Computer Science Curriculum Addendum For Production Development Practices

User Configuration in Widespread Algorithms and Its Effect On Society

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 of the Degree
Bachelor of Science in Computer Science

By
Aditya Penmesta

October 24, 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.

Signed: Aditya Penmesta

ADVISORS

Benjamin Laugelli, Department of Engineering and Society

Mark Sherriff, Department of Computer Science

Introduction

Since the turn of the century, the exponential rise of computer software technology has created a vast job market for new graduate software developers. These graduates are tasked with critical assignments, many of which directly affect the bottom line of companies and provide tremendous value to the project's stakeholders. However, as new technologies branch off of core computer science concepts and continuously adapt, it's becoming apparent that students are not prepared in college to produce production level code that encompasses a solution without bias. From my own experience, I had the opportunity to intern during the summer at Flowcode, a startup based in New York that specializes in direct to consumer products such as QR codes, mobile first landing pages and interactive TV ads (Flowcode), and was tasked with numerous coding intensive problems. Although classes I had taken at UVA helped me understand crucial technical concepts, I felt like my knowledge of the production level of development, how one builds technology that can be used by their intended stakeholders optimally, fell short. This mindset is simply not covered within the scope of the UVA curriculum, meaning most students, like myself, undergo significant learning curves in their first professional experiences, setting them behind in the beginning of their career. Additionally, as machine learning and other artificial intelligence courses are now being offered to all, the consequences of bias in these technologies' designs are not fully covered, leading to many end users being improperly classified by poorly gathered data and assumptions off of limited knowledge by engineers. Specifically, the case I want to focus on is the US Healthcare machine learning issue, where engineers trained an algorithm on incorrect assumptions, resulting in needed healthcare coverage being less likely to be assigned to minority groups. By being inadequately prepared in college, new graduates can develop nonoptimal code that doesn't serve the end users appropriately,

causing extreme downstream consequences. Thus, it's vital to make addendums to the UVA curriculum which teach workplace development practices and foster a user based product design mindset, better effectively preparing students for jobs post graduation. In the technical portion of my paper, I will propose a course of action for better preparing students for the reality of a professional development environment. In the STS research paper, I will analyze how the US Healthcare system adopted a product with embedded biases which negatively affected users, and hope to incorporate these insights into the course proposed in the technical paper.

Technical Project

In order to better understand the technical gap I describe, it's important to first define the difference between production level deployments and the development environment. When initially building software, a development environment is used to construct a solution and test it out with no repercussions to the overall product. Once the code has reached an acceptable level, a developer must start pushing it to production, where additional steps are taken to ensure the security and efficiency of the product (Umbraco). During my internship, I created numerous API scrapers, programming scripts that hit another company's data endpoints with specific credentials to receive back information relevant to the organization, that pulled data from applications Flowcode used as a whole, and rendered that data in a website. Most of the knowledge I needed to think of my solution was provided to me by the UVA CS curriculum, as web development and testing is covered in multiple classes, which allowed me to finish the development portion of my internship in a relatively short timespan. However, the production component of my project took up the rest of my internship and had the steepest learning curve. Specifically, I had to remove security credentials from my development code and create parameter stores, services that store

sensitive information that can be accessed programmatically, in AWS (Amazon Web Services) to feed this information into my code. I also had to create numerous “health checks” within my service, so if for some reason an endpoint stops working, a new one is created that can ensure that the product is not down in the long term. After ensuring these factors, I had to deploy my code to Kubernetes, an open source technology used to manage and run virtual containers (Kubernetes, 2022), and debug numerous errors that appeared along the way. All of these production issues were new to me since they had not been covered in the classes I had taken, making it extremely difficult to both learn and utilize these practices concurrently, delaying the completion of my project. If added to the curriculum, a production development course would significantly enhance the career prospects for future students, making job candidates more experienced than the average entry level software engineer, thus helping them stand out amongst an ever increasing crowd.

In order to prepare students better for practical professional software development practices, I believe more educational content should be provided in the form of a consolidated class which primarily provides integrated assignments. UVA’s closest class to offering actual development practice teachings is CS 3240, otherwise known as Advanced Software Development. In this class, “methods for specification, design, implementation, verification, and maintenance of large software systems” are taught, falling short of the deployment aspect key to a production project (Sherriff, 2022). This trend is largely the same for most colleges, however a great example of software skills being taught can be found in professional coding bootcamps. The objective of these bootcamps is to prepare students with no previous coding experience for a professional development environment in as little time as possible, thus making the teaching of production coding essential. In the MIT xPro Full Stack Development bootcamp, students are

both taught how to design full stack products and also deploy them using the most popular practices used in the industry in the form of numerous coding assignments (MIT xPro). In fact, a study found this the most effective way of molding effective software engineers, as computer science faculty said they believed providing project experience and real world context were essential for improving CS curriculum (Valstar, Krause-Levy, Macedo, Griswold, Porter, 2020). Thus by studying these bootcamps, UVA's curriculum can be adapted to cover the deployment process by emphasizing assignments that reinforce crucial development lessons.

In order to propose specific improvements into UVA's curriculum, more bootcamp curriculums and syllabi can be analyzed to determine practical assignments that can be implemented. Additionally, other college course equivalents of CS 3240 can be analyzed to determine what specific academic content could be added to fill the teaching gaps that on-hand bootcamps might not provide. By combining an analysis at these two levels, UVA's computer science curriculum can hopefully be improved to help students improve their career prospects by being more prepared at the onset of their software development careers.

STS Paper

In order to improve health systems, healthcare companies have turned to machine learning algorithms to better classify higher risk patients in order to provide high quality, efficient care. One such algorithm was implemented widespread throughout a vast system in the US, touching almost 200 million people, and used healthcare costs as the primary predictor of health risk amongst patients. This approach was thought to be efficient since a patient's costs were highly accessible and didn't require further processing, thus making it an easy predictor use in the machine learning model (Vartan, 2019). However, when further analyzed by a research

group, they found that black patients tended to receive lower scores, although high risk black patients turned out to have 26.3% more chronic illnesses than white ones. Thus, although black patients were more likely to have chronic illnesses, they weren't classified high risk appropriately. This was primarily due to healthcare costs as a primary predictive variable; since black people tend to have a lower income, they tend to incur lower costs, thus improperly flagging some of them as low risk patients. The researchers found that correcting this bias by using different variables to classify health risk would increase the percentage of black patients receiving additional care from 17.7% to 46.5%. (Obermeyer, Z., Powers, B., Vogeli, C., Mullainathan, S., 2019). The massive discrepancy that the internal bias introduced in this model caused underscores the importance of effective design practices. Although it should be obvious that bias shouldn't be included in models and should be properly vetted, most companies are indifferent to the downstream consequences and still push designers to build upon their designs. Thus, compounding organizational bias restricts designers and affects users, in this case causing racial bias that potentially missed necessary healthcare for minority patients. By further analyzing controversial algorithms, I can draw key insights into how different development mistakes led to downstream societal consequences, thus providing lessons on how to effectively control bias in algorithmic design. Drawing on the idea of user configuration and using the healthcare case, I argue that the primary actors involved in algorithm development can embed their biased ideas of their end user into the product they created, causing their software to disproportionately represent their intended user identity.

User configuration is the idea that when designers are building a product, they try to define and delimit a user's actions, and thus attempt to configure a user, in order to better understand their end goals (Woolgar, 1991). To compound upon this configuration, sometimes

designers have to follow organizational procedures, making the designers themselves constrained in their implementation, thus further implicating the users to further bias (Oudshoorn, Pinch, 2003). By confining users to a box and not adapting their thinking, organizations and designers can misrepresent them, causing downstream problems that affect both their product and overall society. In order to support my argument that both designers and organizations embed their restrictive biases onto users, I will find additional examples that further illustrate the user configuration and bias prominent in other well known algorithms that served a broad audience, such as healthcare, and negatively affected users. I will then use scholarly articles to understand the social impacts improper user configuration can have, and use that information to further prove my primary argument.

Conclusion

In my technical report, I hope to come up with a new computer science course which could be integrated into the curriculum at UVA that generates key knowledge for students and fosters creative thinking around production development problem solving, better preparing graduates for the professional world. In addition, I hope to gain understanding from my STS research into how developers misconfigured their users due to poor design choices and caused downstream negative effects, which would provide insight into how to effectively teach bias controlling methods to others to avoid such consequences. Hopefully, these lessons will be incorporated into the new course I propose, emphasizing the importance of iteratively checking that the user is properly reflected and ensuring that they can be optimally served with the best production coding practices. By doing so, students can deliver complete and socially beneficial

products in their experiential learning events, resulting in benefits for both their own career prospects and the wider society their technology serves.

References

Flowcode. (n.d). *Flowcode.* Retrieved October 24, 2022, from https://www.flowcode.com.

Umbraco. (n.d). *What is a Development Environment?*. Retrieved October 24, 2022, from https://umbraco.com/knowledge-base/development-environment/

Kubernetes. (2022. July 25). *Overview.* Kubernetes Documentation. Retrieved October 24, 2022, from https://kubernetes.io/docs/concepts/overview/

Sheriff, Mark. (2022). *Course Syllabus.* Retrieved October 24, 2022 from https://f22.cs3240.org/syllabus.html#course-description

MIT xPRO. (n.d). *MIT xPRO Coding Bootcamp | Full Stack Online Coding Course.* Retrieved October 24, 2022 from https://executive-ed.xpro.mit.edu/professional-certificate-coding/?thank_you=true

Valstar, S., Krause-Levy, S., Macedo, A., Griswold, W. G., & Porter, L. (2020). *Faculty Views on the Goals of Undergraduate CS Education and the Academia-Industry Gap.* Retrieved October 24, 2022 from https://dl.acm.org/doi/pdf/10.1145/3328778.3366834

Vartan, S. (2019, October 24). *Racial Bias Found in a Major Health Care Risk Algorithm.* Retrieved December 8, 2022 from https://www.scientificamerican.com/article/racial-bias-found-in-a-major-health-care-risk-algorithm/

Obermeyer, Z., Powers, B., Vogeli, C., Mullainathan, S. (2019, October 25). *Dissecting racial bias in an algorithm used to manage the health of populations.* Retrieved December 8, 2022 from https://www.science.org/doi/full/10.1126/science.aax2342

Wolgar, Steve. (1991). *Configuring The User: the case of usability trials.* Routledge.

Oudshoorn, N., & Pinch, T. (2003). *How Users and Non Users Matter.* MIT Press.