KERT Post-Mortem #01

This document is intended to give an overview of the issues, attempted solutions and in general what went well and what didn't during the coffee machine outage of the last weeks. It shall also provide lessons learned for the future, an outlook and general curiosities.

Incident window: from 2016-06-27 (~20:35) till 2016-07-28 (~18:16) (but it felt like a lifetime)
People involved: Yves Frank, Anna Mitterer, Ivan Puddu, Max Schrimpf, Karl Wüst, Gregor Wegberg
Post-Mortem authors: Max Schrimpf, Gregor Wegberg

Chronology of the incident

2016-06-27: The day it died

- All of a sudden, the coffee machine stopped working. Up to this day, the root cause is unclear (see "Open Questions").
- As an immediate response, all solder joints of the RFID reader were checked by Bastli as they were the source of past incidents and it was assumed that they might be a possible cause once again. However, according to Bastli, after a first check, everything seemed fine.
- Additionally, the coffee service was restarted and the computer running the coffee service as a whole was rebooted multiple times.
- None of this resolved the issue and the coffee machine was declared dead until people with time for a deeper investigation were found.

Hardware problems

After about two weeks with a non-operating coffee machine and tired students, Karl, Gregor and Max ignored their university obligations and formed the "KaVIS Emergency Response Team" (KERT) (referred to as "we" in the following).

- It was tried to connect the hardware (MDB board and RFID reader) using Serial-to-USB converters to Karl's Linux laptop. Running the "Kaffi" code on that machine, however, failed horribly.
- Connecting the same devices to Gregor's MacBook failed due to missing drivers for the Serial-to-USB converter. These are available on the internet but installation requires System Integrity Protection to be disabled.
- It was also tried to run and observe the code on the original coffee computer. However, surprisingly, this didn't work at all.
- Investigating deeper, we found that neither the connection to the coffee machine, nor the connection to the Legi reader worked or delivered any useful data to the coffee computer.

➔

Hast du

VISIONEN

?

Hier ist noch Platz für deinen Artikel!

Hast du etwas, das du deinen Mitstudenten mitteilen willst? Warst du im Ausland und willst von deinen grossartigen Erfahrungen berichten? Hast du ein Praktikum gemacht und möchtest erzählen, was du erlebt hast? Dann nutze die VISIONEN als Plattform und schick uns deinen Artikel! Alle Infos unter: http://www.vis.ethz.ch/de/visionen/artikel

Oder möchtest du aktiv mithelfen? Dann werde Mitglied der VISIONEN-Kommission. Einfach E-Mail an: visionen@vis.ethz.ch

http://www.vis.ethz.ch

- Connecting the Legi reader to Karl's machine failed. We were able to open the Serial port, but no data was received from the Legi reader.
- Connecting the MDB board to Karl's machine allowed us to dispense just one coffee capsule. Dispensing further ones failed without a clear reason. There was a feeling that maybe the state transitions were wrong, however no hard evidence for that was found at first. Karl later discovered some hints in the logs and talked about it with Ivan.
- The software as a whole was deemed to very likely contain race conditions, as multiple threads might interfere with each other. Each of the main components of the software runs in its own thread (e.g. RFID reader communication, MDB communication, etc.).
- The coffee machine's computer seemed to have hardware problems, too. For example, the first time we started the machine in the oVIS to do some debugging, it took 5 minutes to boot into BIOS. Subsequent boots were much faster. No clear explanation could be found, but we also didn't try to investigate the root cause.
- Due to the observed problems, we assumed the coffee machine's internal Serial port and the RFID reader were broken. In addition, we declared the whole computer running the coffee machine reached its end of life and started discussing alternative devices for this service.
- By being able to dispense at least one coffee capsule, we assumed the MDB board to be in a somewhat working state and the core cause to be inside the software.

Social Aspects

- As time went on, the overt negative reactions by a few students increased.
- Additional information was put on display to explain the various problems and why it takes so long to fix the coffee dispenser.
- As far as Gregor could observe, most students understood the problems at hand and were thankful for the additional information.
- At some point, someone wrote on the informational poster and indicated that "the responsible people" did nothing to fix it. This was received very negatively by us and reminded them about their actual obligations during the study season.

VIS <3 Bastli

- Since Bastli successfully deployed a very similar installation, we asked them for help and discussed possible solutions at length.
- One of the results is our common aim for a shared codebase and similar hardware stacks (see "Outlook" and "Related Resources" for additional details).
- As all involved VIS members have only very limited knowledge in the realm of hardware and how to test it on a low level, we asked Bastli for help multiple times. The work was very productive and they were very helpful throughout!

More hardware problems

- In a further examination of the Legi reader, one of the Bastli members declared the device to be dead. The day we were able to get everything running again, another Bastli member told us everything was fine with the reader. The only thing he changed was to connect the reader to a Serial-to-USB adapter and hand it over to us.
- The MDB board was connected to a RS232 Serial port, we also moved it to a USB connection (via Serial-to-USB converter). Now we were able to connect it to modern hardware (primarily, Karl's laptop).

A MoEB jumps to the rescue <3

- Ivan had a bit of spare time and offered us help with the coffee machine. He investigated the MDB module in the coffee service code. He found the function `_enter_st_inactive` not to be used, although it should have been. A detailed explanation can be found in the `2ee7fa333d678d18b21bc3c4e92961fcd009a4e2` commit message of the coffee machine code (see the GitHub project [1] for additional information).

2016-07-28: The day it worked again

- Bastli delivered our RFID reader, connected to a Serial-to-USB converter, and told us it worked for all they know.
- KERT started to put the coffee machine together again. We used all the existing hardware, with the only difference of connecting both external devices (MDB board and RFID reader) over USB.
- The Legi reader, now connected to a Serial-to-USB converter, was put into an envelope into the coffee machine's body in such a way that it cannot move and possibly short the circuit.
- The coffee computer was kept as is, hardware- and software-wise.
- At first, the coffee machine didn't work due to some debugging-related changes to the coffee service.
- After changing the service's source code to a state where it should work as expected, we were able to dispense coffee capsules at will. However, the RFID reader still didn't work.
- The logging output of the coffee service made it clear that the RFID reader worked, but we didn't get the expected data, i.e. no Legi numbers.
- Using Bastli's source code we were able to read out Legi numbers using Python 3.x, latest `pyserial` and Bastli's RFID reader test script. However, using basically the same source code (VIS's coffee service's Legi module) with Python 2.7 and the installed `pyserial` version didn't work. After some additional experiments, we found the culprit to be the installed `pyserial` version used along with Python 2.7.
- Installing the latest `pyserial` version fixed our troubles with reading out RFID data.

- The problem with reading the Legi was completely due to some bug in `pyserial`. We assume the old version had some incompatibility or bug with our new way to connect the hardware (Serial over USB). Another explanation might be that at some point, someone upgraded the package leading to the installation of a incompatible/buggy `pyserial` version. Anyway, `pyserial` was the main problem of the day.

The day after

- Karl remembered that the coffee machine reboots daily in the night. KERT decided to remove this cronjob as we assume Ivan's fix might solve the underlying problem. Original crontab line: `0 5 * * * /sbin/reboot`.
- We still observe behaviour we cannot explain:
  - There is one report where the machine didn't work for multiple people and a short time after allowed to dispense 9 coffee capsules without a Legi.
  - The log contains unexpected errors and notices of deadlocks.
  - We had to restart the service already once after a deadlock log entry.

Outlook

Persisting problems

- Coffee machine computer is in a very bad shape.
  - Lots of unnecessary applications and libraries installed.
  - Bootprocess is still a bit fishy (e.g. big difference in boot time to BIOS).
  - We installed Python 3 and pip for Python 3 using `apt-get`. We did not remove it as it does not bother Python 2 and the coffee computer should be replaced in the short term!
- Device has not enough USB ports to connect the "Ampel" too.

Lessons learned

- Install dependencies (e.g. Python libraries) only using one way (e.g. `pip` instead of `apt-get python-*`!). Same pitfall for all dependency management tools vs. OS package managers.
- Keep the pip related `requirements.txt` up-to-date and only keep those libraries installed. The `setup.py` seems unnecessary for this service. However, if it is kept, it should follow best practice and pin dependencies to specific versions.
- System management tools (e.g. Puppet) must adhere to `requirements.txt`. It must be clear how and who keeps dependencies up-to-date and makes sure the right ones are installed.
- The coffee service should have a shell (e.g. use Python's `cmd` module). This shell should allow to perform all important operations (e.g. dispense capsule, read Legi using the RFID reader, etc.).
- The software should be able to disable single components for debugging without side effects (e.g. disable the RFID reader).

- Each software component should be replaceable with a mock component. This mock should behave like we expect the mocked component to behave. It should feature detailed logging facilities for debugging purposes.
- Having tests (unit, component, module, integration, ...) would have been great while performing changes to the service's source code. Just to be more confident the changes won't break everything.
- To resolve hardware trouble, talk to the Bastli. They are really nice people and very helpful.
- Work in a team and therefore establish a good communication channel (e.g. group chat). This is much more fun and everyone brings its own expertise to the table.
- The current way of logging events is somewhat chaotic. Especially the use of the logging level (`DEBUG`, `INFO`, `WARNING`, `ERROR`, `CRITICAL`) makes no sense in a lot of cases and is inconsistent.
- Do not develop on live systems. No, really, not even that "one single line" change! Write code on your machine, test it, commit, try out, repeat.
- Use tools! It would have been great to just debug it with PyCharm and remote debugging.
- Fixing a system by rebooting it or restarting the software isn't a sound approach. This was the modus operandi, and likely will stay till we get the long term solution (service rewrite) up and running.
- We should have involved the Codeli commission from the very first day!
- The code should document itself and additional documentation (e.g. inside the VISkb) should be available to resolve issues. Right now the source code uses misleading names (e.g. one of the `dispense` methods doesn't lead to dispensing a coffee capsule but rather performs a peripheral task) and has no clean structure. Further, no documentation is available explaining available configuration parameters, how to get started, etc.

Next Steps

- A long term solution is sought out by Bastli and VIS:
    - Common source code base for both machines (beer and coffee).
    - Similar hardware stack used.
- We aim to get our hands on industry grade hardware for the coffee machine. This would be a very similar device to the one Bastli uses for the beer machine.
    - This device contains a computer we can use to run our software on.
    - In addition, it has a built-in MDB board.
    - Bastli will help us in integrating the current Legi reader with this device in a way that it is well connected and won't break again.

Open Questions

Why did it work before?

- We have no real clue due to the many problems we had to solve.
- The whole system was not in a good shape so it is exaggerated to say it worked.
  - The service was restarted on a regular basis.
  - The whole system was rebooted every night.
- We assume the root cause to be a hardware defect (see next section).

What was the root cause for the outage?

- RFID Reader: Maybe one/multiple of the soldering joints broke, maybe one of the wires got loose, or one/multiple components of the additional board built by Bastli long time ago to allow us to connect the reader to the Serial port broke.
- MDB Board: As far as we can tell, it always worked. The only reason for us to connect it over USB was for conveniently debugging the problems.
- Software: Well, here it gets tricky. The need to restart the coffee machine ever so often may be connected to the bug found by Ivan (missing calls to `_enter_st_inactive`). The need to update `pyserial` might be just due to us using the RFID reader and MDB board over USB. It might well be that `pyserial`'s version we used would work well if we hadn't connected the components over USB.
- Maybe our debugging and changes lead us into additional trouble.
- All in all, it seems like the RFID reader was partially broken and our changes lead to additional problems we had to solve.
- However, as mentioned already, this is just a working theory and might be completely wrong. We have no clear proof to support or disproof the above theory.

Final Words

To be very clear: **The fixes and changes done to solve the incident described in this report are of temporary nature! It is very important to work on a long term solution (see "Outlook"). There is no way the current state of affairs will work in the long run. The coffee supply is still at risk.**

Taking into account the coffee machine broke during the preparation phase and exam session we are very happy with the result. The machine, for now, is in a good enough state and should keep working till we get the long term solution up and running.

The only thing left to say is a huge thank you to all people involved. They spent precious hours to solve the incident and provide free coffee once again to thousands of students (VIS, AMIV, VMP).

Related Resources

[1] Coffee code on GitHub: https://github.com/VIS-ETH/kaffi/

Glossary

- KERT: KaVIS Emergency Response Team. The group of people primarily working on resolving KaVIS (the coffee service provided by VIS) incidents.
- MDB: "Multi-Drop Bus" is part of the "Multi-Drop Bus / Internal Communication Protocol" standard by the National Automatic Merchandising Association (NAMA) used by vending machines. Our software uses this bus and protocol to communicate with the coffee capsule dispenser.
- MDB board: Allows to connect our computer to the vending machine over a serial RS232 port. It acts as a converter between the 9 bit MDB protocol and the serial 8 bit protocol.