Reflection paper

Steven Rosendahl

The project was concerned with introducing Ruby to LaTeX. The motivation for this project came from the want to have a more powerful way to execute dynamic code inside of LaTeX. LuaLatex does a good job of this, but Lua just isn't robust enough to do things like pull data from the web or even play a command line game every time that you compile a document. In addition, Lua can be difficult to setup on a computer, but Ruby is built to be cross platform with hundreds of libraries that are able to be installed with just one line in the terminal.

Throughout this project, I found myself looking up more information about LaTeX and the inner workings of TeX than Ruby. It was ultimately the complexity of TeX that made me decide to design RubyTeX to run outside of TeX, as opposed to the other way around. This design strategy was difficult to maintain; I was forced to write a lot of code built to parse TeX and Ruby syntax. I also decided to not use Regular Expressions, since it would have taken me too long to learn the Ruby way of doing so; if I were to re work this project, I may take the time to learn Regex in Ruby.

Originally, the processors were written in C++ since the code would be doing a lot of string processing and I wanted it to be fast. Unfortunately, I had to write a lot of C++ code to do very little, so I ultimately wrote all of the codebase in Ruby. I think this was a smart decision, since I was able to bundle the executable, `rblatex`, with gem, the Ruby package manager. Gem is a very easy to use package manager and I was able to include almost all the files that I needed to, with the exception of the LaTeX package file. I attempted to submit the `.sty` files to CTAN, but the package was rejected due to the lack of documentation included in the CTAN package. CTAN requires that your documentation be in pdf that you can create using a `.dtx` file. I decided to forgo doing this step, since the documentation will be in a Markdown document on Github.

One of the most interesting parts of this project was the fact that I could actually use my project while working on other parts of the Capstone. The log books, for example, are dynamically generated so that I could sum up the total amount of hours worked. The poster was written using `tikzposter`, so I was able to use my package to generate some of the content on the poster as well. Using my project outside of the Capstone course helped me to test it. I realized, for example, that while writing a paper for another class, loops were not correctly translated. I may not have come across this edge case without having applied the project outside of this course. I will also be able to continue using this project outside of my undergraduate studies.

This Capstone project helped me to realize how much time it takes to build a piece of software. There are a lot of moving parts to RubyTeX that were often hard to manage all at once. Some bugs in the program took days to fix, while others took a minute or two. At the beginning of this course, I thought that this project would take me a lot less time that it has, and I had many aspirations that were not able to be implemented because of this. The program does fulfill what I set out to do, so it has been a success.