USAGE PRE-COMPUTATION

COMPARISON OF AUTOMATION IN SOFTWARE ENGINEERING AND MANUAL LABOR JOBS

A Thesis Prospectus
In STS 4500
Presented to
The Faculty of the
School of Engineering and Applied Science
University of Virginia
In Partial Fulfillment of the Requirements for the Degree
Bachelor of Science in Computer Science

By
Jared Nguyen

November 1, 2021

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.

ADVISORS

MC Forelle, Department of Engineering and Society

Daniel Graham, Computer Science

Introduction

Automation is seen as a danger to the workforce, where many fear towering robotic arms will replace the workforce en masse ("Emerald Insight," 2018). Though the word "automation" may bring up such imagery, automation is any technology that reduces human input with some self-regulating method (International Society of Automation, 2022). It can be as simple as an oven that automatically controls the internal temperature that a user has set to as complicated as a self-driving car. Truck drivers, cashiers, cleaners, and more are jobs considered at risk of being automated by technology. Criticisms of automation date as far back as the industrial revolution. Luddites, a group of English textile works from the 19th century, were known for rejecting weaving machines as, "mechanisation would threaten their livelihood and the skills they had spent years acquiring," which led them to, "destroying weaving machines and other tools as a form of protest against what they believed to be a deceitful method of circumventing the labour practices of the day. " (Historic UK, 2018) Similarly, in a recent survey, more than half of respondents felt that, "In the next 50 years, robots and computers will do much of the work currently done by humans." ("Emerald Insight," 2018) Oxford University study predicts that about 47% of the total US employment is at risk of automation, where jobs with repetitive, well-defined procedures are at highest risk (Benedikt & Osborne, 2013). Comparatively, in the software engineering industry, concepts like continuous integration/continuous delivery, which is a methodology that automates the building, testing, and deployment of new code into production applications, exist to reduce manual labor from developers testing new code changes and ensuring consistent development environments. The push for automation has resulted in the creation of new job roles such as DevOps engineers and test automation engineers that both focus on automating the delivery of new code and programmatic tests respectively (SentinelOne,

2017). Though there are many benefits of automation in manual labor jobs such as increasing throughput, consistency, specialization, and decreasing redundant work, there are major disadvantages such as large initial capital cost, displacing workers replaced by automated machines, and potentially increasing unemployment. Automation seems to inordinately benefit massive corporations by increasing profit margins through larger output and removing the necessary safety and benefits required for humans, all the while reducing the available jobs for the workforce. In contrast, automation in the software engineering seems beneficial, serving as a strong complement to the worker by reducing cognitive load and ensuring consistency. Can similar negative effects of automation appear in a highly specialized field like software engineering? As a result, this STS project will make a comparative case between software engineering and manual labor jobs as a means of assessing the dichotomy of automation benefiting or hurting workers. By doing so, it will attempt to alleviate the intense negative sentiment surrounding it in many manual labor jobs by determining the risk factors of automation and will outline polices that can help mitigate the risk.

Technical Topic

A major international company that focuses on retail, ecommerce, advertising, streaming, and cloud computing provides a service that stores users' files. The project was derived from the core backend of the service, where the focus of the project was optimizing an Application Programming Interface (referred to as an API) that returns customer file usage. An API is defined as a way for pieces of software to interact with each other in a documented way[1]. The current implementation of the API required a given customer's file usage to be computed by aggregating individual file sizes, essentially "counting" each individual file per customer. Using

runtime analysis, this can be described using big-O notation as an $O(n)$ operation (*Big O Notation | Interview Cake*, 2022), where the number of computations grows linearly with the number of files a user has. As a result of the growing userbase of the service, an amount on the tens of millions, this approach did not scale in a way that provided a pleasant user experience, both externally to the customers and internally to other services within the company. The data store that performed the aggregation computation was used by a multitude of other services, was under intense load and regularly triggering alarms over latency issues. Alarms in software engineering are used to either send notifications or trigger actions in response to a change in metrics (*Effective Alerting in Practice*, 2022). Attempts to reduce user perceived latency already existed in the form of a caching mechanism, where user usage values existed for several minutes in an external data store, before being considered stale and requiring a re-computation of values.

A major issue with the current implementation was a result of the caching mechanism. In the worst-case scenario, users could potentially update or delete a file and not see change in their usage until several minutes passed and their values were recalculated. Another major issue was related to latency; it was reported that latency metrics would soon exceed common recommended load times of around 0-4 seconds (Baker, 2022). The proposed technical project solution focused on removing the dependence on the stressed data store. Its main goal was to precompute the user usage values instead of relying on the "on-the-fly" aggregation. This would be done by maintaining a running total of usage values per user stored in a new external data store, then examining each individual file upload or delete to increment or decrement their usage values. Though having these changes in the production service was out of scope of an internship, there were still many challenges that occurred. One technical challenge that the project faced was dealing with race conditions. Race conditions occur when multiple processes attempt to operate

on a shared value, in this case, the shared value being the user usage. Another challenge that occurred was integrating previous customer usage with the new solution. This would involve a process called backfilling, which is simply defined as changing data in bulk (Fly.io, 2021). In this context, this involved initializing a value for existing customers in the new external data store before processing incoming user file uploads and deletes.

STS Topic

Failing to address these issues of automation can, in the worst case, puts 50% of the US workforce at risk to be displaced or hurt by automation (Brown, 2020). By comparing the two cases, this STS project hopes to the risk factors and driving forces of automation in the two industries. Manual labor jobs that are highly at risk for automation are characterized by repetitive, well-defined movements, which makes them targets for automation. A main force that drives automation is labor-related costs. The same driver also causes "fissuring," which is defined as replacing employees with outside contractors (Estlund, 2018). This usually takes the form of outsourcing work to other countries for cheaper labor and less strict safely laws. Automation in this use case is inherently political because it serves as a means for those, like corporations, hoping maximize revenue. A deliverable of this STS project is to identify potential policy that will help mitigate risk for those who could be displaced by automation. A possible policy change is some forms of universal income or worker up-skilling program. This has implications into the rate that jobs are being automated, where workers may not be able to upskill fast enough to outpace automation.

In contrast, as a case study that automation will be mutually beneficial for workers is in software engineering. Automation in software engineering strongly complements the labor that developers perform throughout the entire software lifecycle. Software engineers utilize

automation to run test suites that determine the correctness of defined behavior in their code. DevOps is a methodology that uses automation to accelerate the delivery of code in a way that is reliable and secure. In a case study of a growing software startup by Shestakofsky, automation was used to help manage organizational processes and reduce engineering burden. During periods of focused on fast growth and revenue generation, tasks that were automated allowed manager to continually reconfigure the usage of developer resources to perform tasks that had yet to be automated or was unable to be automated. Notably, the process of automation itself required significant engineering work to perform, which often deemed too labor intensive in early phases of growth. (Shestakosky, 2017).

A major STS framework that will be used in the project is Winner idea of political technology. Political technologies are designed in a way that technologies like, "industrial production, warfare, communications, and the like have fundamentally changed the exercise of power and the experience of citizenship." In the context of automation, this framework will be used to analyze how automation is used to circumvent costs related to labor laws, in that the "power" of workers has shifted away to corporations, as workers are displaced by automation machines are outsourced to other workers.

Conclusion

The deliverable for the STS technical portion was an initial implementation of an optimization of an API. By precomputing the usage values, essentially converting the operation to a database read, the proposed implementation was estimated to reduce the number of calls to the stressed datastore by approximately ten million and reduce worst case latency to a scale of tens of milliseconds. Through this technical project, I hope to relate it to the STS portion as a case study to analyze how the project was benefited by automation in terms of developer work

and ensuring correct behavior. The STS research hopes to further understanding of the realistic risk of automation, especially in regards to public perception.

Work Cited

sematext. (2020, July 29). *The Complete Guide to Metrics, Monitoring and Alerting*. Sematext; Sematext. https://sematext.com/blog/monitoring-alerting/

Acemoglu, D., & Restrepo, P. (2018). Artificial Intelligence, automation and work. https://doi.org/10.3386/w24196

Arntz, M., Gregory, T., & Zierahn, U. (2017). Revisiting the risk of automation. *Economics Letters*, *159*. https://doi.org/10.1016/j.econlet.2017.07.001

Bainbridge, L. (1983). Ironies of automation. *Analysis, Design and Evaluation of Man–Machine Systems*, 129–135. https://doi.org/10.1016/S1474-6670(17)62897-0

Baker, K. (2022, April 7). *11 Website Page Load Time Statistics You Need [+ How to Increase Conversion Rate]*. Hubspot.com; HubSpot. https://blog.hubspot.com/marketing/page-load-time-conversion-rates

Benedikt, C., & Osborne, M. (2013). *The Future of Employment Published by the Oxford Martin Programme on Technology and Employment*. https://www.oxfordmartin.ox.ac.uk/downloads/academic/future-of-employment.pdf

Coupe, T. (2019). Automation, job characteristics and job insecurity. *International Journal of Manpower*, *40*(7), 1288–1304. https://doi.org/10.1108/ijm-12-2018-0418

Coupe, T. (2019). Automation, job characteristics and job insecurity. *International Journal of Manpower*, *40*(7), 1288–1304. https://doi.org/10.1108/ijm-12-2018-0418

Dodel, M., & Mesch, G. S. (2020). Perceptions about the impact of automation in the Workplace. *Information, Communication & Society*, *23*(5), 665–680. https://doi.org/10.1080/1369118x.2020.1716043

Estlund, C. (2018, November). *What should we do after work? Automation and Employment Law*. The Yale Law Journal - Home. Retrieved October 18, 2022, from https://www.yalelawjournal.org/article/what-should-we-do-after-work

Fly.io. (2021). *Backfilling Data*. Fly. https://fly.io/phoenix-files/backfilling-data/

Historic UK. (2018). *The Luddites - Historic UK*. Historic UK. https://www.historic-uk.com/HistoryUK/HistoryofBritain/The-Luddites/

International Society of Automation. (2022). *What is Automation? - ISA*. Isa.org. https://www.isa.org/about-isa/what-is-automation

Interview Cake. (2022). *Big O Notation | Interview Cake*. Interview Cake: Programming Interview Questions and Tips. https://www.interviewcake.com/article/java/big-o-notation-time-and-space-complexity

Karhu, K., Repo, T., Taipale, O., & Smolander, K. (2009). Empirical observations on software testing automation. *2009 International Conference on Software Testing Verification and Validation*. https://doi.org/10.1109/icst.2009.16

Law, J., & Woolgar, S. (1991). Configuring the user: the case of usability trials . In *A sociology of Monsters: Essays on Power, Technology and Domination* (pp. 57–99). essay, Routledge.

Redhat. (2022). *What is an API?* Redhat.com. https://www.redhat.com/en/topics/api/what-are-application-programming-interfaces

SentinelOne. (2017, August 21). *The DevOps Job Market | Scalyr*. SentinelOne. https://www.sentinelone.com/blog/devops-job-market/

Shestakofsky, B. (2017). Working Algorithms: Software Automation and the future of work. *Work and Occupations*, *44*(4), 376–423. https://doi.org/10.1177/0730888417726119

Winner, L. (1980). Do Artifacts Have Politics? *Daedalus*, *109*(1), 121–136. http://www.jstor.org/stable/20024652