Statement on Research Philosophy

Hammad Ahmad

To me, research embodies two interrelated goals: expanding our understanding of the world, and training the next generation of scientists. Throughout my graduate career, I have been working on balancing the two goals with an emphasis on interdisciplinary collaboration and student mentorship.

Research Interests

Broadly, my research focuses on software engineering with an emphasis on human factors and pedagogy. In particular, I am interested in (a) understanding the cognition behind Computer Science and software engineering tasks, (b) providing tool support for developers in novel domains, and (c) verifying correctness in complex, high-assurance, safety-critical systems.

Cognition for Computer Science. I believe that investigating cognition for programmers can have potential to inform pedagogy and guide tool development and retraining. This belief is backed by prior work by others demonstrating that cognitive interventions (e.g., a technical reading training course) can improve programming outcomes in introductory computing. As such, I have devoted a significant amount of research effort investigating student cognition for Computer Science tasks and its implications on training and education.

An indicative example of my work here involves the use of medical technologies like functional magnetic resonance imaging (fMRI) and transcranial magnetic stimulation (TMS) to investigate the relationship between brain activity and comprehension tasks. Researchers have previously investigated pedagogical interventions informed by correlations between brain activity for spatial reasoning (e.g., mentally manipulating 3D objects) and that for computer science tasks. I believe that confirming a *causal* link between the cognitive processes at the brain level can help educators tailor interventions to better favor student outcomes. We conducted an IRB-approved human study using TMS to disrupt activity in brain regions of interest and subsequently measured participant performance for the comprehension tasks. Our results have challenged the research community's understanding of cognition for programming by showing no direct causal relationship – where researchers would have expected one to exist – between implicated brain regions and programming, opening up new venues for exploration of TMS for programming. I also had the opportunity to mentor two women students for this project, including training my mentees for safe administration of TMS, data collection and statistical analysis, and advising writing up a paper manuscript for a peer-reviewed conference. The paper was recently accepted to appear at ICSE [2], a top-tier software engineering conference.

Tool Support for Developers. I believe that providing tool support for developers (both professionals and students) can be instrumental in reducing maintenance and training costs, particularly in novel domains (e.g., hardware designs, hybrid and embedded systems). While there exist a plethora of instructional support and automated debugging tools for software programs (e.g., those written using C++ or Python), I believe tool support for hardware designs – that otherwise remains limited – warrants attention from the research community.

On this front, I have developed the first-of-its-kind automated program repair (APR) tool for hardware designs, and investigated its use as a debugging aid for student designers. Our tool, CirFix, uses novel insights about the hardware domain to bridge the gap between hardware designs and software-based program repair to repurpose established APR techniques for hardware. We found that not only is CirFix effective at automatically repairing defects in hardware designs, but also the tool's fault localization approach (i.e., an algorithm to automatically implicate potential sources of a bug) may be useful as a debugging assistant in a pedagogical context. I had the opportunity to mentor a first-generation, pre-doctoral student, Priscila Santiesteban, for this project. Priscila was responsible for conducting a subset of the experiments evaluating

CirFix, and took charge of recruiting participants for the human study and analyzing collected data. Notably, she was less familiar with the hardware design process, and I was able to make this research accessible to her by compartmentalizing different stages of the project (cf. "tasks" in agile software development) and having her initially take charge of components she felt more comfortable with (e.g., performing evaluation experiments where CirFix can be treated as a "black box", conducting the human study, performing statistical analyses, etc.). By addressing the novel components in stages, she was able to master the required material and push the project across the finish line. Priscila was also involved in the paper writing process, and subsequently used this work to pass her Ph.D. preliminary exam at the University of Michigan. Results from this work are published in ASPLOS [3] and Transactions on Software Engineering [6], two top-tier peer-reviewed venues, and have garnered significant follow-on interest – both from academia and industry.

High-Assurance Software Engineering. I believe that testing for bugs in software can be insufficient, and that elevating the status of a critical system (e.g., autonomous cars) from "tested" to "correct" is becoming increasingly necessary. In this area, I have produced new formalisms that allow for time-critical safety properties for software to be expressed and reasoned about, and worked on reducing the amount of manual human effort needed to prove correctness in complex distributed systems.

As an example, I have worked on combining two widely-used logics – signal temporal logic (STL; used for monitoring purposes) and dynamic logic (d$\mathcal{L}$; used for verification purposes) – to produce signal temporal dynamic logic (STd$\mathcal{L}$), a dynamic logic that reasons about a subset of STL specifications over executions of cyber-physical systems. d$\mathcal{L}$ and its variants have been used to verify correctness in safety-critical such systems, but the logics remain limited in their expressiveness. In particular, the logics are unable to reason about intermediate states of a system within specified time intervals, focusing instead on properties like "at all times" and "there exists a time such that". Our formalism, STd$\mathcal{L}$, significantly augments the expressiveness of d$\mathcal{L}$ by allowing reasoning about temporal properties in given time intervals (e.g., "for all executions of a program in a self-driving car, if the car spots an obstacle in its path, the program applies the brakes *within 0.5 to 1.5 seconds*"). This work was published in HSCC [4], and has resulted in industrial interest for autonomous vehicles.

Mentorship

As a graduate student, I have had the pleasure of mentoring a total of four undergraduate and pre-doctoral students (including two women and one returning adult students), with an emphasis on students less represented in Computer Science. Three of my mentees are authors on peer-reviewed publications at prestigious venues [2, 5, 6].

I put a particular focus on seeking opportunities to mentor students from non-traditional backgrounds. For instance, one of my mentees, Zachary Karas, was a returning adult student with a non-computing undergraduate degree. I had the opportunity to mentor Zach on a project investigating student cognition for formalism comprehension using eye-tracking, and to guide him through the process of applying to graduate schools. I made this research experience accessible to Zach by structuring the project such that he was able to use his Psychology background (e.g., with eye-tracking and cognition) to address initial imposter syndrome concerns, following which we had frequent (2–3 times weekly) meetings to allow him to get more comfortable with Computer Science formalisms. I led the first few eye-tracking sessions with Zach as an observer, following which I was able to hand off experimenter duties to Zach, who reported feeling comfortable taking over after observing first. The project resulted in a peer-reviewed publication at the peer-reviewed conference ICSE [5], and I was able to guide Zach to prepare and give his first conference talk. Zach is now a Computer Science Ph.D. student at Vanderbilt University.

I make an active effort to scope a research project to best fit a mentee's interests. This is best exemplified in a recent project undertaken by one of my mentees, Joshua Velten. From the day I met him, Joshua was particularly interested in software development, but was excited to try out research to see if graduate school was the best fit for him. While none of my projects at the time involved developing a new software application, I was able to work with Joshua to come up with three candidate projects, of which two were curated by me and the final was pitched by him. We ended up working together to combine one of my research directions with Joshua's idea, devising a project that let him experience both developing a debugging application and planning a human study to investigate the extent to which students find it useful. Joshua ultimately chose to pursue a career in industry after deliberating graduate school, and I could not be more pleased that he was able to make an informed decision: to me, a major goal of mentoring is identifying a student's interests and presenting the student with guidance and opportunities to pursue those interests, and I was able to do just that with Joshua.

As part of a long term research collaboration with Dr. Stephanie Forrest at Arizona State University, I have also greatly enjoyed working with non-traditional students in an auxiliary advising capacity. For instance, I have worked with a student at ASU who used a wheelchair, and as such, had to face additional hurdles working in-person from a lab. The student also suffered from severe anxiety associated with scientific writing, and was looking for a mentor on this front. I was able to work with him by planning remote meetings with flexibility in mind: in addition to regular remote interactions, I offered and suggested the option to hold impromptu meetings, since the student's mobility challenges made it difficult to interact with other lab mates for quick questions. I often met with the student between my lectures and scheduled office hours for the semester, and was able to offer guidance on best practices for scientific writing and communicating research results. We were able to work together on the student's anxiety by establishing a judgement-free zone where we put words on paper to overcome the "tyranny of the blank page", following which we iteratively refined the text together. This mentorship was a success, and the student was able to significantly contribute to the text writing. The resulting manuscript is published in the peer-reviewed conference PPSN [1].

Similarly, I have worked with a returning adult graduate student at ASU with several years of industry experience working with hardware designs, but less comfort with human studies. I have had the pleasure of sharing my research expertise with the student by offering advice on human study design practices that admit eye-tracking (which aligns with her current research interests), including providing feedback on stimuli creation, choice of hardware equipment, and IRB applications. Conversely, she was able to offer feedback on my research on automated repair for hardware designs, including sharing best industrial practices to be incorporated into the CirFix workflow. I see mentorship as a two-way street, where both mentor and mentee can learn from each other. While this remains true for mentoring traditional undergraduate students (e.g., students often provide a novel perspective to an existing research idea), this mutual benefit is doubly apparent for returning adult students, who have had their shares of life experiences and expertise that can help shape research.

Future Directions

My research on cognition and tool support for developers has suggested promising pedagogical implications, and I hope to investigate these through controlled intervention studies. For instance, my eye-tracking study involving formalisms has brought to light a gap between student knowledge and instructor course objectives, as well as a difference in problem-solving strategies employed by higher-achieving students. I hope to investigate the effects of suggested instructor interventions on student course outcomes across institutions (e.g., collaborating with professors at the University of Michigan). Additionally, while medical technologies like fMRI may not be readily available in research settings at every institution, my expertise with interdisciplinary human studies allow me to use cheaper measures (e.g., eye-tracking) to investigate cognition, and its implications on training, associated with a variety of comprehension tasks.

I acknowledge that conducting human studies can seem like a daunting task for new researchers with less preparation (e.g., getting IRB approval, addressing recruitment challenges, etc.), and to provide a wider array of research opportunities for students, I hope to further develop tool support for hardware designs. For instance, while CirFix laid the groundwork for automated program repair for hardware, many underlying techniques of the tool were a "proof of concept" and left room for improvement that an undergraduate researcher can work on (e.g., improving the standalone fault localization component). Similarly, for students more interested in mathematical and formal methods, I hope to extend my work on verifying safety-critical systems (e.g., by ramping up from student-familiar concepts like first-order logic to d$\mathcal{L}$ and beyond) and implement the reasoning rules into an existing automated theorem prover.

In summary, my previous research and advising experience has prepared me well to devise different types of research projects approachable by undergraduate students, and I am excited by the prospect of training the next generation of scientists at a new institution.

References

[1] H. Ahmad, P. Cashin, S. Forrest, and W. Weimer. Digging into semantics: Where do search-based software repair methods search? In *International Conference on Parallel Problem Solving from Nature*, pages 3–18. Springer, 2022.

[2] H. Ahmad, M. Endres, K. Newman, P. Santiesteban, E. Shedden, and W. Weimer. Causal relationships and programming outcomes: A transcranial magnetic stimulation experiment. In *International Conference on Software Engineering (ICSE), to appear.* IEEE, 2024.

[3] H. Ahmad, Y. Huang, and W. Weimer. Cirfix: automatically repairing defects in hardware design code. In *International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 990–1003, 2022.

[4] H. Ahmad and J.-B. Jeannin. A program logic to verify signal temporal logic specifications of hybrid systems. In *International Conference on Hybrid Systems: Computation and Control*, pages 1–11, 2021.

[5] H. Ahmad, Z. Karas, K. Diaz, A. Kamil, J.-B. Jeannin, and W. Weimer. How do we read formal claims? eye-tracking and the cognition of proofs about algorithms. In *International Conference on Software Engineering (ICSE)*, pages 208–220. IEEE, 2023.

[6] P. Santiesteban, Y. Huang, W. Weimer, and H. Ahmad. Cirfix: Automated hardware repair and its real-world applications. *IEEE Transactions on Software Engineering*, 2023.

Note: A full list of my publications is available in my CV.