Personal, Relevant Background and Future Goals Statement

Andrew Head. PhD Student, UC Berkeley

Code is awful. Code is wonderful. It is an impediment to all progress. It is an enabler of amazing feats of automation. Code keeps us up at night when we grapple to find that one line that causes everything to crash. It keeps us up at night as we furiously type out the systems of our dreams. Code is fantastic. And abysmal. And great.

It's certainly a dichotomy—code enables exceptional technical and creative power, but it's also among the most cognitively demanding things a person can make. When I discovered MATLAB at age 19, code ended my days as an academic nomad. It led me to computer engineering and an intense preoccupation with how to build systems. Now, six years after working with glorious, horrible code, I am certain: to write terrific code, people must be inspired and helped to methodically approach their struggles with code. My role in academia and engineering is to help people more efficiently write great code.

For most of the 90 million estimated programmers in the U.S., code is a part of the job. It is a skill learned just well enough to do work. Programs are pieced together as bricolage for professionals and non-professionals alike. As programmers increasingly turn to online code examples to light the path to quick development, I want to help programmers develop long-term proficiency in their code. I want to **understand and improve programmers' workflows for learning from and reusing code examples**.

My interests and competency in this research began with my first undergraduate research in human-computer interaction—a project to build and evaluate a system for end-user programmers. I contributed to e-Chimera, a visual programming environment that equipped social scientists and psychologists with tools to rapidly develop mobile experiments. As an undergraduate doing research for the first time, my work was in design and implementation. I proposed and implemented features for a multi-level graphical editor that allowed a user to specify a mobile experiment at multiple resolutions simultaneously.

The system was never published, and the usability evaluations I helped conduct have not yet been shared. However, e-Chimera set two inspirations I uphold today: First, the right representations of code make programming and code modification effortless. Second, today's technology allows interactions to be embedded anywhere for in-the-wild use.

During my transition to graduate school, I sharpened my skills as a researcher in human-computer interaction. Collaborating with Jingtao Wang, I served as the primary researcher on the ToneWars project. I conducted interviews, iteratively developed a serious game with a pedagogical focus, conducted usability tests, and presented the work at Intelligent Tutoring Systems 2014 [1]. When I arrived at the Berkeley Institute of Design, I collaborated with Valkyrie Savage on Lamello. We researched a new method of designing passive acoustics-based tangible input devices that could be 3D printed. My contributions to the algorithm and evaluation of the system were included in a paper that turned out to be my first publication at CHI [3], a premier conference in HCI.

In January, I began to pursue my own research focused on a broad goal of helping people efficiently write great code. Advised by Björn Hartmann and Marti Hearst, our group had a powerful synergy. Björn is a leading researcher in development practices for programmers and makers, and Marti is an expert in information seeking interfaces. Our work evolved into the Tutorons project. It explored how the browser can be instrumented to detect and explain the unexplained code that appears in web tutorials.

The project developed rapidly. For three languages frequently embedded and unexplained in web tutorials—regular expressions, CSS selectors, and the wget Unix command—I built "Tutorons". Each Tutoron was a routine on a web server that automatically detected the code from fragmented snippets embedded in HTML, and generated explanations of found code. The explanations they built were rich, including English prose descriptions and usage examples. Around these artifacts, I formulated design guidelines for automatic explanations of code on the web. An in-lab study revealed that the explanations reduced the number of external documents programmers accessed for a set of programming tasks. Quantitative evaluation showed promise for automatic detection of code in web tutorials. Tutorons aimed to reduce the cost of understanding and modifying code that programmers found on the web.

I presented Tutorons at VL/HCC 2015, a prime conference for end-user programming research [2]. The work was well-received, gaining a nomination for Best Paper. But studying the background for the work revealed gaps in our understanding of how programmers make use of online examples. Existing work offers evidence of programmer difficulties reusing code. However, there is no comprehensive catalog, or figures on the frequency or severity, of problems programmers encounter when reusing online code.

This missing work will be the focus of my PhD study. It will document a group of programmers often left behind by current research: those engaging in open-ended projects for the first time without a systematic approach to code foraging and reuse. This group includes upper division CS students, and software engineers that have freshly graduated. Achieving this understanding has the potential to both **improve STEM education** and to **increase and improve public engagement with science and technology** for these programmers.

My research aims to **help people write good code efficiently**. Beyond the lab, my goals broaden. When I came to Berkeley, I found a rich network of peers to support the ideation and execution of superb design and research. Within this community, I strive to enable others to thrive within a culture of design and research.

One channel through which I foster design is teaching. Last summer, I taught as the lead teaching assistant for Berkeley's upper division user interface design course. For many students, this course will be the first (and perhaps only) time they learn and practice need finding, problem definition, interface programming for state of the art hardware, and presentation skills. Throughout the summer, I and the other instructors worked hard to develop new project, lab, and review materials. It paid off with substantial student engagement and growth. At review sessions, we achieved historically high attendance, with around 50% attendance 2/3 through the term. I saw one group I regularly critiqued rapidly and effectively iterate to develop a smartwatch-based haggling assistant app that won first prize at the end-of-semester poster fair from our invited judges. After review sessions on presentation design, we saw students incorporate wordplay and skits into their lightning talks.

Teaching this course fed back into my desire to inspire efficient, clean coding practices among a group who did not seem to have much experience in it. Students frequently heard me (perhaps to their chagrin) directing them to the Android Developer Guide as an investment to develop a strong mental models of the APIs they used. I gained first-hand knowledge of the difficulties programmers have when trying to learn new libraries informally for challenging, open-ended programming assignments. And while many of them spent dozens of hours getting to know new tools and debugging the behavior of unfamiliar APIs, proficiency with this set of tools enabled them to combine these components into phenomenal apps. My experience instructing for this course affirmed that effectively learning from usage examples can be exceedingly hard, but mastery of this material enables creative feats.

The community I identify with most closely is my research group, within which I take efforts foster an environment of design and research. I regularly volunteer at the CITRIS Invention Lab, a student hackerspace, to keep the space open for students developing new interactive devices. I coordinate the Berkeley Institute of Design weekly seminar with Cesar Torres. We invite scholars to speak about their work, offer them a tour of the HCI facilities, and, most importantly, arrange meetings between them and students and faculty with mutual interests to promote the exchange of academic ideas. I organized our discipline's "Visit Day" outing, where five prospective PhD students joined our department on a hike with lightning talks interspersed when we took breaks to view the San Francisco Bay. I also organized a reception to welcome the UIST program committee last Spring, where visiting experts in HCI mingled with Berkeley students and faculty amidst dozens of posters and a handful of demos. Through this active effort within the Berkeley Institute of Design, I aim to enable lively, brilliant academic discourse that I've come to expect from my peers.

This involvement within my academic family sets the backdrop against which I do my current research. Specifically, my work aims to encourage and enable programmers to better learn from, design with and develop quality code using online examples. There is no better place than academia to study this question. Only within academic research is learning, cognition, and problem-solving sufficiently and unanimously valued that I can gain support to understand non-professional programmers and the interfaces that will support them. I see the NSF fellowship as assistive to this work: It would ensure that this research would remain supported through my intended graduation in 2019, regardless of my funding source.

By the conclusion of my PhD, I hope to have contributed to the study of how programmers can better use rotten, marvelous code two ways: First, I hope to contribute to theory that explains programmer challenges in learning from, designing with, and writing code using examples. Second, I hope to build and open source concrete tools that enable new research and contribute value to a larger programmer community. With a professorship or industry research position as the ultimate next step, I hope that this research and its artifacts will inform, enable and inspire the work for both my career and others'.

[1] A. Head, Y. Xu, and J. Wang. "ToneWars: Connecting Language Learners and Native Speakers through Collaborative Mobile Games". English. In: *Intelligent Tutoring Systems*. 2014.

[2] A. Head et al. "Tutorons: Generating Context-Relevant, On-Demand Explanations and Demonstrations of Online Code". In: *VL/HCC '15*.

[3] V. Savage et al. "Lamello: Passive Acoustic Sensing for Tangible Input Components". In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 2015.