

Bachelorarbeit

Entwurf und Implementierung einer
hochperformanten, serverbasierten
Kommunikationsplattform für Sensordaten
im Umfeld des automatisierten Fahrens in Rust

Michael Watzko

Sommersemester 2018
14.02.2018 - 22.06.2018

Erstprüfer: Prof. Dr. rer. nat. Dipl.-Inform. Manfred Dausmann
Zweitprüfer: ... Hannes Todenhausen



Firma: IT Designers GmbH
Betreuer: Dipl. Ing. (FH) Kevin Erath M.Sc.

“Alle Zitate aus dem Internet sind wahr!”

Albert Einstein

“Rust is a vampire language, it does not reflect at all!”

<https://www.youtube.com/watch?v=-Tj8Q12DaEQ>

Inhaltsverzeichnis

1	Einleitung	1
1.1	Motivation	1
1.2	Zielsetzung	1
1.3	Aufbau der Arbeit	1
2	Hochperformant, serverbasierte Kommunikationsplattform ...?	1
2.1	Low-Latency + Entwurfsmuster + Algorithmen?	1
2.2	ASN.1	1
2.3	PER	1
2.4	MEC-View Server und Umgebung	1
3	Die Programmiersprache Rust	1
3.1	Geschichte	2
3.2	Was ist Rust?	2
3.3	Sprachfeatures	2
3.3.1	Option	2
3.3.2	Result	2
3.4	Warum Rust?	2
3.5	Kernfeatures	2
3.6	Schwächen	3
3.7	Performance Fallstricke	3
3.8	Aktuelle Verwendung von Rust	3
4	Anforderungen	I
4.1	Funktionale Anforderungen	I
4.2	Nichtfunktionale Anforderungen	I
4.3	Kein Protobuf weil	I
5	Systemanalyse	I
5.1	Systemkontextdiagramm	I
5.2	Schnittstellenanalyse	I
5.3	C++ Referenzsystem	I
6	Systementwurf	I
6.1	Änderungen bedingt durch Rust	I
7	Implementierung	I
8	Auswertung	I
9	Zusammenfassung und Fazit	I
	Literatur	II

List of abbreviations

III

1 Einleitung

1.1 Motivation

1.2 Zielsetzung

1.3 Aufbau der Arbeit

2 Hochperformant, serverbasierte Kommunikationsplattform ...?

2.1 Low-Latency + Entwurfsmuster + Algorithmen?

2.2 ASN.1

2.3 PER

2.4 MEC-View Server und Umgebung

3 Die Programmiersprache Rust

3.1 Geschichte

3.2 Was ist Rust?

Rust ist...

TODO: Rust -> MIR -> assembler
TODO: MIR/assemblerbeispiele?

```
fn main() {  
    println!("Hello World");  
}
```

3.3 Sprachfeatures

3.3.1 Option

Abbildung 1: "Hello World" in Rust

3.3.2 Result

3.4 Warum Rust?

"[.]Leute, die [.] sichere Programmierung haben wollen, [.] können das bei Rust haben, ohne die [von D] undeterministischen Laufzeiten oder Abstraktionskosten schlucken zu müssen." [5]

"It's not bad programmers, it's that C is a hostile language" (Seite 54, [7])

"I'm thinking that C is actively hostile to writing and maintaining reliable code" (Seite 129, [7])

"[.] Rust makes it safe, and provides nice tools" (Seite 130, [7])

"Rust hilft beim Fehlervermeiden" [4]

"Rust is [.] a language that cares about very tight control" [3]

TODO: unused only rust [1]

3.5 Kernfeatures

<https://www.youtube.com/watch?v=d1uraoHM8Gg>

TODO: no dangling pointers

TODO: no need for a runtime, all static analytics

TODO: memory safety

TODO: data-race freedom

TODO: active community

TODO: concurrency: no undefined behavior

TODO: ffi binding Foreign Function Interface¹

TODO: zero cost abstraction

¹ Beschreibt den Mechanismus wie ein Programm das in einer Programmiersprache geschrieben ist, Funktionen aufrufen kann, die einer anderen Programmiersprache geschrieben wurden. [2]

TODO: package manager: cargo

<https://www.youtube.com/watch?v=-Tj8Q12DaEQ>

TODO: static type system with local type inference

TODO: explicit notion of mutability

TODO: zero-cost abstraction *(do not introduce new cost through implementation of abstraction)

TODO: errors are values not exceptions TODO: no null

TODO: static automatic memory management no garbage collection

TODO: often compared to GO and D (44min)

3.6 Schwächen

<https://www.youtube.com/watch?v=-Tj8Q12DaEQ>

TODO: compile-times

TODO: Rust is a vampire language, it does not reflect at all!

TODO: depending on the field -> majority of libraries?

3.7 Performance Fallstricke

TODO: [6]

3.8 Aktuelle Verwendung von Rust

TODO: firefox

<https://www.youtube.com/watch?v=-Tj8Q12DaEQ>

TODO: GTK binding heavily to rust

4 Anforderungen

4.1 Funktionale Anforderungen

4.2 Nichtfunktionale Anforderungen

4.3 Kein Protobuf weil

5 Systemanalyse

5.1 Systemkontextdiagramm

5.2 Schnittstellenanalyse

5.3 C++ Referenzsystem

6 Systementwurf

6.1 Änderungen bedingt durch Rust

7 Implementierung

8 Auswertung

9 Zusammenfassung und Fazit

Literatur

- [1] Jim Blandy. Why Rust? Trustworthy, Concurrent System Programming. Englisch. 2015. URL: <http://www.oreilly.com/programming/free/files/why-rust.pdf> (besucht am 01.06.2017).
- [2] Wikipedia contributors. Foreign function interface — Wikipedia, The Free Encyclopedia. [Online; accessed 14-February-2018]. 2018. URL: https://en.wikipedia.org/w/index.php?title=Foreign_function_interface&oldid=825105351.
- [3] fgilcher. Subreddit Rust. fgilcher kommentiert. Englisch. 3. Nov. 2017. URL: https://www.reddit.com/r/rust/comments/7amv58/just_started_learning_rust_and_was_wondering_does/dpb9qew/ (besucht am 01.06.2017).
- [4] Sebastian Grüner. “C ist eine feindselige Sprache”. Der Mitbegründer des Gnome-Projekts Federico Mena. Deutsch. 22. Juni 2017. URL: <https://www.golem.de/news/rust-c-ist-eine-feindselige-sprache-1707-129196.html> (besucht am 14.02.2018).
- [5] Felix von Leitner. Fefes Blog. D soll Teil von gcc werden. Deutsch. 22. Juni 2017. URL: <https://blog.fefe.de/?ts=a7b51cac> (besucht am 14.02.2018).
- [6] Llogiq. Llogiq on stuff. Rust Performance Pitfalls. Englisch. URL: <https://llogiq.github.io/2017/06/01/perf-pitfalls.html> (besucht am 01.06.2017).
- [7] Federico Mena Quintero. Replacing C library code with Rust. What I learned with libsvg. Englisch. URL: <https://people.gnome.org/~federico/blog/docs/fmq-porting-c-to-rust.pdf> (besucht am 14.02.2018).

Glossar

Foreign Function Interface Beschreibt den Mechanismus wie ein Programm das in einer Programmiersprache geschrieben ist, Funktionen aufrufen kann, die einer anderen Programmiersprache geschrieben wurden. [\[2\]](#) . [2](#)

Abbildungsverzeichnis

1	“Hello World” in Rust	2
---	---------------------------------	---