Research Statement:

Developing software systems that work as intended has been my passion for over a decade now. In my bachelor's as well as my master's studies, I learned the fundamentals and theories of software engineering and Information Technology and had a chance to apply what I learned in the industry. This gave me a unique opportunity to receive valuable feedback on what are the real challenges for software engineers when it comes to practicing what they theoretically learned. As a software engineer with 14 years of experience in the software industry, I took different roles from software developer to software designer, architect, and team leader, which provided me with the possibility of working with so many talented developers who despite their expertise, from time to time, implemented code that either did not work or did not comply with the desired specification designed by software designers or architects. This contributed to my passion for finding ways to help programmers in developing correct and trustworthy software and thus, motivated me to start my Ph.D. with a focus on automated software engineering. My main goal was to dive deep and find the root cause of the incorrect code implementation by programmers and find a way to help them address this problem.

During my Ph.D., I explored different aspects of automated software engineering including program analysis, program comprehension, repository mining, API recommendation, formal methods, and more importantly, program synthesis. Throughout this process, I also learned and practiced how to become an independent researcher while communicating with other researchers in the field and utilizing available resources to me. I conducted research in three different, yet related, areas. In the first area, I focused on exploring the quality attribute traceability in legacy software, including (but not limited to) software certification and architectural tactic detection (ISSREW'2020). In the second area, I worked on pinpointing and resolving various challenges programmers deal with when trying to correctly implement architectural tactics -as one of the most important design decisions for preserving quality attributes of the code (e.g., security, availability, and performance). To that end, I developed an approach that can inter-procedurally analyze a given program, identify incorrect tactic implementation in the code, and provide API recommendations for fixing the incorrect implementation (published in ICSA'21 and ICPC'21). Next, I took further steps to go above and beyond just providing API recommendations and to fully synthesize the code for programmers and automatically add the code to the given code base. This is an inter-procedural program synthesis approach and can be applied to many different use cases beyond only architectural tactic implementation tasks and can tremendously help programmers, especially novices, to develop error-free code. This approach was recognized as the first-place award winner of the research competition at the International Conference on Automated Software Engineering (ASE'21).
Putting all these works together, I learned how to support programmers by automatically capturing the specification of the to-be-implemented code, generating the desired code, and adding them to the under-development program. In addition to my Ph.D. research, I have had a chance to attend two research internships at Google and PARC, both on projects related to software synthesis, with a focus on Machine Learning (ML)-based (e.g., Large Language Model - LLM) generative models and evolutionary search-based synthesis, respectively. So far, the outcome of these internships is two inventions (on their way to becoming patents) as well as two

papers (to be submitted). Moreover, being a Graduate Research Assistant, I was involved in writing grant proposals for different agencies, e.g., NSF, DARPA, and DoE, and mentoring under-grad, master's, and Ph.D. students.

Based on the lines of work I pursued, the knowledge and experience I gained, and the in-depth research I conducted so far, I will be following three main lines of research for my 5 years plan.

1. ***ML-supported Inter-procedural program synthesis:*** Although the work in the field of automated programming and program synthesis is very promising and all the branches of this field are hot-topic, I found an important gap when it comes to supporting programmers in real-life programming scenarios. Almost all the approaches introduced by researchers so far focus on intra-procedural code synthesis, meaning that based on a given specification, the synthesizer generates a block of code. However, in real-life use cases, programmers need inter-procedural code synthesis that can construct interrelated blocks of code that can be added to different parts of the program to implement a use-case or a scenario. As part of my Ph.D. research, I have developed an approach that can inter-procedurally implement tactic code snippets and add them to the program. This is a semi-formal approach; in the future, I intend to leverage the power of language models for performance improvement. To this end, I will use the knowledge and experience I obtained during my research internship at Google (my research on LLM-based synthesizers) to make it fully functional in the next 5 years.
2. ***Large-scale code repair:*** The current state-of-the-art in code repair is either too formal, requiring extremely explicit specification as input (and therefore, not robust enough toward variations in the given code base), or is ML-based approaches with no guarantee on the correctness of the repair outcome. As a continuation of my Ph.D. research, I intend to work on a hybrid solution, combining formal and ML-based approaches, to identify to-be-repaired parts, automatically generate the required patches, and replace the buggy code with the generated patches. This will also be an inter-procedural approach that can proceed beyond bug identification and repair at only the method level.
3. ***Self-reconfigurable Software:*** Another future direction of my work for the next 5-10 years will be developing a framework/approach that enables software systems to automatically update themselves based on their environment. There are some separate running efforts in API migration, code repair, and even ML-based code translation to update legacy systems based on a given specification (e.g., a list of the API that should be replaced, a template of the correct version of the code, etc.). However, I believe all these efforts could be either placed under an umbrella of a framework or supported by the development of a single general approach that learns how to evolve software from some software evolution examples. I am more interested in the latter and I believe that it could be the generalized version of the first and second items of my future research direction, outlined previously.

Based on my contribution to writing grant proposals for different funding institutions (e.g., NSF, DoE, DoD, DARPA, etc.), I believe that the above-mentioned projects are all in the interest of a variety of funding institutions.

In the end, I would like to thank you for your consideration and for reviewing my application.

Sincerely,
Ali Shokri

Ph.D. Candidate and Graduate Research Assistant
Dept. of Software Engineering and Global Cybersecurity Institute
Rochester Institute of Technology, NY, US
http://a-shokri.github.io