Software Engineering, Assignment 2: Concepts and Future Trends in Software Engineering

Jennifer Andersson

October 31, 2022

1 Concepts in Software Engineering

In this section, three concepts in software engineering are discussed: human-computer interaction, requirements engineering and behavioral software engineering. This will be related to my own research area, which concerns acceleration of scientific computing using machine learning and the very broad area of machine learning in general. I am currently working on a project that involves probabilistic modeling of stellar magnetic fields. The goal is to quantify the entire family of magnetic fields that can be fitted to a given set of spectral observations. Due to the purely academic nature of this research, I will mostly discuss the relevant topics in relation to machine learning in general.

1.1 Human-Computer Interaction

Human-computer interaction is a research field focused on the interaction between humans and computers. It concerns the design of the user interface allowing humans to interact with an IT-system, and has become increasingly important as interactive technology has started to permeate a larger part of everyday life, both in the private and professional setting. Human-computer interaction is an important part of the technological development in software engineering, as the success of an application relies on how easy it is to navigate and use for the correct purpose from the point of view of the intended user. Therefore, the further technology advances, the more important the field of human-computer interaction becomes in order for the intended users to benefit from these advances.

In an overview discussing advances in human-computer interaction, Mathew et al. (2011) identifies two main properties that successful technology needs to address: functionality and usability. In order to accomplish the desired balance between the two, one needs to take into account technological aspects in conjunction with psychological aspects related to how a human thinks and acts when navigating a new application or other IT-system. To address the challenge of incorporating user-friendly interfaces for increasingly advanced technology, I believe that there will be an increasing demand for user-interfaces based on intelligent technology. Mathew et al. (2011) touches on adaptive and intelligent human-computer interaction, exemplifying the latter by a system that can adapt to the user in terms of learning its behaviour and adapting to its needs. My research area, machine learning, obviously has an important role in this development. For example, incorporating intelligent speech recognition and speech generation systems into the user interface can enable humans to interact with computers similarly to how they are used to interacting with other humans. The same is true for intelligent text generation and text recognition technology. The interactive systems can learn to predict the needs and habits of the user, allowing for personalized user interfaces tailored to a specific user.

From a machine learning research perspective, another emerging challenge regards the increasing use of intelligent technology by people who lack sufficient technological knowledge. When intelligent systems are used to make decisions and aid in the everyday lives of the users, a large challenge is related to the lack in understanding what causes these systems to make the decisions they do. Nazar et al. (2021) reviews explainable AI in relation to human-computer interaction, which I believe also plays a large part in improving the human experience in the interaction with technology, by increasing trust through making the intelligent systems easier to understand and making the system decision process transparent. This will allow for smoother interaction, as understanding of the decisions made by technology improves the user experience.

1.2 Requirements Engineering

Requirements engineering is a branch of software engineering related to defining and analyzing the requirements of the software engineering system. This is an important part of the software engineering process, aiming to ensure that the developed system fulfils the original needs and goals. This includes understanding what the customers or users need from the final product or system, analyzing possible solutions and their requirements, effects and constraints, including feasibility in terms of for example technological and financial demands. The requirement engineering process also involves maintaining and updating the identified requirements as part of the software engineering process. An important part of the requirement engineering process is referred to as requirement elicitation. This stage of the process involves collecting, analyzing and merging requirements of different stakeholders. The requirements are then formally and technically specified, after which the requirement validation process begins. This is important in order to detect problems with the specification, which can be done by for example through legality analysis and technical prototypes.[1]

Based on my understanding of the goal and process of requirements engineering, its importance increases the larger the involved team grows, including both the software engineering team and the involved customers and stakeholders. The more advanced technology the system requires, and the greater effects the deployed system will have on society, the greater the challenges of requirements engineering becomes in terms of identifying and merging the requirements of different stakeholders, as well as obeying tightened and sometimes conflicting constraints. Advances in machine learning can make this process more complex, as the software engineering process becomes more complex and requires different specializations and roles, for example by adding requirements on the data collection process that becomes increasingly important compared to standard software engineering. However, machine learning may also help overcome these issues. For example, Zamania et al. (2021) explores how machine learning can be used to improve and automate the requirements engineering process, which is an interesting perspective on the future development. Perhaps some of the challenges to requirements engineering that specifically come with machine learning technology can be solved using that same technology, at least in terms of analysis and validation of technological requirements. To me, this is an interesting discussion and potential application of machine learning.

1.3 Behavioral Software Engineering

The field of behavioral software engineering takes the human aspects of software engineering into account. Lenberg et al. (2021) defines the research field of behavioral software engineering as the study of cognitive, behavioral and social aspects of software engineering performed by individuals, groups or organizations. In contrast to human-computer interaction, which has been previously discussed, behavioral software engineering focuses on the human aspects of the software engineering teams and the individual software engineers within those teams, as opposed to system users.

From my perspective, an interesting aspect is related to differences in the behavioral aspects of traditional software engineering teams and machine learning engineering teams, and challenges in terms of incorporating machine learning engineers and the machine learning engineering process into the software engineering process. As someone who has not been a part of the software engineering process on any larger scale, my initial instinct when learning about best practices in software engineering is often that parts of it does not seem applicable to the requirements of the machine learning process. In light of this experience, the study by Amershi et al. (2019) on software engineering teams at Microsoft developing AI-based applications was very interesting and enlightening. In particular, I believe that the difficulty of applying modularity to the working process in machine learning, as discussed in the paper, is an important aspect that relates to the behavioral software engineering field. If machine learning does not to the same extent as standard software engineering allow for different teams to work on different modules in isolation, this puts additional social demands on communication and collaboration between teams. This in turn relates to the challenge of identifying the modules. In my experience, the difficulty in identifying modules has led to teams decreasing in size and an "all hands on deck" approach. I do not know if this is a problem in larger teams, but balancing the difficulty in modularity and the efficiency in succeeding with it seems like an important discussion, and in that discussion behavioral software engineering is sure to be an important tool.

[1]The definition of requirement engineering described in this paragraph is based on information from https://www.javatpoint.com/software-engineering-requirement-engineering, accessed 2022-10-29.

2 Future Trends and Directions in Software Engineering

Currently, my PhD project is focused on machine learning within the natural sciences. Certainly, I believe machine learning is a powerful tool enabling us to extract and utilize information from the increasing amount of data we have access to today, but in the ongoing discussion of whether machine learning will replace traditional methods in for example the fields of physics and astronomy, I am of the opinion that expert field knowledge can be used to inform and improve purely data-driven approaches, and that traditional methods will therefore be incorporated into the machine learning process. I believe the same is true in the discussion of whether machine learning will ”eat” traditional software engineering. Hence, I believe that, with the impressive progress that is made within fields of machine learning and data science, it will indeed become a larger part of the software engineering practice, but I also believe that traditional software development will inform and improve the machine learning process. The paper by Amershi et al. (2019), analyzing the working process of software teams at Microsoft developing AI-applications, is an example of this. In the study, the authors find that software engineering teams at Microsoft have incorporated the machine learning workflow into the existing software engineering process, utilizing existing best practices. This provides insights, and the study also sheds light on obstacles related to merging the software engineering and machine learning process. This is something that I believe we will see a lot more of in the near future, with behavioral software engineering playing an even more important role.

Indeed, machine learning comes with specific challenges that expands various parts of traditional software engineering process and makes the division of labor within software engineering teams more complex, for example due to difficulty in isolating entangled components in the machine learning engineering process to be assigned to distinct working groups. Challenges also include larger focus on data collection, which I in turn believe will need to be entangled even more with for example social areas of society. One reason for this is the bias induced in machine learning models due to the inherent bias in the machine learning data sets. This is a serious problem in machine learning that I believe will take a lot of focus in the software engineering process for machine learning applications, not the least in terms of requirements engineering and feasibility studies. Biased models can have a significant impact on society when applications are used on a wide scale, and the machine learning software engineering process will need to adapt to a situation where carefully collecting high quality data is considered a high priority and a significant part of the software engineering process. I therefore believe that ethical discussions will become a larger part of the software engineering process.

Much like it is tempting to think that traditional physics will be replaced by purely data-driven approaches that can be applied easily to different physics problems through black-box structures, it may be tempting to believe that software development will be replaced by black-box implementations and code that to a large part consists of inherited glue code. Consequences of this is discussed for example in the paper on hidden technical depth in machine learning systems by Sculley et al. (2015). However, I believe that to gain the most out of machine learning models, we must incorporate domain knowledge, put great effort into collecting high quality data and performing domain-acknowledging prepossessing, adapting loss functions to domain knowledge combined with expertise in traditional computer science to speed up algorithms, and more. Neglecting to see this encourages build-up of hidden technical depth, a discussion that is very interesting in terms of the future development in software- and machine learning engineering.

Indeed, based on the above discussion, I believe that the standard software engineering process will inform the machine learning engineering process, and that machine learning in turn will be used in various phases of the traditional software engineering process, for instance in automated testing and requirement specification. This perspective is important. Machine learning is a powerful tool that can be incorporated to automate, improve and assist various parts of the software engineering process: from scheduling tasks and resource management to analyzing requirement specifications, automating code analysis and even automating code generation. With the significant progress in these areas, machine learning can become standard tools for aiding in producing, reviewing and testing code for both traditional software engineering and machine learning software engineering.

The software engineering part of this course has informed my thinking regarding how machine learning can be incorporated into the software engineering process, and how this can affect how software engineering teams are configured when focused on machine learning development. The relation between machine learning and software engineering is indeed an interesting topic with many different angles, pitfalls and potential, and this discussion will be valuable to my future collaborations in various teams.

3 Bibliography

Amershi, S., y, y, and z (2019). Software engineering for machine learning: A case study. *IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pages 291–300.

Lenberg, P., Feldt, R., and Wallgren, L. G. (2021). Behavioral software engineering: A definition and systematic literature review,. *Journal of Systems and Software*, 107:15–37.

Mathew, A. R., Hajj, A. A., and Abri, A. A. (2011). Human-computer interaction (hci): An overview. *IEEE International Conference on Computer Science and Automation Engineering*, pages 99–100.

Nazar, M., Alam, M. M., Yafi, E., and Su'ud, M. M. (2021). A systematic review of human–computer interaction and explainable artificial intelligence in healthcare with artificial intelligence techniques. *IEEE Access*, 9:153316–153348.

Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J.-F., and Dennison, D. (2015). Hidden technical debt in machine learning systems. *Advances in Neural Information Processing Systems 28 (NIPS 2015).*

Zamania, K., Zowghi, D., and Arora, C. (2021). Machine learning in requirements engineering: A mapping study. *IEEE 29th International Requirements Engineering Conference Workshops (REW)*, pages 116–125.