Assignment 2

The topic of this PhD student's research can be summarized as "data-driven control of thermal energy systems in buildings." Of particular interest is the coordination of the components in Building Energy Systems (BES) to maintain a comfortable indoor climate, at a low monetary cost while using as little energy as possible. But the components of a BES are highly interactive and achieving these goals using traditional control methods is prohibitively difficult. The controller must coordinate the operation of all components in the BES while also considering complications such as the system's dynamic response, and time delays in sensor measurements and actuators. In large buildings, where the number of measured and controlled variables increases, this coordination problem becomes extraordinarily complex, so the need for new control methods is apparent. The goal of this project is to investigate the use of data-driven control methods incorporating different forms of Machine Learning (ML) methods to handle the high levels of complexity.

Continuous deployment

Continuous Deployment (CD) involves the automated, and often frequent, deployment of software to end users. The role of CD in ML pipelines is significant, as ML-systems are meant to improve as new data becomes available, possibly creating the need for rapid deployment of new system versions. CD also creates a strong foundation on which to build new systems and can lead to a rapid and constant improvement of ML models. As one example, the idea of continuously deploying an ML service could aid in finding bugs that might not have been found without the frequent deployment phase. This rapid exposure of the system to actual end consumers could potentially speed up the discovery of bugs by providing immediate user feedback.

In the case of data-driven BES control, CD plays an even more significant role. The only truly robust way of testing a control system is by deploying it in the physical system. Although there are possibilities of testing control systems in silico, this strategy would not scale well in the case of BES control. Strictly speaking, it would require the availability of an accurate simulator for every building where the control system would be deployed, which is clearly not feasible in practice. Furthermore, the complexity of implementing such simulators within an ML pipeline would be immense. In contrast, using the physical systems as a base for testing through CD allows for the evaluation of control systems in a manner that is highly scalable. It also makes it possible to evaluate the performance of BES control systems in the context of a real-world environment in the presence of building occupants and environmental variations. Given the automated operations of CD, the testing of control systems through deployment could also be scaled to entire clusters of buildings. One can envision a scenario where new measurement data from the BES becomes available, a data-driven controller is updated and deployed in a cluster of buildings, and feedback regarding its performance is collected by monitoring key variables within the system.

At this point, it is important to note that the approach of CD of data-driven BES controllers is not completely devoid of risk. As it requires deployment in a physical space in direct connection with real people, control systems that are not fully operational may cause unexpected damage. Also, this damage could be potentially difficult to detect until a structural failure was triggered

by a malfunction in the BES. The importance of such damage detection and prevention in the use of CD cannot be stressed enough. The failure of BES controllers has the potential to lead to a cascading sequence of failure that could ultimately lead to an increased monetary cost for the building occupants. In conclusion, there is significant potential for CD to become an integral component of BES control systems. However, it is essential to be clear about the risk involved.

Human factors

The meaning of human factors in the context of software engineering for ML is two-fold. Firstly, it concerns how people interact with software engineering tools and methodologies in their ML workflows. Due to the complexity of software engineering and ML problems, it is essential to understand how people interact with a tool to ensure that it *does* what it's designed to do. Moreover, one needs to ensure that people *use* the tool for what it is designed to do, and not in some other way. Secondly, human factors in software engineering for ML concerns how people interact with each other in, e.g., software engineering projects. These include issues of interpersonal relationships, interpersonal interaction, work style, communication, project management, and project team dynamics. These factors are crucial to understanding how people work together and why the team as a whole is successful. It is entirely possible that developing the team regarding these aspects is just as fruitful for the productivity as developing the interaction with technical software engineering tools.

The above certainly also holds true for software within the field of BES control. However, one key point of the human factors in software engineering and ML in general was intentionally left out: the reluctance of humans to *embrace* new tools and methodologies. The area of BES control, which resides in the very conservative building industry, is probably a good example of this. In the author's experience, when it comes to ML and software engineering, engineers in BES control simply do not see the point of adopting new tools and methods if their current solution already does the trick. That said, there are cases where new methods lead to increased system complexity, and perhaps also a heavy workload for the project team. Nonetheless, even in those cases, the need to learn new tools and methods is still critical. Therefore, an important future research direction within BES control and engineering in general, is to analyse how to implement continuing education within software engineering methodologies.

Security and privacy

Data security is concerned with protecting the data from unauthorized access, or loss. It can be as simple as password protection, or more sophisticated likebackups, encryption or authentication. Data privacy, on the other hand, concerns the confidentiality of data, and the necessity to safeguard it from unauthorized disclosure and sales. Lately, with the introduction of legislative frameworks such as GDPR, the focus on data security and privacy issues is increasing. As a result of these new laws, companies need to consider the data security and privacy concerns more deeply in their decision-making.

Again, one can use the example of BES control, which relies on large scale data collection and storage for monitoring and billing purposes. With respect to data security, it is the author’s experience that building owners have had a high level of awareness for quite some time. For example, their so-called Building Management Systems (BMS), which are responsible for all data collection in BES, typically do not have a connection to the internet. Instead, they rely on local networks spanning only small clusters of a few buildings which are connected to a central

control station where the data is stored locally. Whether this is a good solution is up for debate, but, indeed, building owners have not traditionally had problems with data security.

In the building industry, the focus has shifted from only using the stored data for billing and fault investigations to also using it for energy analysis and control. Moreover, data from building occupants has become a highly valuable good as it includes information about our many of our everyday habits. As the commercial interest for building data increases, it is imperative that more research be made regarding the data privacy in BES.

Future trends

If the past has taught us something, it is that the future of software engineering is difficult to predict. With the rapid proliferation of cloud services and other types of distributed computing, it is difficult to imagine what changes it will bring in the future. It seems that the increased focus on data is already changing the way organizations operate, and we are likely to witness this more of this change as data is deeply integrated into companies' decision-making models. Moreover, as ML finds its way even into the most conservative of industries, the adoption of software engineering tools and workflows will likely continue to advance. If engineers in all disciplines use ML in some form, they will inevitably be exposed to the same kinds of problems that software engineers face. It is interesting to note that the future of software engineering is somehow entangled with the widespread adoption of ML, both in academia and in the industry.

But the question is whether the software engineering aspects of ML will be automated or "democratized" in the same way ML has. For example, it is known that producing a basic ML-model today requires hardly any intellectual capacity thanks to the many libraries and APIs that streamline this process. Is it unreasonable to suspect that the software engineering aspects of ML will be absorbed into this automated workflow as well? A quick glance at the recent history of ML engineering suggests that the answer is no: the most successful ML tools have in common that they hide their complexity behind several abstraction layers. Chances are that software engineering tools for ML will abide by the same Darwinan logic—the simple, yet useful tools become most competitive. In other words, the question is not whether software engineering will become part of ML, but whether there is a competitive advantage in offering software tools that incorporate seamlessly into the ML-pipeline.

In the case of BES control, one might reasonably question whether the current software engineering tool suite is adequate to cope with the complexities of ML pipelines. For instance, one should not expect a standard control system in a building to be able to manage the complexities of continuous deployment, automated testing, or versioning. In fact, implementing such software engineering and ML tools in today's building control infrastructure would be a highly interdisciplinary endeavour. It would require rethinking everything from connectivity and data warehousing to automated deployment of software in microcontrollers and PLCs. This is an enormous challenge, even if the results of such rethinking are indeed desirable. In any case, one needs to be aware that these complexity challenges need not be insurmountable. In the end, it is largely a question of continuous education of engineers, old and new, that can overcome them. The real problem, then, could lie in ensuring that the engineers have the necessary skills to master the challenges that they will encounter.