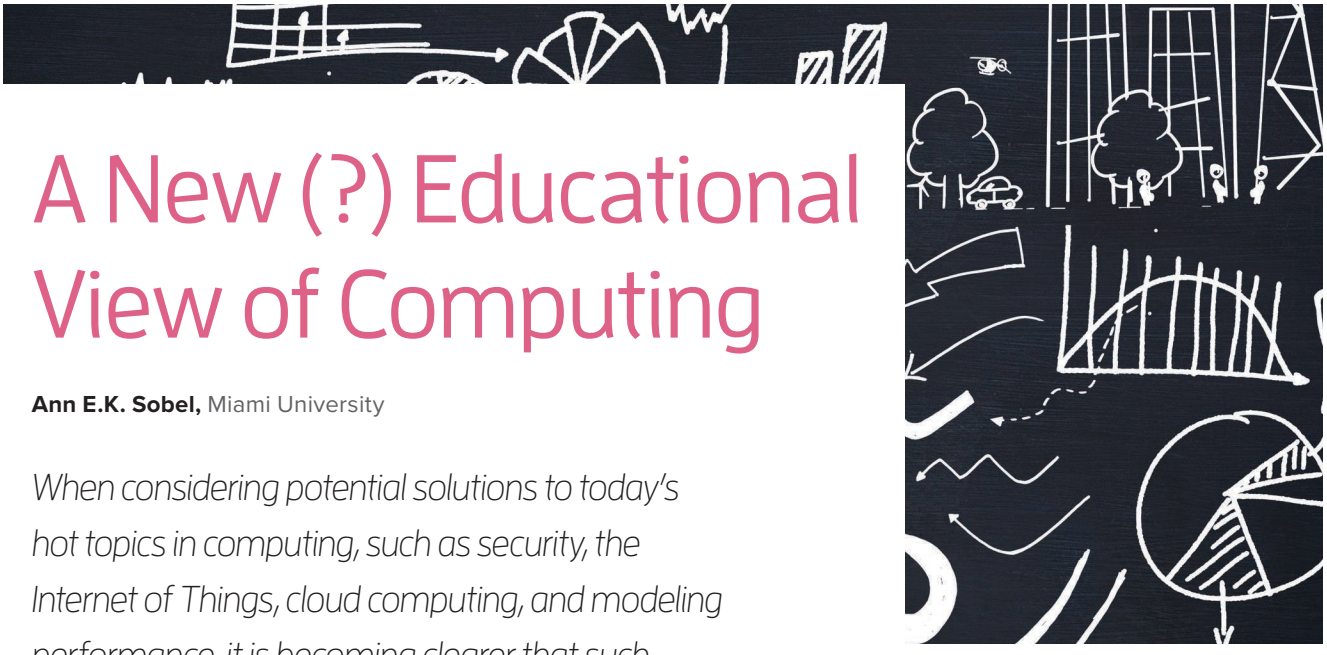
EDITOR ANN E.K. SOBEL
Miami University; sobelae@muohio.edu



A New (?) Educational View of Computing

Ann E.K. Sobel, Miami University

When considering potential solutions to today's hot topics in computing, such as security, the Internet of Things, cloud computing, and modeling performance, it is becoming clearer that such approaches need to incorporate hardware-oriented aspects, which is a step backward in our current educational view of computing.

As an educator who has played several roles in the construction of the IEEE Computer Society/Association for Computing Machinery Undergraduate Curriculum guidelines through the years, interactions with other computing education-oriented colleagues show a consensus that, over the decades, the contents of the Guidelines have moved away from computer hardware. Recommended programming languages have also changed in this way as time has passed, moving from languages like C, which blatantly exposes the machine, to Java, which hides the machine as much as possible. Today, problems, even those already tied to hardware, are growing both in number and significance, indicating that a hardware component in undergraduate computing education is necessary.

To illustrate this point, a partial list of such issues in the pursuit of sound, future-forward solutions is as follows:

- the incorporation of performance/computation model directives in the latest C++ standard library
- the identification/coordination/timing of the massively growing number of the Internet of Things applications
- the extensive use of cryptography, which relies on the construction and interpretation of codes used for encryption; this code has been and will increase in its hardware dependency, given quantum computing on the horizon
- the almost hidden compiler code optimizations by interactive development environments and how the structure of generated code influences the ability to compromise software (the insertion of malicious code); moreover, the structure and meaning of call stacks and their direct relationship to potential malicious behavior
- the pursuit of hardware-specific imaging rendering/analysis/enhancement toward total realism and accuracy
- the movement of generic operating system functions to hardware-customized ones to enhance features and performance
- the continued growth of microcode for embedded systems and robots; while it can be argued that

Digital Object Identifier 10.1109/MC.2019.2930098
Date of publication: 24 September 2019

hardware interaction in the context of embedded systems places this topic in a computer engineering discipline, it becomes less clear when the code communicates with other internal and external software systems.

WHAT SHOULD BE DONE?

I am aware of academic attempts to address some of these issues by offering degree programs, that is "computer science and computer engineering." But does such an approach really address the problem, when this solution just splices together two very different strategies, with each retaining its own views/methods/methodologies? These disciplines need to be integrated so that software construction incorporates the operating system/hardware on which the software will run by merging the best practices of each in a way that they can work in tandem.

As a reasonable start to addressing these problems, students should learn many of the concepts typically taught in a compiler design course, not to learn how to write compilers but to gain an understanding of the interplay between the code generated and the architecture. Focusing on which outside entities can interact with the code makes the programmer consider interactions that support security by design. Furthermore, a programmer needs to understand not only what an executable looks like but how, when, and by whom its sections are used. Again, focusing on such issues brings the architecture into the code generated that is needed by current computing solutions.

As educators, we are continually challenged to find and apply new strategies to solve today's computing problems. We must also make judgment calls as to which and when such approaches should be integrated into our degree programs. While the Undergraduate Curriculum Guidelines exist for the purpose of what the community believes should be taught, there is a transition phase where cutting-edge programs dare to test what may very well become the accepted educational standard in the future. We may just find that taking a step back to a more machine-oriented methodology is the right path forward.

Digital Object Identifier 10.1109/MC.2019.2937717

ANN E.K. SOBEL is an associate professor at Miami University, Ohio, and editor of the "Computing Education" column. Contact her at sobelae@miamioh.edu.