# Informatik 3 Lab 0

09.04.2020

Konrad Ukens s0572411
Pavel Tsvyatkov s0559632
Robin Jaspers s0568739
Niklas Lengert s0563290

1. **Explain why professional software that is developed for a customer is not simply the programs that have been developed and delivered.**

   A customer usually can't understand the program to the same extent as the one who created it, as they are only handed a finished product with which they most likely interact through a surface-level user interface. Thus, software, especially for clients/customers always must include proper documentation. This is also how Sommerville defines software: as both the programs and the documentation, which enables users to learn the new system and understand it to the amount necessary to use it effectively, avoid errors, and integrate it into their total operations. The software maintenance also plays a crucial part.

2. **It is important for software teams to stay connected. How will your team work this semester when you probably won't get to meet face-to-face?**

   We are going to use a combination of WhatsApp, Google Docs, Zoom (or BigBlueButton), Discord or similar chat programs. The combination results from each type of program fulfilling a certain need: Google Docs for creating and updating the to-be-handed-in report, and the chat programs for live communication at scheduled appointments to work on the report. If an assignment warrants it, a task distribution tool such as Trello might also be included.

3. **There is much discussion at the moment of using mobile phone tracking to discover potentially exposed individuals so that they can be located and quarantined. A short (1:14) video can be seen here on the Textonix site, a newspaper article is at the Tampa Bay Creative Loafing site. What are some of the societal challenges of building such software? Does it make a difference if the data is collected**

**centrally from telephone company data or decentrally using bluetooth technology?**

Balancing privacy intrusion vs. anonymity vs. precision (of data and tracking). Building trust in citizens to adapt and use any kind of tracking system to the amount that the resulting data can be used with a sufficiently high reliability. Ensuring that data collection is ONLY used for crisis situations, possibly with a clear timeframe of when it can be deactivated. Centralizing all the collected data has several pros and cons. Some of the pros are a better overview over all your data, a higher efficiency when handling and working with the data. The cons are a more vulnerable environment when it comes to protection or hardware failure/damage.

4. **What are the emergent properties that could show up with a system such as one described in #3?**

   **Functional properties:**
   Considering the speed of how fast a disease can spread, the system would need to collect and interpret data live (or as close to live as possible), which would result in a live mapping of all users. Given the circumstances and urgency of the situation, the system would most likely also be able to send to users instructions of how to proceed/what to do, if they need to get themselves checked up etc.
   If the system is government validated and it's notifications can be used as a kind of identification, this may speed up the doctor visiting times/patient validation process.

   **Non-functional properties:**
   Only after use for some time will the reliability of the system become clear, which not only depends on the system itself but general acceptance, acceptance amongst users, accuracy of data and ultimately statistics on infection cases and the general impact over an appropriate time frame. This will also be dependent on how usable the system is, how conveniently it integrates into users' every day life and how much effort using it takes. The user profile for such an app is extremely ambiguous, if it is to include all citizens of a country, and thus the acceptance and use of such an app can only be observed once it is in use.
   Regarding security, if the app is meant to reach as many citizens as possible it must be designed to be compatible with all the operating systems that end users have on their phone (considering smartphones, the largest groups will be Apple iOS and Android systems) and all the versions that are being used. Not all users will be keeping their operating system up to date, which means that there will be lots of considerations to be made in this respect.

5. **Why should we spend effort to fix legacy systems? Wouldn't it be better to just make a new system?**

   There are way more systems in this world running on older technology or programming languages than new developed systems. In your time as a software engineer you have to work with a lot of older systems and in order to create a complete new system you would have to use a lot of resources and time. Often it is not efficient to replace an old system instead of updating it.

   Businesses as a whole may also be deeply independent on exactly how their (software) systems are built and interact, and restructuring this may, unless all processes are exactly documented and regarded in a new system, fail to wholly rebuild the old system. This is a risk that companies may not be willing to take, as it might endanger their stream of business.

Time spent on each question:

| | |
|---|---|
| Question 1 | 10 min |
| Question 2 | 5 min |
| Question 3 | 10 min |
| Question 4 | 20 min |
| Question 5 | 15 min |