WASP Software Engineering Assignment

David Björkstrand
dbjorks@kth.se

August 29, 2024

1 Introduction

I'm doing a Industrial PhD project for the company Tracab. Tracab is a sports technology company which with multi-view camera systems installed in sports arenas, tracks the ball and players in professional soccer games. Historically, Tracab has mainly done center-of-mass tracking of the players, but recently the capabilities to do full human 3D pose tracking has been developed and deployed. My project resolved around human 3D pose and motion modeling.

Today, human 3D pose and motion modeling mostly involves deep learning. The goal of development when it comes to deep learning is finding a good algorithm for your task, which naturally involves implementing such algorithms in a programming language. I'd say for the algorithm to perform well it needs to be correctly implemented which already necessitates good software engineering processes. Furthermore, to be able to find a "good" algorithm, and to make sure we have a "good" algorithm in the end we need to be able to evaluate the algorithms. Evaluation involves both code and data and it's absolutely critical that the code is bug free and the data is correct for me to be able to draw correct conclusions.

From the last paragraph we can clearly conclude that good Software Engineering for Machine Learning is important for my research. However, as I'm doing an Industrial PhD, there is a wish for my research to be integrated into products. This expands the number of stakeholders of my algorithms from other researchers to industrial colleagues, clients and more. This further necessities that my algorithms are correct and fulfills requirements.

2 Robert Feldts lecture discussion

In this section I will discuss requirement engineering and quality assurance and how it relates to both my research and my industrial work.

2.1 Requirements engineering

Requirements engineering (RE) is the process of defining, documenting and maintaining requirements [2]. While I don't think I do any RE in the strictest sense I do have requirements on my algorithms originating both from the research community and industry.

The field of human pose is very benchmark focused, as is the wider field of deep learning. In my experience, it seems to be very important to be able to either perform better than prior works on these benchmarks, or perform similarly will being better on another important non-functional requirement (NFR) such as being more computationally efficient. So in a sense the requirements originating from the research community is documented and maintained in prior works. Martinez-Fernandez et al. [4] argues that our understanding of NFRs for Machine Learning (ML) are fragmented and incomplete. I would say that our understanding of functional requirements (FR) are also incomplete, both in the research community and industry. So for me when writing papers, it is important to be mindful of metrics commonly used and to potentially suggest improvements to these.

From the industry, some requirements are very well understood and well defined while some are not. The well understood and well defined requirements originate from, for example, third-party certifications or from my colleagues decade long experience in the our industry. From third-party certification we got requirements on accuracy. From industry experience, we got requirements such as being able to run the algorithms in real-time, which is very appreciated and important for many clients and applications. A less well-defined but nevertheless important requirement for us and the other AI-based software systems [4] is robustness. Indeed my research so far has been concerned with increasing the robustness of our human 3D pose estimation system using human pose and motion priors. However, robustness is not trivial to measure and how to measure robustness is a important part of my research in it self.

2.2 Quality assurance

In this section I will discuss the verification part of quality assurance. In other words what tools and techniques do I use to check whether I am building the algorithm and product right, and which tools and techniques would help if I started using them. Again it differs a bit between my research and industry work. But one thing that they have in common is that I do a lot of manual testing and I think tools like unit testing could help.

I think the coding I do in my research can be roughly divided into three categories. Algorithm coding, data coding (parsing and so on) and evaluation coding. I'm a firm believer that your evaluation code is a great testing tool both for data code and algorithm code. During refactoring (i.e. changes that are not supposed to change functionality) you simply re-train a model and make sure the results are the same. However, there is a couple of problems with this approach. First, it might be inefficient since a re-training can take considerable resources to do, in my case it is inefficient but I think it is fine. Another problem is that the evaluation code has to be correct. In my case I do a lot of manual testing to make sure it is, but unit testing I think could improve this process. The last problem is if the results are different, indicating that something is wrong, there is no indication what could be the problem. Again, I'm using extensive manual testing in this case and I also think unit testing could help.

In my industry work we extensively make use of code reviews, integration testing and system testing. I think if I added unit testing into my code it could also help here.

3 Guest lecture discussion

3.1 Behavioral software engineering

The SAAB guest lectures had a presentation in part about Behavioral Software Engineering (BSE) and defined it as a sub-field of Software Engineering that focuses on understanding and integrating human and behavioral aspects into the software development process. It might be a stretch but I think as I'm doing the PhD, BSE as it relates to my research will be about my behavior. However, even though I'm doing the PhD, I'm far from alone in the process. I have weekly meetings both with my academic supervisor and industrial supervisors (along with impromptu communication). Both which I think are of great value for my behavior and outlook. I get feedback on my work, I get help with my work and I get the opportunity to discuss ideas. My point being that supervisor-student is a great way to organize a PhD education.

3.2 Machine Learning for Software engineering

The Volvo guest lecture touched on using Machine Learning for Software Engineering (SE4ML). Specifically, using LLMs as a tool to help with testing (mocking, "fuzzy" search). Although I don't think writing code is boring (I would probably be in the wrong field if that was the case), I think that writing code is the least fun part of programming (designing solutions are more enjoyable to me). Thus, I'm all for using tools such as Large Language Models (LLM) to make writing code more efficient. However, I think it is important to point out that any organization will in the end be held accountable for the correctness of its code. So at a first glance I think using LLMs today for testing sounds risky, as stakeholders do not care that the LLMs think the code is correct. That said, I think there could be a trade-off. Humans are by no means infallible and as such, if a organization think that the use of LLMs could decrease errors, then it is the correct thing to do.

4 Paper 1

The first paper I have investigated is Investigating the Impact of SOLID Design Principles on Machine Learning Code Understanding [1]. The focus of the study is to investigate the effects of the SOLID design principles [3] on machine learning code understandable. SOLID is one of several design principles suggested for making code not just more understandable but also flexible and maintainable.

The motivation for choosing SOLID over other design principles is due to common inclusion in popular software engineering and machine learning content as well as having been studied in a academic context investigating design principles in general. They motivate the investigation of SOLID principles applied to machine learning code specifically with not being aware of previous studies doing so. They also argue that machine learning practitioners face unique challenges due to its exploratory nature and the diverse potentially non software engineering backgrounds of the practitioners.

The study finds evidence for SOLID improving machine learning code understanding and leading to the perception of several other benefits related to machine learning code maintainability.

As I write machine learning code for my research it's easy to see how this paper is related to it. My research on robust human 3D pose estimation has already been (and further results will continue to be) integrated into a larger human 3D pose estimation system consisting of many different parts such as non machine learning code, like video handling, as well as other machine learning code. Principles like SOLID are already applied to this system and the benefits are multiple. Code understanding is important in many aspects of software engineering. It makes it easier for new members of the team to understand the code and be able to contribute. Code understanding in it self also makes it easier to re-use code in addition to the designs principles other effects on re-usability. Furthermore, even though our system is in a stable state there is no shortage of feature requests. Good code design in general helps with making it easier to extend or system.

From a research perspective, implementing good design principles can help making it easier to do ablations and also make it possible to re-use a majority of your code between research projects and papers. It also helps integrating my research code into the larger human 3D pose estimation system. With good separation of concerns a lot of the code can be re-used. Data input and output handling is often the only code that has to be specifically re-implemented for the larger system, do to data sources and destinations changing.

I already try to write code following good design principles to the best of my ability. However, I don't think the knowledge is enough and a lot of practice is required and it's something that I'm always try to improve.

5 Paper 2

An Exploratory Study of Dataset and Model Management in Open Source Machine Learning Applications [5], the second paper I have studied, investigates how machine learning applications manages datasets and models. This is important because machine learning applications do not just produce traditional software artifacts (often managed with version control systems such as Git) but also produces machine learning specific artifacts like the two aforementioned. As with traditional software artifacts, data and models also evolve during the application development process, calling for their meticulous management.

They find that a lot of machine learning applications store their data and models in file systems. Locally or remotely if the files are large. This can cause issues for availability and eventually reproducibility. They also find that data and models often lack version information. If they are versioned with a version control system they are rarely updated. This can cause issues for reproducibility and traceability.

Again, it's easy to see how this relates to my research so I will go straight into how I'm thinking about these issues and how my method can be improved. For me a dataset can be divided roughly into to parts; raw data and dataset generation code. The raw data is stored in a central place, available for all, such as a database. The dataset generation code is stored in a repository (thus being versioned). The dataset generation code is parameterized with a configuration file. Given the configuration the dataset generation code takes the raw data (or a subset of it) and does some dataset wide pre-processing. The dataset is stored as the data itself, the configuration for the dataset generation code and they Git hash of the current commit. I think this can be seen as dataset versioning with extra steps. Given the configuration and git hash the dataset can be recreated if the raw data does not change. This of course is a big assumption that might not hold, and the implementation of a actual dataset versioning system would improve this process.

Now for the model, which is a artifact consisting of weights and model code. This artifact is produced using data and code. With every model I produce I save the model, the git hash of the training code, the dataset generation configuration and the dataset generation git hash. Thus we can see this as model versioning with extra steps. Also in this case there is a couple of issues. Firstly, it inherits any issues from the dataset versioning process. However, assuming I have a correctly versioned dataset it seems like I can consistently re-produce the model with what I save. However, there are sources of randomness when it comes to model training. The first being random number generators. I can mitigate this problem by manually setting and committing seeds, which I do. Hardware is a second source of randomness, if everything is the same except the GPU the model is trained on, the end model might be different. In conclusion, this process could be improved by versioning the model.

References

[1] Raphael Cabral et al. “Investigating the Impact of SOLID Design Principles on Machine Learning Code Understanding”. In: *Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering-Software Engineering for AI.* 2024, pp. 7–17.

[2] Gerald Kotonya and Ian Sommerville. *Requirements engineering: processes and techniques.* Wiley Publishing, 1998.

[3] Robert C Martin. “Design principles and design patterns”. In: *Object Mentor* 1.34 (2000), p. 597.

[4] Silverio Martínez-Fernández et al. “Software engineering for AI-based systems: a survey”. In: *ACM Transactions on Software Engineering and Methodology (TOSEM)* 31.2 (2022), pp. 1–59.

[5] Tajkia Rahman Toma and Cor-Paul Bezemer. “An Exploratory Study of Dataset and Model Management in Open Source Machine Learning Applications”. In: *Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering-Software Engineering for AI.* 2024, pp. 64–74.