Kelly Deng
CSE 488
Fall 2017 – Final Report

This semester, I interned as a part of the engineering team at a company that offers bare metal cloud services. This internship is actually a continuation from the summer, so while I will be mainly focusing on my accomplishments during this semester, I will also be talking about the things I learned and experienced throughout the entire internship as a whole.

Through this internship, I was able to gain exposure to languages – I'm using this term very loosely here – and tools I probably wouldn't have learned on my own (mainly because they weren't as high on my priority list as, say, C or Python would be). I was able to grasp basic AngularJS and Ruby on Rails by coding for their portal (frontend portion of their website) and API (backend portion), respectively. I was even able to learn how to test my code using Cucumber, a tool I have never even heard of before. I still remember when I first encountered it, I was quite surprised that the tests seemed to have been written in plain English! I eventually realized that the lines of "plain English" were defined behind the scenes with actual code, but nonetheless, the point of it is to be able to write tests that are easy to understand at first glance.

I was also able to find out how they were able to keep their code so uniform and organized. Every time a commit is pushed, a series of checks are run to make sure the code is up to par with the standards set. The one I was most impressed with was a linter and static code analyzer called Rubocop that is specific to Ruby. It fails the build even if there are offenses such as too many embedded *if* statements. It is meticulous down to checking if there are space characters missing between enclosing braces (i.e., $\{puts\ "Hello\ world."\}$ versus $\{\ puts\ "Hello\ world."\ \}$). In general, I was just very amazed by how resourceful they were in

terms of automating what I naively thought would need to be done manually. In my opinion, this was definitely one of the more important things I learned through this internship – that things like these exist.

With the knowledge I had accumulated in school along with these newly acquired skills, I was able to complete a number of tasks. It started with the easy ones, which were to simply change labels using HTML and CSS. Then there were the moderate ones: editing a shell script, making sure the user's IP blocks were shown as blocks and not just a single gateway IP, and – although not an actual assigned task – with command line, setting up their API locally so that it works in conjunction with their frontend code already set up locally on my machine (in order for me to test the code I write). Finally, there were the really difficult ones: catching fraudsters attempting to reuse SSH keys and making sure IP addresses were being deleted and created only as needed during deprovision. These last two were especially hard to me because it required understanding of something else in addition to knowing the language they were coded in, which was why it took me more than a month to complete each.

As a result, if I were to count the number of tasks I accomplished, quite honestly, there weren't many. Whenever I reflect on this, I start to ask myself if I'm really suited for this field, because I know for sure some of my classmates would be able to finish those in a much shorter time. Not to mention, these tasks that were assigned to me are only the easy ones that the full-time employees are too busy to want to spend brain cells on. Yet, to me, they are among the most demanding assignments I've ever encountered.

But then I remind myself that that is why I am there – to learn, and more importantly, to find out what I need to work on. That thought, rather than comforting me to stop worrying, is motivating me to work harder. That thought is what keeps me from giving up. As such, I will

focus this report more on what I discovered I am lacking rather than what I was proud of having accomplished.

My studies here at Stony Brook so far were akin to building blocks towards my work. There were a few times when these building blocks applied directly to what I was doing – the tasks with just HTML and CSS. But for the most part, they worked as a base off of which I expanded my knowledge. For learning AngularJS, I applied my knowledge of JavaScript, and for learning Ruby on Rails, I applied my knowledge of object oriented programming. That, along with using the code that was already written as reference, I am able to read and write basic code from the two.

However, I realized that I was actually heavily relying on these code references to learn. I found that out from the time the company's project manager assigned me a task from their new portal, which used React. As the new portal was still work in progress, there was barely any code for me to sample. As a result, I only understood the general idea of what was going on, but I couldn't put together all the pieces to form the complete puzzle (which is why I can't put React into the list of skills I learned). I also realized that I have been lucky in that I was able to find similar pieces of already existing code to serve as a template for what I needed to write.

While this method has gotten me through quite a few tasks (during the summer and this semester), I don't think this is enough. This is exactly like a person who is able to draw well by basing their illustrations off of references, but coming up blank when asked to sketch an original idea. I thought to myself, "There has to be a better way to learn new languages and frameworks easily, especially since new ones are pushed out every few years and everyone else is somehow able to keep up." This is the main reason that pushed me to register for CSE 307 (Principles of Programming Languages) next semester. I remember just last year I was reading the course

description and thinking how boring such a class was, but now I am anticipating it more than ever. Hopefully, after this course, I will be able to spot the right similarities more easily, having grasped the basic concepts of programming languages.

Then comes the second flaw that I discovered: my coding style is far from modular. When it comes to coding for homework assignments, I tend to just take the easy way out. For example, I have mentioned in a weekly report that in CSE 320, I had trouble with passing in arrays by reference and actually modifying the array passed in. So eventually, instead of writing a function for a repeated portion of my code, I just wrote same code inline repeatedly, just to make sure it works. I also tend to brute force my way through if I can't think of better logic at the time. This results in a lot of embedded *if* statements. Yes, this was exactly how I found out about what Rubocop tests for.

Thinking back, I have done this often in the past as well. In CSE 219, since it was a multi-part project, I ignored design patterns and wrote whatever worked at the moment, for the part that was currently due. In the end, my final product was a disastrous dish of spaghetti code. I also ignored the MVC pattern for the most part, which I now realize was a big mistake as I saw that (unsurprisingly) the company's code follows that pattern very closely. Then again in CSE 220, I wrote code inline instead of writing helper functions, since it is required in MIPS to manually store onto the stack the values of registers needed when the code returns from a *jump and link* to a function – including the *return address*! As such, there were times when the *jump return* returns to the wrong place despite the fact that I had stored the return address that was in the *$ra* register. So many times, instead of having to worry about checking if/where the stack has been overwritten, I chose to jump to functions as less often as possible, which again, introduces repetitive code. I had originally wanted to try writing better code for CSE 320 for the remainder

of the semester, but due to time constraints, I again reverted back to my old habits. So I made a compromise with myself to clean up my code after the semester ends so I could encounter the problems I would have had during the semester and work to correct my habits from there, in small steps.

Now, for more uplifting matters, I found that CSE 320 really helped me with my internship and vice versa. Actually, to be exact, it was CSE 220 that prepared me for my internship first, as I have to deal with a lot of command line for work. Prior to CSE 220, I had never done anything in command line. When it was required that we submit our homework through Sparky, I ran into trouble uploading my file from my computer to SSH because the tutorial provided by the TAs didn't work for me. As such, I had to search for an alternative route and in the process, I learned more than what was required, such as renaming files and overriding security privileges with *sudo*. The latter proved to be especially important for when I first started my internship in the summer, I had to install a lot of programs through command line.

Later on in the internship, I moved on to using Git for version control. This was probably the most useful thing I learned in preparation for CSE 320. There were a few times while working on my homework I wanted to revert to a previous commit because I discovered a bug that I don't remember having previously, but I only want to revert a specific file. I knew to do $git\ reset\ --soft\ HEAD\sim\#$ and then $git\ checkout\ --<file>$ because I had done so before at work many times.

As for how CSE 320 helped me with my internship, part of the process during provisioning servers at the company was installing operating systems. As I was learning the fundamentals of operating systems in the course, naturally, some of the terms I learned in class appeared in their scripts. That, in conjunction with my increased use of command line – at one

point, we even had to code our own shell – I was able to follow along with what was going on a bit more, being able to recognize terms such as *SIGTERM*, *SIGKILL*, etc. It also helped me with one task where I was asked to edit a shell script. Even though at the end, that task really boiled down to searching up the syntax for *if* statements in shell scripts, being able to read shell commands helped me feel confident in what I was doing was correct.

All in all, I am extremely grateful for this internship experience. Not only was I able to learn many new skills, but more importantly, I was able to realize my flaws. This will allow me work on them for the remaining time I study here at Stony Brook and make the most out of my courses.