Internship summary

aicas GmbH – Lukas Miedema

From 1st of February till the end of June (2019) I took an internship at aicas GmbH as part of my master Computer Science at the UT. aicas (spelled with a lowercase "a") is a German company located in the city of Karlsruhe, near the French border and not even 200 kilometers from Switzerland. aicas develops and maintains a special Java Bytecode VM called *JamaicaVM* for realtime software applications. Realtime applications are applications where extreme determinism is required, and with JamaicaVM it is possible to use Java (or other JVM languages) for this purpose. Their VM has a completely deterministic garbage collector, and also other aspects of the VM are written in a way that their performance is highly predictable. Furthermore, this VM also runs on a lot of embedded platforms where this predictability is more appreciated.

aicas employs about 40 people, and the company is located in the Technologiepark in Karlsruhe. There, it is situated on the fourth floor of one of the buildings (with a great view). The company is further structured in a few teams, with focus both on building and maintaining the VM as well as building solutions and frameworks on top of the VM.

My assignment

One key aspect of JamaicaVM – one where it differs from many other Java Bytecode VMs – is that it doesn't have a Just-In-Time (JIT) compiler. This is in part because a JIT produces very good peak and average-case performance, but is not deterministic. Instead, JamaicaVM ships with two special tools called the *Builder* and the *Accelerator*. Without going into too much detail: these two tools can perform Ahead-Of-Time (AOT) compilation of JVM bytecode to machine code, boosting performance without a JIT. The team that maintains this is called the *Tools* team, and as part of my assignment I worked within this team.

My assignment was to work on the *Accelerator* tool. This tool takes a JAR file and compiles all the bytecode in that file to native code. Then, it adds all this native code back to the JAR file as a shared library. At runtime, when loading this JAR, JamaicaVM sees this native code and use it instead of interpreting the bytecode.

This native code has unique ways of interacting with JamaicaVM – it needs to be as fast as possible, and as such has deep dependencies on structures within the VM. My assignment was to take a look at how this interaction takes place and improve it by defining a clear interface between the VM and the accelerated code. I designed a new VM Public Interface – an interface via which the native code generated by the Accelerator can interact with JamaicaVM. This is a first step in making the interaction between the compiled bytecode and JamiacaVM easier to maintain.