Assignment, WASP Software Engineering Course Module

Emma Granqvist

August 28, 2025

1 Introduction

In recent years, artificial intelligence (AI) and machine learning (ML) techniques have transformed the drug discovery process by enabling automation of different steps of the drug discovery pipeline while providing valuable insights. This development is a result of the recent development of AI and ML methods and the from the growing amount of chemical data. Thanks to these technological advancements, the often long timelines and costly processes can be made more efficient by enabling automation and higher precision at different stages in the drug discovery process. Organic synthesis plays a central role in drug discovery and aims to efficiently manufacture organic compounds through series of chemical reactions. This is a complex task which can often be a bottle neck in the drug development process because of the huge search space of building block molecules and the multiple ways to combine these.

The core of my research is to investigate and apply Artificial Intelligence (AI) and Machine Learning (ML) methods to drug discovery and more specifically with a focus on combining synthesizable and *de novo* drug design, which aims to find potential drug candidates which are also synthesizable. A substantial part of the chemical space is in fact not synthesizable given the known chemical reactions available and starting materials. Which is why the synthesizable constrains are an essential although somewhat overlooked part of molecular discovery. In my research, I am developing a reinforcement learning framework for synthesizability constrained molecular design. This is approached by generating synthesis routes using known chemical reactions and starting materials in a bottom-up fashion and conditioned on desired chemical properties of the final product.

2 Lecture principles

2.1 Quality Assurance and Testing

Key concepts discussed in the course and in software engineering (SWE) as a field is quality assurance and testing and there are many reasons why. When building software systems there are many things that could possibly go wrong and that needs to be caught. Hidden bugs in the code could have tremendous effects and could cause the software to be unsafe, doing the wrong thing, being less usable, of lower quality or give unreliable or faulty results. The risk of hidden bugs increases with number of collaborators and expansion of the code base. Therefore testing and quality assurance is a key component of SE and crucial in order to deliver a safe software. When it comes to developing software and codebases for research purposes it seems however like testing and quality assurance is not a priority. During the lectures for the SWE module we had the opportunity to discuss the lecture topics with peers and the majority of fellow PhD students didn't include testing to an extent that they would like, myself included. Even though far from all code developed for research will be productionized, the validity and quality of the code is of course still very important. If the results in a publishing paper comes from partially untested code - how could one trust the results? Although the discussions revolved around to what extent the codes were tested it seemed like people considered this important and did smaller tests although few reported that they performed extensive test. Specifically for my project this could mean to write more unit tests for the individual functions and regularly run tests to see that new versions of the code pass the tests. I believe that spending more time on this could even reduce the total time spent on the project as possible bugs would be found faster.

2.2 Behavioural Software Engineering

Behavioural SE is the study of how human aspects such as cognitive, behavioural and social aspects relates to SE and how the development of the software is dependent on the human factor eventhough the engineer is not aware of this. This analyse can be done in three levels; organization, group and individual. On the individual level one thing that impacts the software engineer (SWE) is confirmation bias, this refers to the tendency to pay more attention to information that confirms our prior beliefs in contrast to information that contradicts them, together with availability bias, the tendency to being influenced by information that is recent, easy to recall or widely spread, impacts the thinking process and might prevent the engineer from making rational decisions in development. More specifically, this could prevent the developer from finding the correct source of program failure by insufficient testing by for example running more tests that confirms that the program works than actually finds the problem. On a group level things such as group dynamics and norm has a big impact on how a team works together and ultimately what decisions are made. This could for example take the shape of group thinking which is a desire of the group members to keep together and reaching agreements than of making the right decision. The general effectiveness of a group can be measured by collective intelligence. The collective intelligence of a group is only mildly correlated with the individual intelligence of the individuals in the group, thus having only very smart people in one group does not necessarily result in a effective group. Instead it seems like the team members ability to recognize human emotions are of higher importance and inclusive conversations, where all people would contribute to the conversation compared to conversations dominated by few persons where indicators of higher collective intelligence. The human aspect is also highly relevant to scientific research. The lecture reminded me to be aware of my personal presumptions and biases. Working as a PhD student most of my work is individual however I am also part of two teams. Being part of groups of course has an influence on the research topic, how I perform my research and what software I tend to use.

3 Guest-Lecture Principles

In one of the guest lectures Julian Frattini talked about requirements engineering. This refers to a process in software engineering which focus on collecting, documenting and managing the requirements, needs and expectations of stakeholders to ensure a satisfactory final product. This process can also be applicable to my research project although the connection might be less clear compared to traditional software engineering. I will discuss two of the key processes namely; *Stakeholder elicitation* and *goal modelling.*

3.1 Stakeholder elicitation

Stakeholder elicitation involves identifying all stakeholders and gather all their needs and requirements for the final product. In my case, the involved stakeholders involves my academic and industrial supervisors and groups, myself, examiner and also in a way the wider scientific community. My supervisors are key stakeholders as my research should be in alignment with their research and the overall goals of the groups. In my case, both my industrial and academic group focuses on AI for drug discovery where the focus of my industrial group includes synthesizability and one of the focus area of my my academic group includes molecular design. As for myself being a stakeholder, it is important that the research is align with my professional interests and that the research is contributing to my professional and personal development as the research I am doing is also part of my PhD journey. In a way also the wider scientific community as it is of my personal interest to conduct research which would be considered important for others in the field.

3.2 Goal Modelling

Another crucial part of requirements engineering is goal modelling which aims to define and map out the objectives which the stakeholders wants to achieve through the system. The goals can then be used to define and refine the requirements of the system and finally to validate that the system full fills the intended purpose. The goals can also be divided into either *Usage goal*, *System* *goal* or *Business Goal*. Where the usage goal is directly relevant for the end users, the system goals are directed at the system properties and capabilities while the business goals are strategic and organizations specific. The usage goals for my project would be defined in accordance with the stakeholder with the end user in mind and is also influenced by the norm in the scientific community. For example, the overarching goal of my project would be to develop a method for molecular design where the produced molecules are synthesizable while sub-goals could involve obtaining a certain result on public dataset or publish the results in a specific venue. As research often is more unpredictable than classical software engineering, the goals might be less specific and can also come to change if the direction of the research changes.

4 Data Scientists versus Software Engineers

In *Machine Learning in Production* [Kas25] data science and software engineering are described as two mostly complementary fields with two distinct focus. Data science is described as mostly model centric where the overall objective often is to develop AI/ML models with the highest possible accuracy. The data scientist should be skilled in statistics and data science in order to understand the data at hand, identify and train a suitable algorithm and model architecture. Contrary, SWE focus on building and testing software *systems* which may or may not include AI in some way. SWE instead needs to be able to design and build scalable systems and understand user requirements. SWE also needs to consider constraints such as usability, scalability security development time and cost a greater extent. Although, this description of the two fields is an overgeneralization and simplified of reality, I would tend to agree with the general distinction between the two fields. However, I also believe from my own experience that in reality often the same person is expected to take on both roles and that the distinction is more fluid than it comes across in the text. Depending on the extent of the project or the size and budget of the group, the data scientist might be expected to productionize the project or the SWE might be expected to also do the job of a data scientist, thus making this distinction less clear in practice. This is also mentioned in chapter 1 under the title AI engineer or ML engineer, referring to positions that has a clearer focus on engineering in machine-learning projects.

AI and ML have in the recent years become more and more available and Software engineering will have to adopt to account for these recent advancements and an increased understanding of AI systems will be required. In addition many productivity AI based tools have become available that can be useful to a software engineering to automatically generate or review code in an efficient manner. Using these sorts of tools will be crucial as these can increase the work efficiency significantly. I believe that in some ways the roles of both the software engineer and the data scientist will change to adopt to the developments and the growing need for both competencies. However, I believe that there will most likely still be a need for both roles in different positions although, someone skilled in both areas will be a valuable resource to any team.

5 Paper analysis

5.1 Paper 1: Investigating Issues that Lead to Code Technical Debt in Machine Learning Systems

The paper *Investigating Issues that Lead to Code Technical Debt in Machine Learning Systems* by Ximenes et al [Xim+25] publish their work on technical depths in Machine learning projects. The concept of technical debt is an analogy between financial borrowing and software engineering and describes how fast deliveries can lead to a debt which similar to the financial debt will accumulate over time and demand eventual repayment. The intersection between ML and technical debt has lately attracted increased interest although the understanding and management of technical debt in ML remains a relatively unexplored area which is a gap that Ximenes et al. are trying to fill with their research. In their article, they assessed issues which could lead to technical debt i ML systems by analysing problems related to typical ML code development including the occurrence and relevance of the identified issues. Among the 34 identified issues, the list where refined down to 30 in accordance with a group of nine experienced ML practitioners. Out of these, 24 were considered highly relevant by the authors. Areas that were found to be particularly critical included data pre-processing where 14 of the highly relevant issues were found. Other areas that were particularly sensitive included data collection, model creation, training and evaluation. It is important to discuss the technical debt in ML systems and as an engineer of AI systems to be aware of the risks at each phase of the development. Increasing the awareness of the risk of technical debt by incomplete/insufficient SE practices could in effect reduce the risk of making the mistakes.

The findings can be applied to my own research as well as I am working with developing a AI framework. Making mistakes early in the process could slow down the development time and/or generate untrustworthy results which is highly undesired and finally could increase the total time spent on the project as the time required to manage the code will increase. The publicly available chemical datasets are of varying quality and if combining data from several sources there can be some inconsistencies in format or in the way the data was collected. When considering chemical reaction data negative data is generally not available and reaction conditions might also be insufficient. Taking the compiled list into consideration could highlight specific parts of the development where additional care should be taken to avoid this.

Imagine a project where a group of AI engineers are trying to develop a self-driving car where this particular group is trying to detect objects on the road using video. The training data might be collected from different datasets and the video material might have inconsistent formats and inconsistent labelling which could lead to noisy data. The inconsistency in data handling in combination with the lack of an agreed-upon process for pre-processing can affect the training process negatively and lead to unreliable and unreproducible results. It is crucial to build a robust system from the beginning and the findings by Ximenes et all highlight the importance to pay additional attention to the earlier phases such as the data collection and pre-processing stages in order to mitigate the technical debt. For the self-driving car project having a dataset which covers the possible scenarios that could occur is crucial for the system to be safe in a real-world setting.

Independently of the details of the project, the key to mitigating technical debt when developing AI projects is to first identify the potential sources and to continuously address it during development by testing and refactoring of the code. Code review can be one strategy for mitigating debt as well as fostering a workflow which facilitates debt management such as clean code practices, agile development, version control, code reviews and proper testing practices. Technical debt has been discussed in SWE but is not discussed to the same extent in ML engineering. Specifically in research project it is my experience that the risk and mitigation strategies are overlooked which is why this study is important.

5.2 Paper 2: Addressing Quality Challenges in Deep Learning: The Role of MLOps and Domain Knowledge

The paper *Addressing Quality Challenges in Deep Learning: The Role of MLOps and Domain Knowledge* [Del+25] by del Rey et al explores how to integrate MLOps practices and domain knowledge when developing deep learning systems and how this impacts system quality.

MLOps is similar to DevOps but is adapted for the specific needs of ML systems and focus on tasks such as experiment tracking, model versioning and automatic performance monitoring. Integration of MLOps processes enables transparent and reproducible experiments and facilitates mapping design decisions to model results. In this paper they focus on MLOps in terms of improving and monitoring with respect to correctness, time efficiency and resource utilization in the context of deep learning (DL) for image recognition by employing open-source software to track the experiments and the effect on the metrics and energy consumption. Tracking the experiments offered valuable insights to decision making process. To further improve traceability, they automated the data collection and processing process. Further, the authors show how including domain knowledge can achieve significant better results in term of accuracy, latency and energy consumptions compare to the domain-free approach. By incorporating domain knowledge into the system's logic the model design can be simplified and this was demonstrated on an example where computer vision was used to recognize the state of a chess board. The domain-aware algorithm included domain-knowledge such as the previous state and the legal moves and the overall goal with including this information was to reduce the complexity of the search space compared to the domain-free model.

When developing any DL project, it is important to assess quality attributes of the system such as correctness, time efficiency, and resource utilization. The authors suggest that the experience and insights presented in this paper can be beneficial for practitioners when developing ML systems to manage their time and energy consumption and at the same time improve the transparency and reproducibility of their project. Further, they argue that this is especially important in DL where there often is a tendency to focus on a single qualitative metric, such as accuracy, while overlooking the overall quality of the system. Software developed for research such as my own doesn't have the same functional requirements as it is primarily designed for experiments rather deployment and thus naturally have different constraints and requirements. For my own research, I believe that the strategies presented could be very helpful to develop more robust and scalable system. It is also my experience that correctness is very highly valued in research while energy consumption is somewhat overlooked. When developing software for deployment, the energy consumption is naturally very important for the product to be useful but also when working with limited computing resources, such as often is the case in research, attending to the energy consumption would be very important as to be able to make the most out of the available resources also for environmental reasons.

In addition, considering how one could incorporate domain knowledge into the model and that this could reduce the model complexity and energy consumption. In my current reinforcement learning framework for molecular design, domain knowledge is incorporated as the algorithm is aware of the available chemical reactions and considers only the legal reactions (actions) at each state. However, it is interesting to consider if additional domain context could be included to improve the model or its resource utilization. When using the framework for molecular generation, a high diversity for the generated molecules is desired however there is a possibility for the model to get stuck in a local minima and therefore generate similar compounds. Some context information could here be included to enforce diversity. In addition, including the experiment tracking and monitoring of energy consumption would be important steps to consider when planning for long-term maintenance and development of the software.

For project deployed in real-world scenarios, such as the self-driving car project (briefly described in Section 5.1), the techniques discussed in the paper by del Rey et al can be very valuable. When deploying systems there are specific case dependent requirements beyond accuracy such as the speed of the system and energy consumption in order for the system to be useful. These challenges can be aided by using the are to be expected such as functional requirement and a reasonable energy consumption and including the suggested processes can be used to faster guide the development towards a more reliable and quality system. In addition this fictional project should consider including domain knowledge in the detection algorithm which in this context could be previous time frames.

6 Research Ethics & Synthesis Reflection

To select papers for Section 5 I started by browsing the titles of published articles for 2025 and for the ones that I found interesting I also read the abstracts. I was looking for papers which I found useful for my own work and not necessarily the specific techniques but rather that could improve my workflow and that could be useful not only for my specific project but rather for any ML project as the ones that I chose focused more on the development process. I didn't encounter any papers that I found misleading and didn't find the selection process difficult and based the selection on finding papers that I found interesting and useful. To ensure originality, all text in this assignment have been written by me. I have only used LLMs on a few occasions to suggest alternative expressions or synonyms when I was not happy with the original one but I have avoided any copying of text.

References

[Del+25] Santiago Del Rey et al. "Addressing Quality Challenges in Deep Learning: The Role of MLOps and Domain Knowledge". In: *2025 IEEE/ACM 4th International Conference on AI Engineering–Software Engineering for AI (CAIN)*. IEEE. 2025, pp. 184–189.

[Kas25] Christian Kastner. *Machine learning in production: from models to products*. MIT Press, 2025.

[Xim+25] Rodrigo Ximenes et al. "Investigating Issues that Lead to Code Technical Debt in Machine Learning Systems". In: *2025 IEEE/ACM 4th International Conference on AI Engineering–Software Engineering for AI (CAIN)*. IEEE. 2025, pp. 173–183.