DOI:10.1145/3319081

Emmanuel Schanzer, Shriram Krishnamurthi, and Kathi Fisler

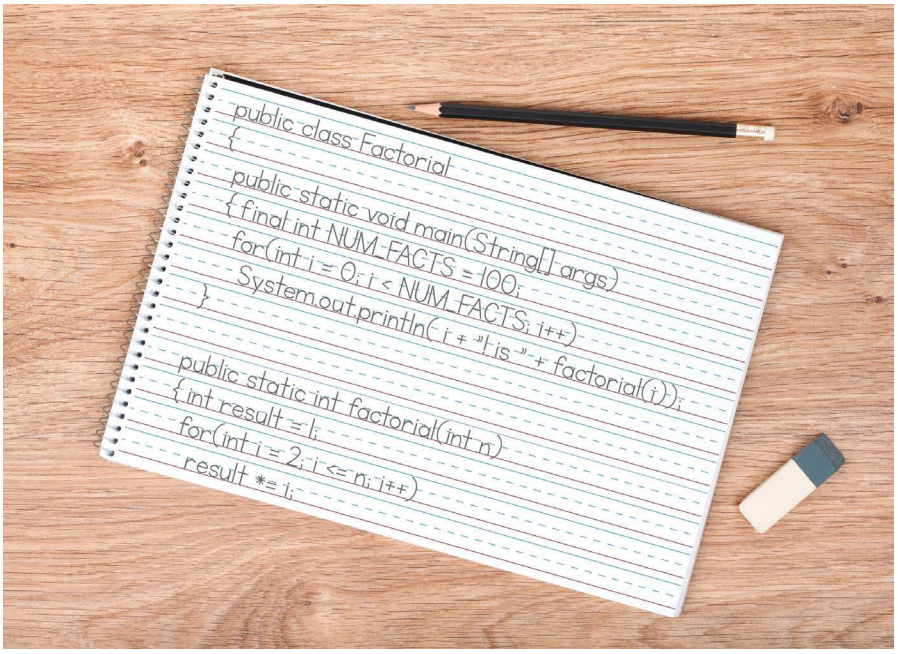
▸ **Mark Guzdial,** Column Editor

Education

What Does It Mean for a Computing Curriculum to Succeed?

Examining the expansion, proliferation, and integration of computing education everywhere.

COMPUTING EDUCATION IS suddenly everywhere. Numerous U.S. states and many countries around the world are creating requirements and implementing programs to bring computing to their students. Tech innovators have jumped in, too, sometimes to "disrupt" the educational system. Opinion pieces create parental anxiety that their children are not being trained properly for the future; products claim to mollify these anxieties (while perhaps simultaneously amplifying them). Academics, looking to address the Broader Impact criteria of funding agencies, are eager to burnish their credentials by giving guest lectures at local schools. In certain neighborhoods, toystores feel compelled to stock a few products that claim to enhance "computational thinking."[3]



Unfortunately, a lot of current discussion about curricula is caught up in *channels* (including in-school versus after-school courses), *media* (such as blended versus online learning), and *content* (for example, Java versus Python). As computer scientists, we should recognize this phenomenon: a focus on *implementation before specification*. Instead, in sober moments, we should step back and ask what the end goals are for this flurry of activity. Is a little exposure good for everyone? How many Hours of Code will prepare a child for a digital future? If a few requirements are good, are more requirements better? In short: What does it mean for computing education to succeed?

Specification: Three Worthy Goals

Every program would benefit first from a clear articulation of its goals. These goals should be as close as possible to concrete and measurable (and hence go significantly beyond anodyne phrases). We believe a truly ambitious project would have the trio of goals depicted in the figure in this column.

Readers might wonder if this is a "pick two" situation (or even a "pick one"). Indeed, dropping one or more of these demands greatly simplifies curriculum design, but with worrisome consequences. One can, for instance, obtain massive scale with a very simplistic curriculum (of which we see a good deal of evidence right now), with a focus on "engagement" but little to no rigor. A few high schools already have very rigorous computing curricula (students take several years of computing, reaching material well beyond the first year of college), but these are extremely difficult to scale. Elective classes can be very rigorous, but can easily lose equity: self-selection easily creates a vicious cycle that reinforces existing biases. Expensive curricula (especially

IMAGE BY ANDRIJ BORYS ASSOCIATES. USING PHOTO BY EVGENY KARANDAEV

involving physical devices—such as fancy robots and sensors—that must be bought *and repaired*) are very difficult to scale. Trying to pair teachers with working computing professionals may work fabulously in large cities with a big tech population, but would not scale to most rural areas.

Clearly, the outcomes of compromising are undesirable. Not compromising is, indeed, an intellectual and moral imperative:

- *Equity* is severely lacking in computing. Large-scale curricula with massive investment that ignore equity can only make the problem much worse.
- *Rigor* is critical to impart content of value. In its absence, we get the light entertainment that passes for many computing curricula today.
- *Scale* is essential to get computing into the hands of all of today's students who might be tomorrow's users, creators, or even victims of it.

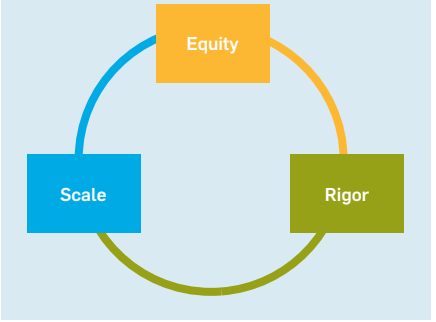
Rather than lay out how their implementation will address Equity, Rigor, and Scale (or other equally worthy goals), many of the players in this space are quick to use the rhetoric of "disruption" to gloss over the challenges outlined in this column. This is not altogether surprising, as many of them share a cultural heritage with (and often financial backing from) a tech industry that is infatuated with the term. Without question, some form of "disruption" is sorely needed—we do, after all, want a much larger and vastly more diverse population to learn rigorous computing—but the question remains which implementation mechanisms will best achieve it. Let's evaluate how two existing avenues fare.

Mechanism 1: Stand-Alone Computing Courses

The most obvious solution seems to be: *add computing courses to every curriculum*. This runs into some natural roadblocks:

- Schools must find funding to pay for all those new computing teachers.
- Those teachers need to be qualified, or else rigor will suffer; in a terrific job market, they are extremely difficult to find. (In fact, some great teachers we know have left for industry. Paradoxically, the time when people pay most attention to a field may be the time it is most difficult find enough teachers for it.)

Three worthy goals.



- Finding qualified teachers can be even more difficult in poor and rural schools than in cities (as we are finding in practice).
- Schools must make time in the day and space in the building to teach another subject. What will they displace? The humanities? Art? Physics? Statistics?

Some places that are following this route are currently funded generously by the tech industry (usually in return for offering only their chosen curriculum). Since it is unlikely the funding will flow endlessly, what happens when budgets are cut or the largesse dries up? Odds are those courses will be the first to be cut in all but the wealthiest districts, and computing will suffer the same fate as music and art in the USA. Furthermore, because planning interdependent courses is hard, these courses will likely run in a vacuum, making it even simpler to cut them when it becomes convenient to do so.

One growing response is to *mandate* computing courses throughout some geographic region. This automatically achieves equity and scale. However, it comes with its own subtle problems. The problems of funding and qualified teachers do not go away; if anything, they are exacerbated because of the significantly greater demand imposed by a mandate. But there are also subtle problems: if a class is mandatory, there is a perverse incentive to lower the rigor of the course. After all, who wants to see a student held back or lose a scholarship simply because they struggle in their Python class?

Ironically enough, there is not even anything "disruptive" about this model! It more closely resembles an enterprise business deal or a top-down diktat than the kind of organic, bottom-up groundswell the fans of disruption preach. The funding model chosen by disruptive companies turns out to be

Calendar of Events

May 4–9
CHI '19: CHI Conference on Human Factors in Computing Systems, Glasgow, Scotland, UK, Sponsored: ACM/SIG, Contact: Geraldine Fitzpatrick, Email: geraldine.fitzpatrick@tuwien.ac.at

May 5–6
Expressive '19: Joint Symposium on Computational Aesthetics and Sketch Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering, Genoa, Italy, Sponsored: ACM/SIG, Contact: Joaquim Jorge, Email: joaquim.jorge@gmail.com

May 9–11
GLSVLSI '19: Great Lakes Symposium on VLSI 2019, Tysons Corner, VA, Sponsored: ACM/SIG, Contact: Baris Taskin, Email: taskin@coe.drexel.edu

May 13–15
HotOS '19: Workshop on Hot Topics in Operating Systems, Bertinoro, Italy, Sponsored: ACM/SIG, Contact: Mirco Marchetti, Email: mirco.marchetti@unimore.it

May 15–17
WiSec '19: 12th ACM Conference on Security and Privacy in Wireless and Mobile Networks, Miami, FL, Sponsored: ACM/SIG, Contact: A. Selcuk Uluagac, Email: suluagac@fiu.edu

May 17–19
CompEd: ACM 2019 Global Computing Education Conference, Chengdu, China, Sponsored: ACM/SIG, Contact: Ming Zhang, Email: mzhang_cs@pku.edu.cn

May 21–23
I3D '19: Symposium on Interactive 3D Graphics and Games, Montreal, QC, Canada, Sponsored: ACM/SIG, Contact: Sheldon Andrews, Email: sheldon.andrews@gmail.com

the very model they eschewed on their path to success.

Mechanism 2: Integrated Computing

Let's instead consider an alternative model of computing education. It recognizes computing is a new creative medium and vehicle for exploring myriad subjects, ranging from mathematics, biology, and physics to social studies. Why not, then, *integrate* computing into each of these subjects?

Presumably, most people do not believe all other disciplines are going to collapse and be replaced by computing; rather, computing will enrich and enhance those subjects. Therefore, those subjects should start modifying their presentation to show the impact computing will have. In social studies, for instance, there are already well-established means of asking and answering questions (surveys, ethnographic studies, literature reviews, and so forth). Computing does not displace these but rather supplements them, providing a new and rich way to pose questions: *a program is a way of posing a question of a dataset*. In turn, not every student is enamored of computing, either, and a generic introduction to computing is unlikely to sway them. In contrast, a contextual introduction in a subject that already interests them is far more likely to get them to see the value of computing.

Integrated materials can achieve all three of the goals we have described in this column. By embedding into already-required courses (such as math), they achieve the same diversity and scale as required computing courses do, without the same constraints. Rigor follows much more directly because of the existing rigor of subjects it embeds into: teachers in those subjects would not accept a curriculum that does not seem to make a meaningful contribution to how they teach their discipline. All this can be done at far lower cost, because it does not require entire new cadres of teachers to be hired or new classes to be added; the burden shifts to training the teachers already in the system or those entering it.

Curiously, integrated computing adheres far more closely to the model of disruption so beloved in our industry. It is lightweight: it does not require large outlays of time, space, and money. It has few dependencies, so it is easy to parallelize. It usually follows from bottom-up, grassroots interest. It is "sticky": it is unlikely to disappear when a generous donor's priorities change. And it lends itself to strong network effects in multiple ways: teachers within a discipline reinforce and improve the computing integration for their discipline, while teachers within a school support and reinforce student computing education for each other.

But integrated computing is imperative for another reason, too: computing should not fall victim to the same peril that befell mathematics.

This, of course, is the good news. The bad news is that integrating computing is far more difficult than delivering it as a stand-alone subject. Teachers in other disciplines need to be convinced that computing has anything to offer. Airy promises of the power of "computational thinking" are met with appropriate skepticism from teachers in other disciplines, because more than 100 years of quality education research shows the difficulties of achieving transfer across disciplines.[1] Validated research is much more compelling, and this takes time and effort. Also, teachers feel pressure to choose between doing more of their own discipline, or sacrificing some content they know and love to make room for computing. Thus, an injection of computing must be judicious, focusing on content that is *meaningful in the host discipline*; it must also "pay its own way," providing large value for small investments of time. Achieving all this is difficult.

Difficult, but not impossible. Programs like AgentSheets,[a] Project GUTS,[b] agent-based modeling,[2] and Bootstrap[c] are existence proofs that we are making substantial progress toward our stated goals. Thus, we believe integration is a strategy well worth pursuing, in parallel to stand-alone, required computing.

Mathematics: A Cautionary Tale

In short, integrated computing can achieve all three criteria we have described, which stand-alone approaches struggle to meet. But integrated computing is imperative for another reason, too: computing should not fall victim to the same peril that befell mathematics. While math dramatically impacts numerous disciplines, it is routinely siloed into stand-alone classes; as a result, the connections between math and other disciplines are often invisible to K–12 students. (In contrast, some institutions have tried to institute "writing across the curriculum," to help students improve their writing in a context meaningful to them.) Computing has a chance to avoid this fate, and the evidence so far is that we can succeed at integration. Moreover, stand-alone courses would be much richer if their intake consisted of students already versed in computing from other disciplines. Thus, with the right models of curricular design, integration strategies, and funding, we can achieve sustainable Equity, Rigor, and Scale. C

a See http://www.agentsheets.com/
b See http://www.projectguts.com/
c See https://www.bootstrapworld.org/

References

1. Bransford, J.D. and Schwartz, D. Rethinking transfer: A simple proposal with multiple implications. In *Review of Research in Education, 24*. American Educational Research Association, 1999, 61–100.
2. Wilensky, U. and Rand, W. *An Introduction to Agent-based Modeling: Modeling Natural, Engineered, and Social Complex Systems with NetLogo*. MIT Press, Cambridge, MA, 2015.
3. Wing, J. Computational thinking. *Commun. ACM 49*, 3 (Mar. 2006), 33–35.

Emmanuel Schanzer (schanzer@bootstrapworld.org) works at Brown University, Providence, RI, USA. He is a co-Director and founder of Bootstrap. A former public high school teacher in Boston, MA, USA, he designed Bootstrap:Algebra as a curriculum for his own students after being exposed to the Program by Design methodology in college.

Shriram Krishnamurthi (sk@cs.brown.edu) is a Professor of Computer Science at Brown University, Providence, RI, USA, and a co-Director of Bootstrap. He co-founded the Program by Design project, which inspired Bootstrap.

Kathi Fisler (kathi@bootstrapworld.org) is a Research Professor of Computer Science at Brown University, Providence, RI, USA, and a co-Director of Bootstrap.