Sensing Computers: The Impact of Increased Abstraction on the Programmer's Experience

Micaela Simeone, Class of 2022

As computer programming becomes increasingly characterized by abstraction and automation, a rift grows between the nature of programming in decades past and the nature of programming today. Whereas, years ago, the discipline necessitated a detailed understanding of a computer's internal architecture, a programmer's interaction with the machine today is mediated by multiple, often obscured layers of abstraction. Consequently, the experience of programming and even the definition of programming itself have shifted. However, historical texts suggest that the ideals of problem-solving, analytical thinking, and creativity that shape the discipline have consistently characterized how programming is taught and how programmers see themselves.

For my project, I traced these ideals alongside technological developments in computer science before considering how analyzing the experiences of programmers across the years can reveal important insights into the changing relationship between programmer and computer. Ultimately, centering the programmer's experience in a conversation about computer science's evolution can prompt us to critically reevaluate the status of the programmer at a time when computers themselves are absorbing the work of programming. My research culminated in a 27-page paper that, essentially, examines the changing experience of the programmer using historical and industry texts as well as archival materials exploring the history of computing at one institution—Bowdoin College. My analysis also focuses on insights from interviews with Bowdoin Computer Science faculty and students, as well as with two senior programmers. In short, my paper highlights the implications of developments in programming for the relationship between programmer and computer, ideals and realities that shape the discipline, and the importance of emphasizing the programmer's experience.

I separated my research into three phases: texts, archives, and interviews. I began by reading historical works on the history of computing spanning the nineteenth century to the present day as well as textbooks and memoirs by programming pioneers and experts. Next, I conducted archival research at the Bowdoin College Dept. of Special Collections & Archives to acquire information about computing at Bowdoin since the 1960s. In my third phase, I designed and carried out interviews with Bowdoin Computer Science faculty and students to gather opinions about changes in computer science and the programmer's experience.

In my paper, using findings from my historical readings, I emphasize how, since its origins, programming has been framed as a field granting limitless possibilities. Next, gleaning from readings, archives, and interviews, I also examine how the computer is described as the programmer's tool, show how an increasing amount of programming work is being offloaded to the computer, and consider how this shapes the possibilities that are open to programmers. I explore how programmers, educators, and others describe programming, showing how programming is ultimately framed as a mode of thinking that requires creative problem-solving skills. I then begin to consider how the ideals of programming have played out over time, focusing on how technical evolutions have altered the programmer's ability to recognize, trace, and solve errors and fulfill their role as problem-solvers; I also use readings and interviews to question how technical changes have impacted what it feels like to program. The final section of my paper considers how my analysis fits into the reality of computing as an ever-evolving discipline, and I conclude by framing my analysis in the context of AI and automation.

As computers absorb an increased amount of programming tasks, programmers will need to carefully delineate the importance of their role. The extent to which they remain "in the loop" will depend in part on creating new definitions for the title of "programmer" and on redefining the role of programmers and their relationship to computers.

Faculty Mentor: Professor Crystal Hall

Funded by the Surdna Foundation Undergraduate Research Fellowship