. Libraries have the key to that solution: books.

In 2007, Newbery medal winner Neil Gaiman visited China for its first-ever state-sponsored science fiction convention. When asked why the government was supporting the convention, a party official revealed that it was actually a well-researched initiative towards technological innovation. Frustrated with their tech industry's lack of original ideas, the Chinese government sent researchers to tour facilities at Microsoft, Google, and Apple. What they found was surprising: many of the employees coming up with innovative new ideas had a deep love for science fiction (Gaiman, 2016). Elon Musk, Jeff Bezos, Mark Zuckerberg, among other famous names in computer science also count themselves as huge science fiction fans. In fact, product managers at Facebook were required to read Neal Stephenson's 1992 novel *Snow Crash*, as it was used by Ben Narasin to develop theories about the commercialization of the early internet (Shenoy, 2016).

Working with books is within the realm of even the least tech-savvy of youth services librarians. Encouraging thought and discussion while using science fiction books like the ones that inspired our current computer science celebrities, will help breed innovation and encourage the "feedback loop between science fiction and technological fact" (Shenoy, 2016) in the future.

In addition, already established successful programs such as LEGO Clubs or summer learning activity sheets could be adapted to add elements of computational thinking into them. Challenging builders at a Block Party program to build a house for a Bee-bot that must drive into his new house, or adding a game on the Hour of Code website as an activity option on their summer learning logs, will give youth services librarians a sense of comfort and confidence. Working within a familiar framework they know is already successful will give librarians a sense of ownership over the project, and encourage them to continue to learn and build on their successes.

Finally, more research needs to be done pertaining to computational thinking specifically in a public library setting. The majority of research is done for classrooms, a strange irony given the fact that public libraries were tapped to do computational thinking because only 40% of public schools offer computational thinking programs (Prato, 2017). Research specifically targeting public libraries would ensure that the methods that rise as best practices are best practices for public libraries and not adapted schoolroom methods.

Conclusion

The state of computational thinking research is still new and rather messy. Without an official definition of computational thinking from the computer science field, various visions of what computational thinking is and what it should look like for youth programming have emerged. With no foundation to base programs and methods on and no clear vision of what outcomes should be, youth librarians have faced several challenges in creating, implementing, and assessing computational thinking programs.

Ultimately, computational thinking for public libraries should be designed around their unique informal learning environment and existing technology services. An authoritative body such as the American Library Association or the Libraries Ready to Code initiative should help create this definition and framework so that youth services librarians can use it as a foundation for their own programs.

Further, youth services librarians should create computational thinking programs around their strengths, such as their collections that may inspire innovation or their already-successful programming. This will give youth services librarians ownership of their computational thinking programs and encourage them to learn more about computational thinking.

References

Allan, W., Coulter, B., Denner, J., Erickson, J., Lee, I., Malyn-Smith, J., & Martin, F. (2010). *Computational thinking for youth*. Retrieved from stelar.edc.org/sites/stelar.edc.org/files/Computational_Thinking_paper.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. *Journal of Educational Technology & Society, 19*(3), 47-57. Retrieved from www.j-ets.net/ETS/journals/19_3/6.pdf

Braun, L. & Visser, M. (2017). Ready to code: Connecting youth to CS opportunity through libraries. *Libraries Ready to Code*. Retrieved from www.ala.org/advocacy/sites/ala.org.advocacy/files/content/pp/Ready_To_Code_Report_FINAL.pdf

Brennan, K. & Resnick, M. (2012). *New frameworks for studying and assessing the development of computational thinking*. Boston, MA: American Educational Research Association. Retrieved from scholar.harvard.edu/kbrennan/publications/new-frameworks-studying-and-assessing-development-computational-thinking

Cole, N., Walter, V., & Mitnick, E. (2013). Outcomes + outreach. *Public Libraries*, *52*(2), 38-43. Retrieved from publiclibrariesonline.org/2013/05/outcomes

Denning, P. (2017). Remaining trouble spots with computational thinking. *Communications of the ACM, (60)*6, 33-29. doi:10.1145/2998438

Drouillard, C. (2017). Lecture on computational thinking. Personal Collection of C. Drouillard, Valdosta State University, Valdosta, GA.

Duill, M. (2012, September). Reversing robotic regression: Why our culture rejects robotics in school. In D. Obdrzalek [Ed.], *RiE 2012: 3rd international conference on robotics in education.* Prague, Czech Republic: MatfyzPress. Retrieved from www.ksi.mff.cuni.cz/rie2012/proceedings/2012RiE-13.pdf

Gaiman, N. (2016). *The view from the cheap seats*. New York, NY: HarperCollins

Guzdial, M. (2015, July 13). Growing CS ED through schools of ED, and CT is unlikely: Report from Oldenburg [Blog post]. Retrieved from computinged.wordpress.com/2015/07/13/growing-cs-ed-through-schools-of-ed-report-from-oldenburg

Kafai, Y. (2016). From computational thinking to computational participation in K-12 education. *Communications of the ACM, 59*(8). doi:10.1145/2955114

Kalelioglu, F., Gulbahar, Y., & Kukul, V. (2016). A framework for computational thinking based on a systematic research review. *Baltic Journal of Modern Computing, 4*(3), 583-596. Retrieved from www.bjmc.lu.lv/contents/vol-42016-no-3

Libraries Ready to Code (2018). The what and why of computational thinking. Retrieved from www.ala.org/tools/readytocode/computational-thinking

Martin, C. (2017) Libraries as facilitators of coding for all. *Knowledge Quest, 45*(3), 46-53. Retrieved from files.eric.ed.gov/fulltext/EJ1125376.pdf

Matthews, J. (2010). Evaluating summer reading programs. *Public Libraries*, *49*(4), 34-40. Retrieved from publiclibrariesonline.org/2013/05/evaluating-summer-reading-programs-suggested-improvements

Prato, S. (2017). Beyond the computer age: A best practices intro for implementing library coding programs. *Children and Libraries*, *15*(1), 19-21. Retrieved from journals.ala.org/index.php/cal/article/view/6238

Reich, J. & Ito, M. (2017) *From good intentions to real outcomes: Equity by design in learning technologies*. Irvine, CA: Digital Media and Learning Research Hub. Retrieved from clalliance.org/wp-content/uploads/2017/11/GIROreport_1031.pdf

Shenoy, G. (2016, October 21). India needs to read more science fiction to imagine and create. *Factor Daily*. Retrieved from factordaily.com/india-read-more-scifi-imagine-create

Snelling, J. (2018). Don't stress about coding: Focus shifts to teaching problem solving not computer skills. *School Library Journal*. Retrieved from www.slj.com/?detailStory=dont-stress-coding-focus-shifts-teaching-problem-solving-not-computer-skills

Sullivan, A. & Bers, M. U. (2015). Robotics in the early childhood classroom: Learning outcomes from an 8-week robotics curriculum in pre-kindergarten through second grade. *International Journal of Technology & Design Education*, *26*(1), 3-20. doi:10.1007/s10798-015-9304-5

Tedre, M. & Denning, P. (2016, November). The long quest for computational thinking. In *Proceedings: 16th Koli Calling International Conference on Computing Education Research*. Koli, Finland: ACM. Retrieved from denninginstitute.com/pjd/PUBS/long-quest-ct.pdf

Toh, L., Causo, A., Tzuo, P., Chen, I., Yeo, S. (2016) A review on the use of robots in education and youth children. *Journal of Educational Technology & Society, 19*(2), 148-163. Retrieved from www.j-ets.net/ETS/journals/19_2/12.pdf

Wing, J. (2006). Computational thinking. *Communications of the ACM, 49*(3), 22-25. doi:10.1145/1118178.1118215

Wing, J. (2016, March 23). Ten years later. [blog post]. Retrieved from www.microsoft.com/en-us/research/blog/computational-thinking-10-years-later