My research topic is about robustly leveraging cloud resources for control systems, such as compensation for poor network conditions and implementations of complex control algorithms. Currently the thing I have worked on is adapting a controller to time delays and packet losses by simulating the system. In the future, focus will probably be on the implementation aspects of known control laws, continuing the work of one of my predecessors.

Maintenance and Evolution

The classic method for developing, waterfall, has faded in favour of agile development. Being adaptive to what happens enables greater focus on urgent matters, such as evolving a feature. I have however heard complains about companies using the term "agile" improperly and how the increased efficiency can be wasted on meetings instead of being productive.

For research, hotswapping control algorithms without causing instability would be relevant for my topic, allowing control systems to seamlessly evolve. This could be applicable not only for cloud controlled systems but maybe also classically implemented microcontrollers. Running the algorithm remotely however can provide easier diagnostics and maintenance.

Commercially, there already exists several different cloud providers. From only the platform as a service to function as a service. There is also the possiblity to use telemetric data in order to focus development effort, this however can cause issues with privacy (such as the Audacity controversy). A trusted party (however that would be created) for telemetry could hence be an opportunity, although probably with a low, if any, return of investment.

Security and Privacy

Providing security guarantees is an important selling point, especially in the field of automatic control where software controls physical interactions in the real world. Software bugs or hijacking of systems may have fatal or expensive consequences. For more classial software applications security conserns would be more related to keeping data secure and the absence of bugs, where flaws have more indirect consequences (someone still has to perform an action, or be unable to, based on the extracted data).

Research areas include larger focus on integrated testing which maybe could be automatically generated. Akin to adversarial networks, there could be methods to codevelop processes or emulated users focused solely on breaking the sold product/service.

Privacy, especially with data being processed and stored in geographically distant servers, is a weakness that has to be robustly solved for companies handling sensitive or confidential data (i.e. a lot of them). It thus

goes hand in hand with security. Every now and then there are news articles regarding poorly handled data (such as student information but also on government levels with the Transport Agency). Research areas include the application of homomorphic encryption, allowing encrypted data to be processed without decrypting. From what I've understood, there are still improvements needed before allowing arbitrary functions in real-time. With sufficient performance this could be very useful for my field, allowing measured signals to remain encrypted when applying control laws.

Human Factors

Software engineering could be seen as a way to remove deviations from best practices, allowing higher quality products and easier collaboration. Although I think creativity can be more fun than following strict guidelines, quality ensurance is important. Still, someone needs to actually invent the concepts. A typical usage of AI would be to detect and correct human errors (the definition of AI can easily become blurry here, would e.g. a linter count as AI?), although that description is very broad and already in use for e.g. grammar and writing. From my experience the opinion towards AI, applications using "big data" and server halls varies greatly; as with all technologies there are early adopters, skeptics and everything inbetween. Research opportunities could be how to affect the public opinion of these usages. For example, by falsely claiming an AI controls a vehicle and see how test subjects behave. There could also be identification of overhyped techniques as a way to predict when companies are only using buzzwords without actually creating value, perhaps ironically via clustering?

Time will tell how cloud nodes will be applied in the field of automatic control, but I have noticed several companies being interested in exploring possibilities. It will increase the importance of real-time data processing. This will also majorly depend on hardware, which limits data speed and bandwidth. With 6G already on the way there will be plenty of capacity to exploit. One of the possible applications of real-time data processing would be online learning about a system.

Currently a very actual question is power consumption, which should affect computer usage. Heavy computations (such as machine learning algorithms) should be efficiently implemented, which belongs in the field of software engineering. This is also relevant to overengineered solutions consuming irrational resources for marginal gain. While difficult due to hardware dependence, I hope power efficiency becomes a more important aspect.

I think software engineering and machine learning will remain separate fields; machine learning is more than just combining preexisting items and writing some glue code. For me the standout part of ML systems is the mathematical aspect and training the model. The mathematical model should definitely not be a part of software engineering. The implementation itself is just a program with large amounts of data, which should adhere to software engineering practices though, since big data is not specific to machine learning. The training of the model is similar to testing and tuning in other fields, with e.g. performance measures. As such I think training itself should mainly be done by domain experts. I would classify data preprocessing more as statistics than software engineering, although it is useful to have guidelines and best practices.