John Harrington

Spring, 2023

MS in CS Degree - Portfolio Option - Reflective Essay

When I began my graduate studies in Computer Science, I did not have a full appreciation for how broad of a discipline it is. This made it difficult to determine a specific direction or specialization most suited to my talents and interests. Immediately upon beginning the foundational courses, I realized that I had a strong interest in writing programs and learning new programming languages. After initially learning Python and writing some basic programs to interact with users and perform logic, I could better appreciate the object oriented capabilities that Java offered. Learning C++ shortly thereafter, I gained further appreciation for the complexities involved in manually managing memory, and that I had been taking for granted the garbage collector in Java and Python's object model. As my graduate studies continued, I would learn and explore many additional languages and frameworks as well as important concepts in Computer Science.

Programming languages served as a concrete means to implement the concepts taught in coursework, and frameworks provided an abstraction for completing meaningful work using these languages. However, I believe that the real advantage of this degree program compared to a coding boot camp is the level of detail and completeness with which the core topics in Computer Science were taught. Early on, I took a course in Computer Architecture which provided a low-level explanation for how computing systems work from the fundamental logic gates used within processors, to processor design and considerations such as instruction size, and culminated in learning how a basic compiler can be implemented. At the beginning of my degree, I could not fully appreciate this course, but after surveying a broader field of topics in Computer Science, I am grateful to have received such excellent instruction in these foundational topics.

The most difficult course completed during my degree program was Data Structures and Algorithms. The very same components of this course that were so difficult were also the most valuable: learning how to use common data structures and algorithms to solve complex problems such as dynamic connectivity, efficient searching in the context of an auto-complete search field, and computing lexical similarity on a large dictionary of English words. This course was excellent preparation for learning to program in C, a language which does not provide many data structures as part of its standard library.

The most useful course completed during my degree program was a course in Cloud Computing. This course allowed me to gain hands-on experience completing weekly assignments that required learning new frameworks and cloud based services offered by Amazon Web Services, Microsoft Azure, and Google Cloud Platform. These assignments were so valuable because they required writing code in languages already learned and well understood, but in the context of modern frameworks such as Terraform and Docker Compose with the purpose of provisioning cloud resources, containerizing small applications, and deploying these applications to the cloud where they could be immediately tested and improved.

A course in Software Reliability and Testing also provided excellent hands-on experience. In this course, I learned about edge and corner cases, how to anticipate them, and how to effectively and gracefully handle them. This class was taught by a faculty with industry experience, and so provided a lot of practical means to improve the quality of my code and provided expectations as to what working as a software engineer after graduation would be like. In this course, I wrote functional tests, unit tests, mastered dependency injection to increase code cohesion and decrease coupling, and worked through several code reviews.

As I matured in this degree program, I realized I had a strong preference to design and build application logic. Exploring mobile application development provided a way to learn and explore relevant topics such as design patterns, project architecture, and the differences among common platforms such as Android, iOS, and cross-platform tools such as Flutter. As my mobile applications became more complex, it became necessary to implement back-end services running in the cloud to effectively provide full functionality to these applications.

One such project involved using a Raspberry Pi to control an air quality sensor, take measurements, and upload these measurements to cloud storage. I also developed an iOS application that would visualize these measurements as a time series and display a user a push notification as soon as an air quality measurement exceeded World Health Organization guidelines. In order to implement this last part, I had to create a cloud hosted web-server that would register iOS devices that had installed my app, and would receive measurements from the Raspberry Pi and forward a request onto Apple's own back-end Apple Push Notification Services (APNs). Doing so, I learned how much I enjoyed the logic involved in creating complex back-end services.

In my final semester, I have received several job offers, the most interesting involving energy generation and distribution systems written in C and the development of a high performance real time operating system used on modern supercomputers. As such, my final courses in the degree program: Operating Systems and Embedded Systems, are extremely relevant. In my final semester, I am excited to learn more about building and controlling prototypical embedded systems as well as exploring the Linux kernel and writing device drivers, using system calls, understanding interprocess communication, and CPU scheduling algorithms.