Computational Thinking should just be Good Thinking

by Mark Guzdial, Alan Kay, Cathie Norris, and Elliot Soloway[1]

Jeannette Wing's 2006 paper on *Computational Thinking* ignited a worldwide movement to give students' new knowledge and skills to solve problems in their daily lives. Quickly, teachers, curriculum and standards writers, and other education specialists were proposing what children needed to know about computation and how to develop a computational mindset. There is still little evidence that knowing about computation improves everyday problem-solving, but there is no doubt that Wing's call to action led to a broad and dramatic response.

The computational thinking movement puts the onus on the student and on the education system. They argue that if we change *humans* to think in ways that are informed by how we now work with *computers*, that will have problem-solving advantages for the humans.

Maybe.

If a city doesn't work for the residents, we could change the residents. Alternatively, we could re-design the city. The best urban re-design has citizens understanding the purpose and actively participating, so that there is parallel development of both the city and the citizens.

Children today *already* think with computation. If we want better thinking and problem-solving, we have to improve the computing and use that to change our teaching. We put the onus back on the computer scientists and other computationalists. It's our job to design better.

For our Children, Computational Thinking is Just Thinking

Tool use shapes thinking. While we might not think like a carpenter when we start using carpentry tools, if we apply ourselves (e.g., reflect on our doing, as Dewey suggests), we can

[1] The lead author organized the effort, and the rest of the authors are in alphabetical order.

develop carpentry thinking. We can learn to see what is possible with the tools of carpentry, the way that a carpenter thinks.

Closer to home, the “kids these days” use all manner of digital – read: computational – tools. Hmm. Before drawing the obvious conclusion, consider the following vignettes:

Vignette 1: Consider the following two problems, drawn from a research study:

- (Algebraic Context): Given the following statement: "There are six times as many students as professors at this university." Write an equation that represents the above statement. Use S for the number of students and P for the number of professors.
- (Computer Programming Context): Given the following statement: "There are six times as many students as professors at this university." Write a computer program that will output the number of students when supplied (via user input) with the number of professors. Use S for the number of students and P for the number of professors.

While the equation in both problems is the same – $S=6P$ – significantly more undergraduate engineering students provided the correct equation in the Computer Programing Context than in the Algebraic Context.

Vignette 2: Now, consider the following research finding:

- "[K-12] Students using word processors for writing generally produce longer, higher-quality writing than students using pencil or pen and paper." The computational tool plays a role in students’ ability to write. We might say that using professional writing tools leads to performance that’s more like a professional writer. It’s honest use of the real thing.

Vignette 3: Now, consider the following comment:

- TikTok is the MOST downloaded app on the Apple App Store. TikTok supports users in making videos, including videos that play in synchrony with other user videos. Video producers collaborate around the world to make duets, without ever meeting. Using TikTok isn't about *writing* like a professional. TikTok is an entirely new medium, enabled by computation. It leads to writing and saying *differently* than one could *without* computation.

Vignette 4: Finally consider the following.

- Fortnite is one of the most successful video games of all time. In playing Fortnite, players use a broad range of computational tools to solve significant problems, from map navigation, to team collaboration, to managing complex ecological systems. Few children get the opportunity to engage in these kinds of activities in their everyday world outside of the computer. The computational environment allows students to *engage* with complex and interesting problems. We can ask if these are honest versions of the problems, if students have deep understanding of what they're doing, and if they are developing skills for the real world — and we *should* ask those questions.

The activity in Vignette 1 aligns with the notion that computational thinking is embodied in computer programming. Vignette 2 shows us that it's not just programming that can impact thinking. A wide variety of computational activities can impact thinking. In Vignettes 3 and 4, we argue that these activities illustrate "computational thinking" – though the activities in those vignettes have nothing to do with computer programming.

The users in those Vignettes are using computational tools to do computational thinking. They are using abstraction and decomposing problems, though they may not use those words. Much of the effort to implement computational thinking in schools has been about identifying the computing *ideas* and *practices*. Maybe the kids are already learning those, but on different terms, without our language.

Baby Boomers may feel that computational thinking is something special – and for them, it may well be. For the children growing up today, who are increasingly using digital tools to mediate their everyday lives, computational thinking is, well, just thinking! But that's just not enough. Learning to compute should give students a qualitative leap, so that they can think about new problems and think about the world in new ways.

How do we prepare our children for never-seen-before problems? We might start by re-designing TikTok and Fortnite.

Whether and Wither "Computational Thinking"

We already use computers to help many kinds of thinking, but much of that thinking would be the same without computers. We might get expanded thinking if we follow along the lines of extending mathematics and systems organizations to model complex situations that go beyond our commonsense reasoning, as seen in many scientific, engineering, medical, mathematical, and literary fields. Computing simulations has already revolutionized many fields. We might significantly impact society if all fields used this expanded thinking. So: there is a bird to be caught if we can sprinkle salt on its tail.

A strong rubric is "making systems about systems," and this accords well with the first Turing Award winner Alan Perlis' characterization of our field as "The science of processes; all processes." A subset of these processes are primarily algorithmic in nature, but to deal with the large range that computation can model, it is much more apt to "think all systems" and to see that the representational possibilities of the computer make it a great fit to be *the dynamic mathematics needed to make and understand systems*.

This is a much larger — and in our opinion — much more useful characterization of computing as a subject in K-12, and it leads to a number of important differences from current practice. The big one is to help children learn about dynamic systems with interacting parts of all

kinds, and how to make and model dynamic systems for deeper understanding (and considerable fun also!) Imagine something as engaging as Fortnite where the system is inspectable, where users might model their strategies and test them in simulation first, so that students might learn to use the power of expanded thinking.

A modeling and simulation point of view also serves to criticize the languages being taught today. For example, none of the common K-12 programming languages today are very good at modeling inter-communicating processes — despite both natural and human engineered systems working that way. Most of the languages that we put in school today can only handle one thread of control without ungraceful excursions into fragile and tricky designs. We shouldn't teach a qualitatively weak subset of something to children when we have better options. It might make later learning of a more powerful version more difficult.

Instead we should take our inspirations and goals from Jerome Bruner's assertion and challenge: "Any subject can be taught to any one at any age in an intellectually honest fashion if their level of development is heeded." Keeping the "intellectually honest" part means that — especially for young children — it will be necessary to invent real variants of adult versions of the subject matter — as has indeed been done so well by Montessori, Papert, Bruner, and others. We imagine a comprehensive suite of intellectually honest computing-based models for understanding systems can lead to much better notion of programming — for both adults and children. These will lead to much better programming language designs and environments as part of a larger curriculum made from the most powerful ideas about systems, processes, science, math, engineering, and computing itself.

One of the main ideas of K-12 schooling is to prepare children in general for their next phases of life, and subjects such as reading/writing/literature, science, mathematics, and history are taught to all to provide a "richness" of thought about both civilizations and how to be a citizen who supports civilization. Understanding civilization as a system is a powerful idea for

all citizens. In our metaphor, we want citizens to participate in the re-design of the city and understand the rationale for its design. Students need fluency in order to be able to understand models and systems. Important thresholds of understanding have to be reached before they can be part of one's thinking tools. Finding and inventing these thresholds for the general population of children, and how to teach to them, is the critical need of our time!

Representations to help thinking — language, mathematics, computing — are all best taught in context. Children should use computing with all the other fields of thought, rather than mostly in isolation. Rather than teach computer science as a separate topic that *might* transfer, we should teach *with* computational models in every field.

Conclusion: Montessori's Fortnite.

We can and should improve schools to give students access to expanded thinking. We in computing have a powerful lever. We can change the computation.

Maria Montessori made the observation almost one hundred years ago: children are set up by their nature to learn their surrounding environment and culture. Changing the environment naturally leads to different learning. Montessori wanted her children to have qualitatively different thinking, so she invented new kinds of school.

Changing school today impacts only one part of today's children's lives. Changing computing impacts their environment both in and out of school. If Montessori were alive today, she would still want to re-design school, but she would likely want to change the computing, too. That's part of the child's whole environment. How would Montessori *re-design* Fortnite? What would she design *instead* of Fortnite?

Teaching computing as it is today is unlikely to have dramatic impact on students' everyday lives. It's our job to re-design computing, to give children new power to make sense of their world and change it.