Chapter 12
Coding Across the Curriculum: How to Integrate Coding Into Content Areas

Janna Jackson Kellinger
University of Massachusetts, Boston, USA

ABSTRACT
This chapter explores why teacher educators should teach teachers how to integrate coding across content areas and how to do so by applying concepts of computational thinking such as using algorithms, flowcharts, and Boolean logic to all fields. Teaching teachers how to teach coding across the content areas offers opportunities to diversify people in a field where intimidation, discrimination, and lack of opportunities has effectively kept the field of programming largely white or Asian and male. In addition, as our lives become more and more infused with technology, Rushkoff warns that we either learn how to program or become programmed. This means that not everyone needs to become a computer programmer, but everyone needs to understand how programming computers works. In other words, coding across content areas would help prepare all students, not just those pursuing the field of computer science, for the 21st century.

INTRODUCTION
Up until recently with the advent of “Everyone Can Code” movements such as “An Hour of Code” and “No Fear Coding”, coding has been regarded as the realm of the geeks, relegated to “computer science” courses that imply that only those in the sciences can comprehend and produce these new languages. Because of this, most content area teachers have shied away from even considering coding as within their subject area domain. Indeed, most people have, unless they fit the white or Asian male stereotype of geeks reified by movies in the 80s like the Breakfast Club and Revenge of the Nerds. Because of this, the most promising field in the workforce is largely male and white/Asian and rampant sexism has kept it so (Myers, 2018). When those of us who do not fit this profile dare enter the field, not only do we face discrimination, but sometimes explicit threats like what happened in GamerGate where female
video game designers were harassed and even doxed (Dewey, 2014). This results in a large swath of the population being iced out of one of the most promising fields. According to the U.S. Bureau of Labor Statistics (2019), “Employment of software developers is projected to grow 21 percent from 2018 to 2028, much faster than the average for all occupations.” In addition, that field is missing out on potential talent at a time when new, innovative technological solutions are needed to solve the world’s problems, such as the climate crisis and election interference.

My own history is laced with the sexism emanating from this field. In my very first summer job, despite being in Advanced Placement (AP) Computer Science, I was assigned to do data entry while the male students from my high school were assigned to the computer room. When my sister, who is now an engineer, went to her high school guidance counselor to find out what elective courses were available, he told her about the gourmet cooking class but not the engineering one and then later discouraged her from applying to engineering school. When I was the only female in a computer science class in college, the professor constantly pointed it out by saying things like, “Gentlemen (pause) and lady” and exaggerating “Miss” when saying my name. By now, we would hope this atmosphere would have changed, but statistics about the computer science field speak to the long-term effects of discrimination (Myers, 2018). Because of intimidation, discrimination, and lack of opportunities, the field of programming has effectively stayed white/Asian and male.

One way to open up the field to a wider audience is to introduce programming across the curriculum so that content areas traditionally associated with other demographic groups can introduce students to coding. Since coding is essentially writing a set of decision-making directions, recipes in gourmet cooking are coding; choreography is coding; telling someone how to get from point A to point B is coding; the list can go on and on. Viewing coding this way allows us to see how coding cuts across all content areas. The possibilities are endless. In fact, when I was taking that AP computer science class in high school, the teacher was also the Physical Education teacher. When viewing coding as writing a set of decision-making directions, you can see how this is a perfect fit. After all, playing sports involves making a series of decisions within the confines of the rule set defined for that sport and the referees are the compilers (programs that translate computer code to basic machine language and send out notifications when code does not conform to the conventions of the programming language), who call someone out when they do not conform to the grammar of the sport.

**TEACHERS AS CODERS**

By expanding coding to a set of decision-making directions, we can also see how teachers are natural coders. In fact, they write code every day in their lesson plans. They then are the computers that implement this code, sometimes even changing it up on the spot. Because they are coding humans, or rather creating conditions for humans to learn optimally, you could argue that their coding is even more complicated than coding a computer. Larry Cuban (2001) has lamented about teachers being reluctant to dip a toe into digital technologies. However, introducing coding to teachers in this way and then demonstrating options for ways in which they can integrate coding into their content areas uses Lee’s (2001) cultural modeling to allow teachers to see the connections between coding, teaching, and their subject area.