Leveraging Communication: Learning and Achieving in an Agile Workplace

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Ryan Pope

Spring, 2022

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Sean Ferguson, Department of Engineering and Society

Rosanne Vrugtman, Computer Science

Leveraging Communication: Learning and Achieving in an Agile Workplace

CS4991 Capstone Report, 2021

Ryan Pope
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
pope@virginia.edu

ABSTRACT

During my entry-level internship at Yext, a DC-based search engine optimization and consulting software company, I led a summer-long project to migrate Yext's consulting site building process. I leveraged the scripting efficiency of Go to create a comprehensive repository (repo) migration tool, allowing existing company repos to integrate seamlessly with Yext's continuous integration (CI) tools. This innovation saved significant amounts of time in the site development process used company-wide. Working on this project also helped me to develop numerous soft skills, especially communication and project management. I frequently interacted with teams on both sides of the company and was able to achieve a serious impact through positive and frequent communication.

1 INTRODUCTION

Over the summer of 2021, I worked as an entry-level consulting software engineer in a team-based technical environment at Yext, based in Arlington, VA. Because Yext allowed interns to make a serious impact on the company's products, I was able to frequently publish my code to production. Additionally, at the beginning of the summer, Yext gathered all the software engineering interns around the company and gave us a pool of research & development projects to choose from. These projects ranged from Slack bots to announce pull requests to AWS Lamba function monitors, to my project – migrating Yext's internal build tools to a continuous integration (CI) system. Many of these research ideas had been floating around the company for months or years, as the company doesn't have much time allotted for full-time employees to pursue these projects simultaneously with their sprinted work. In this vein, I was very excited that I was making a lasting and relevant impact on the company, leveraging my time as an intern to complete a project that's been in the queue for quite a while. My main goal for the summer was to complete or get as close to completing my project as possible, so that either (a) the company could begin capitalizing on my contribution as soon as possible, or (b) I could leave the project at a good point for another intern or full-time employee to pick it up and complete it.

The project was a bit daunting to me at first, because I had little experience taking total charge of a long-form project like this one and was also wildly unfamiliar with the technologies Yext was using in their continuous integration system. My approach from the start was to foster consistent communication between myself, the teams responsible for the internal tools I'd be leveraging, and the subject matter experts (SMEs) for my specific project. Luckily, my workplace used Slack, so it was incredibly easy to contact anyone I needed to communicate with in the company. Yext had channels for every team, project, and general topic, and so I was frequently able to simply search for the channel to post my question or comment in.

2 BACKGROUND

In Yext's consulting department, there are two main focuses – creating search experiences using the custom Answers search engine, as well as providing consulting services to major companies, designing their site applications, making changes, and adding features as necessary. The motivation for the consulting services side ('Pages') is to guarantee to the companies which contract with us always appear as expected on major search engines, so that when a location's information is updated, it appears updated everywhere at once. This is facilitated by making use of Yext's centralized database, otherwise known as the "Knowledge Graph." Companies submit new data to the Knowledge Graph, and this data is updated across all site pages multiple times per hour.

To ensure data is always fresh, Yext makes heavy use of templating languages to generate frontend apps, chiefly Google Closure Templates (Soy). Each dynamic page on a site is composed of one or more Soy templates, which are later compiled locally by engineers into JavaScript and HTML, to be reuploaded to Yext's servers and served to the clients' customers.

3 RELATED WORKS

In planning my project, the first step I took was ensuring that I was aware of any existing documentation that could help me better understand the goals of the project. Many of the resources I received from the SMEs and product engineers was internal (and

confidential), but I was also helped greatly by the extensive Go documentation available online, and found that almost any question I had about Go could be answered with a simple search through the official documentation. Additionally, I was encouraged by my mentors to follow the principles of the Clean Code Handbook, and so I very rarely was asked to revise the structure/readability of my code. I feel that my project would have certainly fell short of its goals without the help of this existing documentation, and so I was very grateful for Yext's emphasis on leveraging related existing works.

4 PROJECT DESIGN

The motivation for my summer project was that almost the entire site-building process for these consulting sites relied on the engineers running scripts locally and immediately publishing the outputs of those scripts ('build artifacts') to GitHub along with the source files. This was a huge time sink for the engineers, as well as for some larger clients, since local builds could take up to 15 minutes and were very resource-intensive. Additionally, this resulted in an over-usage of storage in GitHub, as build artifacts were directly related to source files, but existed in the same repository and were not generated dynamically.

In the Answers-focused part of the company, engineers had already been making use of Yext's custom continuous integration tools to build the search experience frontend. These applications were often small and build artifacts could be produced reliably inside a remote container (via Docker). For this reason, Answers apps were already 'CI-enabled,' meaning in their repo's configuration JSON, there were working instructions for the CI to create build artifacts. This meant that when engineers needed to make changes to Answers pages, all they needed to do was publish source files to GitHub, and the CI would trigger automatically and generate the build artifacts to serve. Because of the disconnect between the Answers build process and the Pages build process, the SME for my project decided that it would be a worthwhile investment to leverage my time as an intern to try to get Pages builds working inside the CI container, similarly to the way Answers artifacts were generated.

4.1 OBSTACLES

From the start, there were several obstacles to getting the Pages process working inside a Docker CI container. The first and most daunting obstacle was that the structure of the Pages repos was not uniform at all. Yext has been offering Pages since around 2012, and so naturally the way these sites are organized, as well as the stack they use has evolved over time. This meant that to create one cohesive solution for migrating Yext's repos, I had to account for every single iteration of the site structure. This became a serious challenge because of the wildly different stacks that were used over time – some sites were still using Coffeescript and vanilla CSS, whereas the modern standard was JavaScript and SASS.

Additionally, when I started my internship, Yext was in the process of open-sourcing all of its template Node dependencies, but this effort was not yet complete. This would not have been an issue, except that the Docker containers used for continuous integration were not allowed to have any logins/ secrets stored for security reasons. This meant that any Node dependencies that were usually only available internally to Yext engineers were not visible to the CI – which meant practically all Yext's modern sites could not be built by CI.

The final major obstacle I encountered was the novelty of the internal CI tools. Only a couple of months before I started, Yext was still using Jenkins (a third-party CI tool) and so the likelihood of bugs and flaws in the internal tools was quite high. I knew I would have to frequently and effectively communicate with the teams responsible for the CI to work out these bugs. Fortunately, I saw this as an additional opportunity to create a positive impact at Yext. I was not only streamlining the build process but also improving the quality of Yext's products.

4.2 RESEARCH/PLANNING

In developing my plan for this project, I first had to understand the input set of repositories I was working with. Because these repositories were centrally available in GitHub, I was able to make use of GitHub's API as well as Sourcegraph to determine as much as possible about these repos. By querying for files inside of repos, I was able to generate rough counts of how many repos had certain conditions, such as presence of Coffeescript, custom modules, or usage of private Node dependencies. This allowed me to calculate approximate 'coverage' of Pages sites – in other words, an estimate of how many sites I could migrate with my project, without manual tweaking.

Communication was truly key in the planning process. I was having almost daily meetings with my project's SME to bounce project ideas off him, and to confirm my findings in my research. The SME also helped me to understand some of the existing Yext tools I could leverage to carry out a batch migration of repos. The main tool I decided to use for the migration was an internal tool called Bulku. Essentially, its purpose was to take a list of Pages repos, mutate them, and push the repos back to remote.

Once I felt technically prepared, I laid out a week-by-week plan for steps in the process, as my goal was to complete the project within my 12-week internship. I then presented this schedule to both my team lead and my manager and requested their feedback to gauge how realistic my expectations were. Once I had incorporated their feedback into my plan, I was confident I could finish the project within the summer

and began creating sprint tickets for myself to track my progress.

The second component of my initial research and planning was to get a single test site working in the CI. To choose the test site, I contacted several senior engineers at the company to find the site with the highest possibility of CI failure. I chose this because I figured starting with the largest challenge would diminish the amount of effort required to adapt the migration to other sites. There were already a couple reference examples to work with, as we had manually enabled CI for a couple clients in the past. I used these sites' configurations as a starting point and began manually tweaking the configuration. This tweaking ended up leading to not only the optimal CI-enabled configuration for Pages sites, but also exposed dozens of existing bugs within the internal tools. Throughout the process of developing the configurations, I was having daily meetings with members of different product department teams to discuss bugs I'd been experiencing, replicate the bugs on their end, and kickstart their process of implementing a solution. This led to a quick ticket turnaround, and all process-breaking bugs were resolved within a week of report.

4.3 DEVELOPMENT PROCESS

Once I had confidence in my ability to enable CI, based on my experience with the test site, I began to explore live repos, some of which were our largest clients. I began making local copies of these repos and experimenting with their configuration files. After doing this a few times, I had refined the process down to discrete steps, which I made note of for later use in the Bulku scripting for the migration. Each time I tested a site's configuration, I would spin up a CI docker container and ensure that valid artifacts were generated. Iterating this process, as with my initial research, provided more insights into bugs and shortcomings of the product software.

Once I had refined the steps of the process and was confident in my ability to translate those steps into components of a Go migration script, I scheduled a meeting with the SME for my project. I discussed each step with him in detail, and received his feedback, as well as several existing examples of similar scripts that had been used within Yext. This was an incredibly valuable resource, as many of the mutations I was looking to perform on the Pages repos were already supported in the default Bulku library. This streamlined the script development process greatly, and I was able to produce a preliminary copy of the script within 3 weeks.

This preliminary copy was designed to work on the largest subset of our Pages sites – those developed within the last 2 years. This subset was the easiest to deal with since the build tools had somewhat stagnated since then – anything before then, the tools were being updated quite frequently and so structures could vary widely. I began writing tests for the migration, as well as manually testing migrated sites within the Docker container.

In the weeks after the completion of the preliminary copy, I scheduled several meetings with the teams who would be using my tool and began to ask them what features they would want most in the CI migration. I requested insight into which sites could be special cases, to determine if I could include them in the coverage of my migration script. One of the most common features I had requested was a GitHub integration for the completed migration. Essentially, the feature would detect if a repo had been migrated successfully, open a pull request between the migrated branch and master, and send that pull request to the appropriate team. This communication was critical, as I had a general idea of what the teams would want, but I ended up being able to provide 'want's instead of just 'need's.

I was able to implement this tool using GitHub's command line API, and so as the Bulku script went through sites to migrate, it would compile a list of pull requests to review. This streamlined the process of testing and review, and I was able to receive feedback and bug reports as soon as possible when testing new sites. Encouraging consistent communication throughout the development process allowed me to not only quickly achieve the goals I laid out in my plan, but also to work with my colleagues to create new goals for the rest of my internship. These features ranged from small things such as special case support, to large things such as a dynamic tracking spreadsheet to monitor migration status of all Pages sites in one place.

4.4 DOCUMENTATION

One of the most challenging portions of my internship was developing the documentation for my project. Documentation is a strange form of communication in that it's sort of a one-way conversation. Naturally, I struggled to develop answers to questions that hadn't been asked yet. I decided to explain as many lines of code as possible and ensure that I used good coding practices so that future engineers could integrate their work with mine easily. I also scheduled weekly meetings with my manager and SME to gather input from them on the quality of my documentation. Once my internship had ended, I had created several pages of quality, effective documentation and felt confident that future engineers could easily understand my code and pick up where I left off if necessary.

5 RESULTS

At the end of the summer, I was asked to present my project to the entire consulting side of the company. I leveraged my existing documentation to create a comprehensive breakdown of my project, and received positive, constructive feedback from my peers regarding the project. I followed up with my manager several months later, and he informed me that almost half of the sites were completely migrated using my

tool, and that he estimated about 20% of Pages development time was saved by the introduction of CI.

6 CONCLUSIONS

In terms of my skills, I felt far more confident in my ability to leverage the power of communication by the end of the summer. Throughout the development of my project, I found that communicating with my colleagues, whether through weekly meetings, surveys, or Slack messages, accelerated my learning process greatly, and provided a profoundly greater impact than I expected coming into the job.

7 FUTURE WORK

Although I accomplished all my goals in developing this project, one step I wish I could have seen into action was the actual deployment of the migration script. Unfortunately, my internship was only for the summer, and I went back to school before this step could happen. After I left, no sites had yet been migrated, but members of my team and others were very excited to begin using my tool. I would have loved to be able to oversee the deployment and provide any insights to the team as necessary. Additionally, throughout my project I was able to solve numerous bugs in the Bulku migration system as well as the CI and recognize that there is still plenty of work to be done in optimizing those systems.

8 UVA COURSE IMPACT/NEXT STEPS

The two classes which contributed most to my success as an intern were CS3240 Advanced Software Development and CS4750 Database Systems. Both classes had a long-form project component and my experience with those greatly helped me develop the plan for my summer R&D.

One shortcoming of the project environments in these classes, though, was that they cannot truly simulate a workplace. The amount of communication required to achieve goals in a workplace is far higher than that of a four-person group, and I did feel somewhat unprepared for that coming into my summer job. Additionally, in CS3240 we were required to write quite a lot of code, but there wasn't much of an emphasis on documentation. At Yext, I unknowingly was expected to write documentation for nearly all of my contributed code, and because I had not had much documentation practice yet, I had quite a steep learning curve.

One major improvement that could be made in these classes is trying to better simulate the work environment, or at least incorporate into the course material some best practices for commonplace workplace expectations. This could help students feel more confident when entering the workplace and would certainly lead to higher efficiency and progression of learning.

REFERENCES

1. Go Documentation. Go. (n.d.). Retrieved November 16, 2021, from https://golang.org/doc/.
2. Martin, R. C. (2009). Clean code: A Handbook of Agile Software Craftsmanship. Prentice Hall.