WASP Software Engineering Assignment

Noel Arteche
Lund University
noel.arteche@cs.lth.se

July 17, 2024

1 Introduction

My research lies on the mathematical side of WASP, usually called AI/MATH inside WASP's internal organization. In particular, I do research in theoretical computer science and specialize in computational complexity theory, the branch of the theory of computation studying the fundamental limits of computational intractability. In my research group, some members heavily use software engineering techniques and methodologies for software development, since many of the ideas we work with are directly related to SAT solving and industrially-competitive combinatorial optimization solvers.

Sadly, however, my own research lies on the purely theoretical side, merely understanding the theoretical limitations of such algorithms and their more general implications for structural complexity. As such, I am not directly involved in software development, and my particular scientific contributions are precisely in *lower bounds*, which are impossibility results. For example, recently we proved that under standard lattice-based cryptographic assumptions, quantum computers *cannot* be used to efficiently automate mathematics [ACG24]. So, in fact, no software engineering team will ever work on developing such an application.

Despite the lack of a direct connection of my research project to either AI or SE, the WASP Software Engineering course was a nice experience to delve into topics away from my main area of expertise and helped to refresh and expand on many topics that I covered long ago during my undergraduate degree in Computer Science.

The rest of the assignment is structured as follows. In Section 2 I compile four ideas from the lectures, two from Robert Feldt's lectures, and two from the SAAB lecture. While there is no software engineering in my research, some of the concepts and practices presented in the lectures are actually relevant to the way mathematical research is conducted and happens in social groups, and I highlight mostly concepts related to the *behavioral side* of software engineering. In Sections 3 and 4 I summarize two papers from the CAIN conferences that seemed interesting and intriguing, although this summary has no pretense of being applied to my own research[1]. In each section,

[1] In particular, I am unable to really answer questions (c) and (d) as per the assignment statement on how to related these papers to my research work.

I explain why the particular paper caught my eye, and why I found it interesting about my own mathematical research, even if at a more conceptual level.

2 Ideas from the lectures

I highlighted below four concepts from the lectures that I found useful and inspiring, even for my own mathematical research.

From Robert's lectures

1. **DRY: Don't Repeat Yourself**. Just like when writing code and developing large-scale software projects, in mathematical research one should not repeat themselves. You should always be aware of what others have done and try to build on top of it. Contemporary mathematics is a highly modular building; once a theorem is proven, we build on top of it rather than trying to reprove it. The only exception to this seems to be when one reproves something *in a completely different way* that leads to conceptual breakthroughs (in theoretical computer science, this could be, for example, Dinur's reproving of the PCP theorem [Din07] that lead to her Gödel Prize in 2019).

2. **People Are Not Rational**. I found the takes from Behavioural Software Engineering by far the most intriguing in the course. In particular, the fact that "people are not rational" applies as well to mathematical research. Just like in software developmenent, in mathematics, even if the final product seems to be a finely crafted prooduct of cold facts and logic, like a piece of code, its creation is done by humans who are very much irrational at times. By far the most interesting insight in my first year of PhD has been to see how other human mathematicians interact and work *in practice*, and this is a good motto to to keep in mind.

From the SAAB lecture

1. **The company organigram**. The company organigram and its "realistic" funny counterpart on page 6 of the slides is quite insightful to understand the human side of a company, and the take of behavioral software engineering is again applicable also to mathematical research. The social aspect in how mathematical cultures and communities are organized and operate has been studied by the sociology of mathematics and the philosophy of mathematical practice (e.g., [Man08; Wag17; Sri24]), and I suspect this goes very much in line with what behavioral software engineering has to say about software development practices.

2. **Q: How do you use SE at SAAB? A: We need it, and we don't know of any better alternative.** This question and answer on the need for SE at SAAB on the last slide of the talk was pragmatic but true. I liked that it seems to reflect how, in principle, they are not opposed to *some other form of solution* to address the problems they encounter when developing software – but, while open minded

about future options, this is in fact the best we have, and thus one should master it the best they can. Again, highly applicable to other domains as well.

3 PAPER I: What Is an AI Engineer? [MHS22]

Marcel Meesters, Petra Heck, and Alexander Serebrenik. 2022. **What Is an AI Engineer? An Empirical Analysis of Job Ads in The Netherlands**. In *1st Conference on AI Engineering - Software Engineering for AI (CAIN'22)*, May 16–24, 2021, Pittsburgh, PA, USA

Through the lectures we discussed that contemporary software development combines both profiles coming form data science and AI as well as people from software engineering per se. This comes together under different names: AI engineering, MLOps, SE4AI, and more. In this article, Meesters, Heck, and Serebrenik try to understand what is an AI engineer and what is expected of them by carrying out a systematic study of job advertisements in The Netherlands between 2018 and 2021. The goal of the paper is to identify the job tasks expected by the respective employers, as well as the required tools and soft skills, in an attempt to see where this expertise and training is coming form (software engineering, data science, or both).

The study proceeds similar to a literature review, by filtering and listing job adverts in The Netherlands during April 2018 and April 2021. They arrive at the definition of AI engineering as "a combination of machine learning and software engineering with the goal to build production ready machine learning systems" (p. 127, §2.1). They observe that the job tasks expected from these offers can be distributed in five main categories: business understanding, data engineering, modeling, software engineering and operations engineering. For these tasks, companies look for three kinds of profiles, (i) data science engineers focused on the development of AI models, (ii) software engineers for development and (iii) a so-called "generalist" AI engineer focused on both modeling and on software. Interestingly, around 40% of the offers are looking for this kind of generalist engineers. While tools often include fluency in Python, C++ or Java, the authors observe the job offers often stay silent about more specific machine learning tools. When it comes to soft skills, indications are even more vague: "ability to learn", "communication skills" and "team work mindset" are common terms found in these ads, which are not very different from classical software engineering soft skills.

The study is important for the engineering of AI systems in that it tries to identify what is needed by the industry in order to make well-informed decisions at an educational level, so that the now-emerging AI degrees at universities follow the industry's trends and train students in the skills needed. At a more fundamental level even, the study of AI engineering should be well-founded and have a practice-based perspective on what is is that it is doing.

I find the article interesting in that it tries to pinpoint what it is exactly that this new concept of AI engineer is by looking at the practice of its very own job market. The paper has clear applications (though perhaps debatable) in lying the foundations of the academic curricula of new AI engineering degrees. (In this regard, it is interesting to note – although they do not make this explicit – that the authors of the paper come from

the Eindhoven University of Technology, precisely where Edsger W. Dijkstra, one of the minds behind the classical computer science curricula around the globe, worked between 1962 and 1984.)

I would like to remark, however, that many of the paper's findings can be read as simply translating the current confusion of the industry to the academic setting. While it is fair to assume that companies do know what they need, these ads seem to suggest otherwise sometimes. For example, over 40% of adverts ask for this generalist type of AI engineer; while this can be understood as the emergence of this new profile by virtue of merging previously existing job categories, the vagueness in some of the offers also suggests that this generalist may in fact be working as some kind of a jack of all trades due to the very indefiniteness of the job positions. Similarly, many job offers do not seem to make a difference between applicants coming from a university background from those coming from am "applied" higher education background, suggesting that perhaps companies themselves are not completely sure of their needs.

Personally, I can relate to some of this in that theoretical computer science, as the intersection of many different fields, some more mathematical, some less so, is itself an emerging category and one can learn a lot about what it means to be a theoretical computer scientist by observing practices and looking at job posting in the academic and non-academic job markets.

4 PAPER II: A Meta-Summary of Challenges in Building Products with ML Components [NZL+23]

N. Nahar, H. Zhang, G. Lewis, S. Zhou and C. Kästner. 2023. **A Meta-Summary of Challenges in Building Products with ML Components – Collecting Experiences from 4758+ Practitioners**. In *IEEE/ACM 2nd International Conference on AI Engineering – Software Engineering for AI (CAIN'23)*, Melbourne, Australia

In line with the previous paper, this work is focused on understanding challenges that arise in the practice of AI engineering (referred to as "Software Engineering for AI (SE4AI)") by conducting a meta-summary of existing literature on the challenges in building software products that incorporate machine learning components. In particular, the study looks at 50 different papers that either interviewed or surveyed a total of over 4758 participants, and summarize the kinds of challenges that emerged in this setting.

Similar to the previous paper, the idea here is that by looking at what the existing practice-based literature has shown, one can extract useful patterns and insights. As the authors point out, "we have reached a point where the practices have settled and research on challenges approaches saturation – we think now is a good time to step back and survey the collective findings of the research community" (p. 171, §1). As argued also by Meesters, Heck, and Serebrenik in the previous paper, the goal here is that this meta-summary report can be a useful resource in identifying the key points in which both industry and academia need to focus to respectively have more efficient development cycles and train the future practitioners better. In this sense, this kind of

work is essential for the development of a good understanding of this emerging field of AI engineering.

The 50 papers were selected by standard literature review filtering techniques, in the span of twelve years, 2010-2022, aiming to include literature around the seminal paper of Sculley *et al.* [SHG+15] on the hidden technical debt of of ML software infrastructure. Interestingly, despite the interdisciplinary nature of AI engineering, most papers came from software engineering venues, with 30 out of the 50 papers published at such kind of conferences (five of which at CAIN).

The meta-summary concludes by identifying five main areas of struggles within AI engineering: (i) requirements engineering, where a lack of technical expertise leads to unrealistic expectations from customers and vagueness of specifications prevents from correctly measuring performance; (ii) architecture, design and implementation, where bringing together the model-centric structure of an AI system is often at odds with the traditional architecture of other software applications; (iii) model development, including all the issues arising from the more data-scientific activities; (iv) data engineering, given the difficulty in obtaining and maintaining reliable sources of data for production purposes; and (v) quality assurance, where testing and ensuring reliability of AI systems can be problematic precisely due to the aforementioned vagueness in the specifications.

References

[ACG24] N. Arteche, G. Carenini, and M. Gray, "Quantum Automating $\mathbf{TC}^0$-Frege Is LWE-Hard," in *39th Computational Complexity Conference (CCC 2024)*, 2024, 15:1–15:25. DOI: 10.4230/LIPIcs.CCC.2024.15.

[Din07] I. Dinur, "The PCP theorem by gap amplification," *Journal of the ACM (JACM)*, vol. 54, no. 3, 12–es, 2007.

[Man08] P. Mancosu, *The philosophy of mathematical practice*. OUP Oxford, 2008.

[MHS22] M. Meesters, P. Heck, and A. Serebrenik, "What is an AI engineer? an empirical analysis of job ads in The Netherlands," in *Proceedings of the 1st International Conference on AI Engineering: Software Engineering for AI*, 2022, pp. 136–144.

[NZL+23] N. Nahar, H. Zhang, G. Lewis, S. Zhou, and C. Kästner, "A meta-summary of challenges in building products with ML components – collecting experiences from 4758+ practitioners," in *2023 IEEE/ACM 2nd International Conference on AI Engineering – Software Engineering for AI (CAIN)*, 2023, pp. 171–183. DOI: 10.1109/CAIN58948.2023.00034.

[SHG+15] D. Sculley *et al.*, "Hidden technical debt in machine learning systems," *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[Sri24] B. Sriraman, *Handbook of the history and philosophy of mathematical practice*. Springer Nature, 2024.

[Wag17] R. Wagner, *Making and breaking mathematical sense: Histories and philosophies of mathematical practice*. Princeton University Press, 2017.