Software Engineering Essay

Joana Palés Huix

August 28, 2023

1 About me

I completed both a bachelor's and master's degree in medical engineering, focusing my master's on machine learning, a field that intrigued me deeply, despite never having delved into computer science or programming in a formal sense. Currently, I'm pursuing a PhD affiliated with AstraZeneca, and my research is concentrated on developing sophisticated deep learning tools tailored for the analysis of medical images, specifically on chronic kidney disease (CKA). CKA is a widespread condition that affects millions globally, and as such is important its early detection for effectively managing symptoms and slowing disease progression. My primary goal is to design deep learning algorithms that can precisely identify and categorize the level of kidney deterioration (stage of the disease) by analizing biopsy images. My work stems from the research of a fellow PhD student who successfully created similar tools using data from mice. However, my research is based on a dataset collected in a human clinical trial in the UK, which also includes valuable biomarkers and proteomics data. This dataset facilitates the establishment of networks that integrate various data modalities. The main challenge arises from the particularities between human and mouse anatomies, which require adaptations of the model during its development to extend findings to human cases. Furthermore, the data originates from different medical centers, introducing potential disparities and noise. To overcome these challenges, I'm employing advanced deep learning techniques to adapt the mouse models to human scenarios and address the variations stemming from different medical centers. In addition to my main line of research, in chronic kidney disease, I'm also actively participating in a project aimed at predicting the malignancy of ovarian tumors via ultrasound images. This initiative draws upon data collected from diverse centers across Europe and harnesses state-of-the-art deep learning methodologies to achieve diagnoses with physician level performance.

2 Main lectures

One concept that caught my attention in Robert Feldt's lectures is the distinction between validation and verification. As someone deeply involved in engineering research, my typical approach starts with identifying a problem or an intriguing idea. I then dive into the subject, seeking to learn as much as possible and crafting new tools along the way. Verification is all about ensuring our tool is constructed correctly, devoid of any errors. This step is considered crucial because even a small mistake can undermine our research, diminishing the impact of our results in the field. On

the other hand, validation is often overlooked. It involves confirming that the tool being developed is indeed the right fit for the intended task. This lack of attention to validation is a significant issue, particularly in fields like healthcare, where numerous promising tools often remain unused due to their misalignment with the needs of doctors and medical professionals. In one of the projects I'm currently involved in, which focuses on the detection of ovarian tumors, our primary goal is to create a tool that seamlessly integrates into hospital practices. This objective has necessitated multiple adjustments to our research direction. The ultimate aim is to produce something of genuine value for the medical doctors who will be utilizing it, ensuring alignment with their needs and contributing positively to patient care and diagnostic processes.

Another concept that captured my attention during the lectures was the so-called "Right BICEP" rule for testing. As someone relatively new to the coding world, the idea of testing with structure to systematically identify potential implementation errors is incredibly valuable knowledge. This rule encompasses six distinct steps. The initial step involves checking the results to ensure they are indeed *Right*, effectively aligning with the specified requirements. Next, it's essential to examine *Boundary* conditions to ascertain the tool's performance limits. Additionally, assessing the *Inverse* can offer valuable insights; verifying that certain input-output relationships hold true enhances the robustness of the tool. *Cross-checking* the results with an alternative algorithm can also highlight errors. Purposefully introducing *Errors* conditions and observing the ensuing behavior can expose vulnerabilities. Lastly, evaluating the *Performance* of the tool ensures that it meets the expected standards.

3 Guest's lectures

During Linda Erlenhov's lecture, we dived into AI/ML Bots within the context of their application in supporting Software Engineering purposes, often referred to as DevBots. These tools help with different tasks, ranging from code generation and review to documentation, all aimed at enhancing productivity, quality, and efficiency. To refine this explanation, it's important to first understand what constitutes a DevBot. The lecture emphasized that the definition of such bots is flexible and there exist different types. Some can engage with language complexity to varying degrees, possess intelligence, exhibit adaptability, autonomy, and even identity. It's worth noting that not all these attributes are mandatory for a bot, and the use of AI/ML is not a prerequisite. These tools, designed to simplify the lives of developers, also offer significant assistance to researchers like myself who are venturing into AI-driven code development for research purposes, yet lack a comprehensive computer science background and familiarity with software engineering best practices. These tools contribute to producing higher quality code and validating its functionality. This is key as, even though the core focus remains on the research itself, the quality and efficiency of the code cannot be overlooked, as they play a vital role in facilitating the research process.

The lecture also touched on the role of DevBots in augmenting developer productivity, a concept that has evolved beyond its traditional definition. Productivity now encompasses a multifaceted SPACE that refers to Satisfaction and Wellbeing (including health and contentment), Performance (measured through outcomes and impact), Activity (involving actions, design, and coding), Communication and Collaboration (both within teams and during onboarding, as well as documentation), and Efficiency and Flow (achieving tasks with minimal interruptions). In my viewpoint, this expanded notion of productivity extends beyond Software Engineering. Drawing from various

personal professional experiences, it's evident that results are closely linked to motivation and the work environment. Moreover, in my PhD journey, effective collaboration and idea sharing have shown to be vital, provided they are balanced with dedicated, uninterrupted work time.

4 Software Engineering topics

For this section of the essay, I will go into two topics that consider the human element in the field of Software Engineering. This is an area of research that intrigues me, as I have a natural interest in psychology. I believe that this aspect, which significantly impacts a worker's performance, has traditionally been disregarded, particularly in roles closely tied to technology.

4.1 Behavioral Software Engineering

Behavioral psychology promotes the idea that the environment shapes our actions. In essence, our surroundings teach us how to act. According to behaviorists, in order to alter behavior, the environment must be changed. This concept can be extended to various fields, including Software Engineering, where it consists in understanding of how various individuals or teams partake in the activities necessary for the process of software development.

At its core, software development involves humans crafting products for fellow humans. This inherently human aspect is integral and cannot be overlooked. There are many different aspects of psychology involved at different levels: there are individuals, each with their own unique personalities and motivations, working in groups, which can be more productive and creative when they work well together in a positive environment, and there's also the larger organization, which decides what's most important and what needs to be done. These levels, though seemingly distinct, are profoundly interconnected. In the prospective study by Sánchez-Gordón *et al.* [3], the concept of "voice and silence" is explored. *Voice* refers to the empowerment of individuals to voice their opinions and impact their work and the interests of managers and owners. This can prove advantageous as it aligns employee engagement with company objectives, creating a sense of ownership, and boosting motivation. On the other side, *silence* highlights circumstances where individuals either lack opportunities to express themselves or consciously choose not to, often due to apprehension or feelings of powerlessness. This silence can detrimentally affect performance. This study, which is a work in progress, proposes to examine the effects of psychological safety and perceived impact on voice and silence, and their impact on withdrawal (such as burnout) and performance. Such type of studies shows potential to increase people's performance and overall satisfaction by doing minimal changes in the companies policy. While reviewing related literature, it's surprising to observe the underexplored nature of this subject, given its clear influence on organizational success and ultimately, to companies economy. In my opinion, the subjectiveness of the topic and the absence of robust evaluation metrics likely contributes to this underrepresentation, suggesting the necessity to find objective measures to enhance motivation and teamwork. Besides, I believe that these topics extend beyond the boundaries of software engineering. Thinking in the context of my own PhD situation, which is similar as of that many others, motivation is key. While a PhD research often being a solitary pursuit, the relationships with mentors, peers, collaborative capacity, and the autonomy to steer one's research trajectory are essential. In fact, I believe that these factors may have even greater influence on the success of a PhD journey than the accumulation of knowledge

or the specific research undertaken.

4.2 Human aspects in Software Engineering

This next topic is closely related to the one discussed in the previous section. As mentioned earlier, software development is a process centered around people. Thus, it's only natural that all human-related characteristics impact this field. Human factors, in general, encompass environmental and individual human factors that influence behavior at work [1]. In essence, this field involves studying any human aspect relevant to Software Engineering, including personality, teamwork ability, communication skills, emotional intelligence, and more. These aspects are significant for both those developing software and those using it. In my view, this forms the foundation for any research involving the human dimension of software engineering, such as behavioral SE or human-computer interaction.

Restrepo-Tamayo and Gasca-Hurtado conducted thorough literature research to identify the human elements with the greatest impact on Software Engineering and the most promising potential for analysis [2]. They identified 15 independent variables and 11 dependent ones that have been studied in cause-and-effect relationships, with personality being one of the most extensively researched. However, as mentioned repeatedly, a challenge lies in the tools and metrics used to objectively evaluate these factors. Methods vary from questionnaires to observations or self-assessments, and common metrics include descriptive statistics, correlation, and regression, among others. While their review paper offers valuable insights, it also underscores what, in my opinion, is a hindrance to research in this area: the lack of a systematic approach to analyzing and obtaining generalizable results. It's widely understood that personality, for instance, influences human interactions, thus impacting the work environment and productivity – a fact often experienced by anyone working in a team. While research that confirms this is essential, there's still a need for quantification, systematic personality matching, or structured solutions that can genuinely enhance efficiency. In my perspective, there's great room for improvement and numerous research paths within this field.

5 Future trends

When considering the future trends in Software Engineering, there are various aspects to discuss, but one topic currently stands out. As technology continues to advance, work environments are constantly evolving and adjusting to the changing landscape. This transformation affects all professions, yet it's intriguing to contemplate how individuals engaged in AI are essentially creating tools that could significantly influence their own roles. A concern revolves around how these tools will shape job prospects and whether the roles of software engineers might become less essential in the near future. In this regard, the initial tools that spring to mind are ones like ChatGPT or GitHub Copilot. These tools have the ability to support engineers with coding, code refinement, efficiency enhancement, systematic test creation, documentation, and a wide array of tasks. While some might find this prospect "scary", I personally perceive it as a chance. Looking back at history, there have been various impactful technological advancements that have reshaped job markets. Nonetheless, what remains constant is that people's unique strengths always hold value. I believe that we can be clever in how we make use of these tools to our advantage. Asking the right questions of ChatGPT, for instance, can help generate higher-quality work with greater efficiency,

freeing us from monotonous and repetitive tasks. However, if we don't use these tools thoughtfully, we might end up wasting time or creating code that doesn't function as intended.

On a different note, I can't stress enough the importance of mastering the art of producing high-quality code. This is an issue that I personally struggle with and have mentioned multiple times throughout this essay. Like many researchers in applied AI, I lack a formal background in computer science. My coding skills are self-taught, and while the code I create might function, it often ends up messy, lacks scalability, and could potentially introduce bugs. The tools I develop serve the purpose of research, but they often fall short of meeting industry standards. This can become a roadblock when attempting to transition projects for practical use in the market. Furthermore, there's a noticeable gap between the perspectives of software engineers and the realities of AI. To bridge this gap, traditional software engineers and AI specialists need to collaborate closely. This prompts me to believe that the concept of specializing in AI engineering is fast becoming a reality. We're already seeing various job roles adopting this title. Currently, it's either a small specialization within the broader computer science field or a skillset acquired through real-world experience in the AI industry. Given the prominence of AI and the emergence of numerous specialized areas like generative AI, AI for computer vision, Large Language Models, and more, the natural progression towards AI engineering specialization appears imminent.

References

[1] Introduction to human factors. URL `https://www.hse.gov.uk/humanfactors/introduction.htm`.

[2] L. M. Restrepo-Tamayo and G. P. Gasca-Hurtado. Human aspects in software development: A systematic mapping study. In *International Conference on Collaboration Technologies and Social Computing*, pages 1–22. Springer, 2022.

[3] M. Sánchez-Gordón, R. Colomo-Palacios, M. A. Akbar, and M. K. Holone. A perspective on the role of human behaviors in software development: Voice and silence. *arXiv preprint arXiv:2304.12903*, 2023.