Softwareentwurf

FlipCoin und B3
25.05.2021

Wichtige Hinweise:

- ***Die in diesem Dokument aufgeführten Beschreibungen in Kursivschrift (außer einigen Abschnittüberschriften) sind beispielhaft und erläuternd und müssen aus dem fertiggestellten Bericht entfernt werden.***
- ***Kennzeichnen Sie welcher Abschnitt dieses Berichts von welchem Teammitglied erstellt wurde.***

1 Produktbeschreibung

Beschleunigung: Jeder Truck hat die Möglichkeit zu beschleunigen.
Bremse: Jeder Truck hat die Möglichkeit zu bremsen.
Wähle Führer: Ein Algorithmus entscheidet und erkennt, welcher Truck der führende Truck wird.
Lese Fahrzeugstatus aus: Die Trucks können ihren Status (Geschwindigkeit, Position im Platoon) angeben.
Passe an: Alle Trucks hinter dem Führer passen sich an(Geschwindigkeit, Abstand).
Lenken: Die Trucks haben die Möglichkeit zu lenken.
Stoppen: Die Trucks haben die Möglichkeit zu stoppen.

Unser Projekt kann erkennen, wenn ein Truck das Platoon verlässt. Es wird eine Nachricht weitergegeben, um zu erkennen, an welcher Stelle es eine “Lücke” gibt. Die Lücke wird geschlossen indem Trucks beschleunigen oder bremsen.

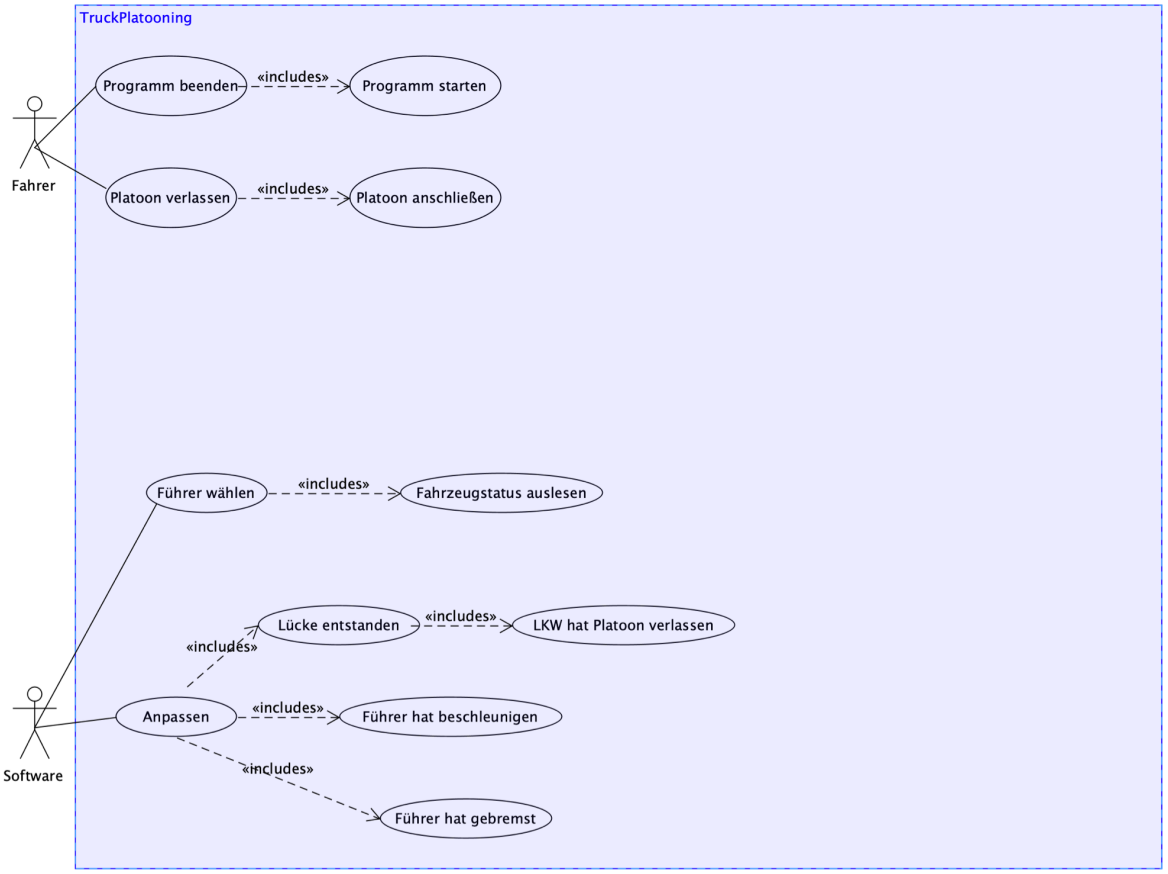
Der führende Truck bestimmt die Schnelligkeit aller anderen Trucks. Beschleunigt er, beschleunigen auch alle anderen. Dasselbe passiert bei dem Bremsen.
(Geschrieben von Florian Eifert)

2 Technologien

- Wir haben uns für **Microservices** entschieden, um die Anwendung in kleine Services aufzuteilen, die unabhängig voneinander laufen.
- Als Programmiersprache haben wir uns für **Java** entschieden
- Als Framework kommt bei uns **Spring Boot** für das Backend zum Einsatz
- Als Versionsverwaltung Tools werden wir mit **Git** und **Github** arbeiten.
- Als DevOps werden wir eventuell **Jenkins** einsetzen.
- Wir nutzen als Build-Management-Tool **Maven**, um die Abhängigkeiten automatisiert zu managen.

(Geschrieben von Lemine El Agheb)

3 Use-Cases

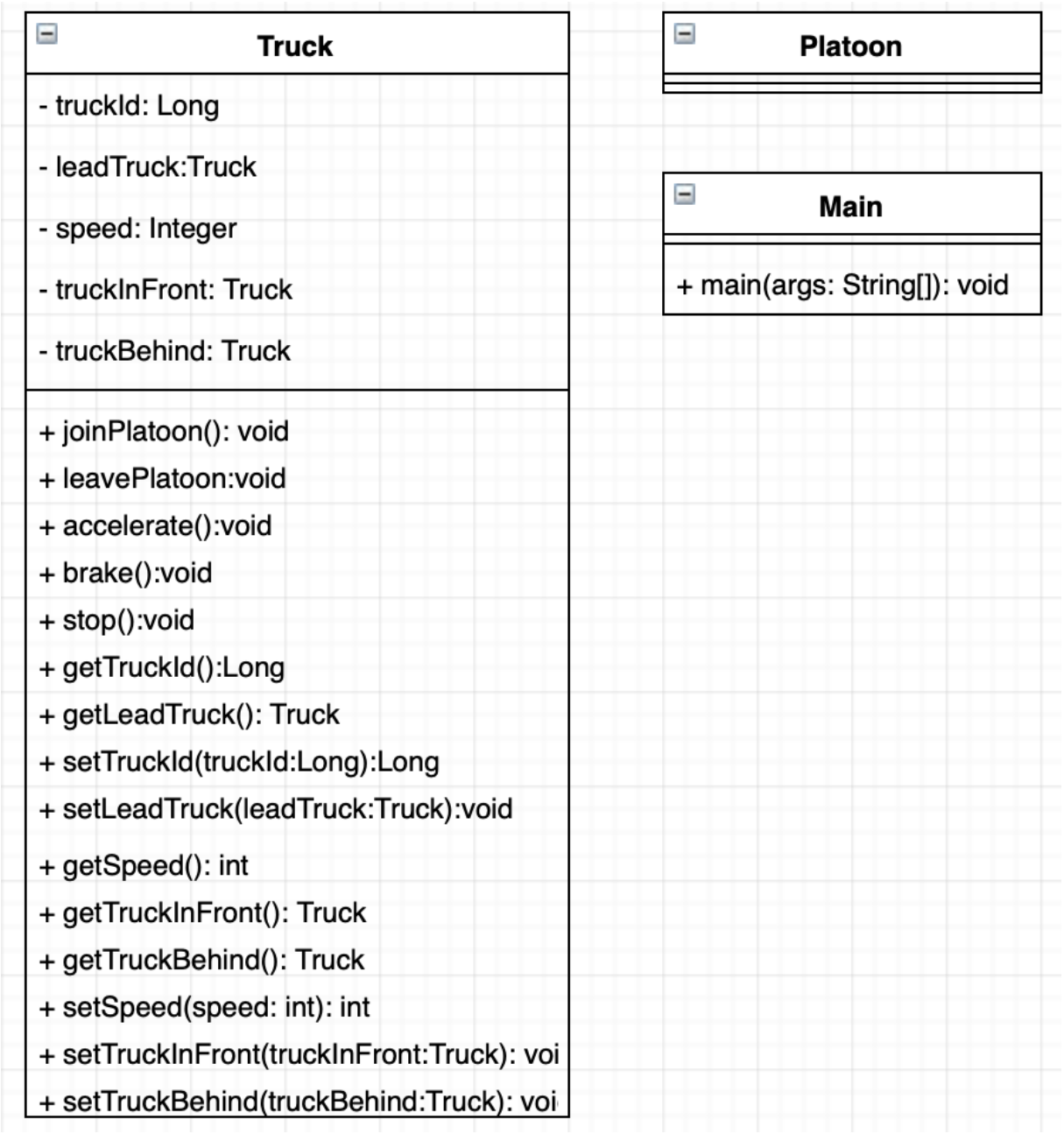


In unserem Use-Case Diagramm besteht “Anpassen” aus bremsen, beschleunigen, lenken und stoppen.
Der Fahrer kann das Programm starten und beenden. Damit das Programm beendet werden kann, muss es schon gestartet worden sein.

Der Fahrer kann sich einem Platoon anschließen oder es verlassen. Um das Platoon verlassen zu können, muss der Fahrer sich dem Platoon angeschlossen haben.

Im Falle, dass eine Lücke entsteht, der Führer beschleunigt oder bremst, passt die Software die Geschwindigkeit an.
Damit die Geschwindigkeit angepasst wird, muss eine Lücke entstanden sein, der Führer beschleunigt oder gebremst haben.
Damit eine Lücke entsteht, muss ein LKW das Platoon verlassen haben.
(Geschrieben von Lemine El Agheb + Calvin Kluk)

4 Aufbau der Software

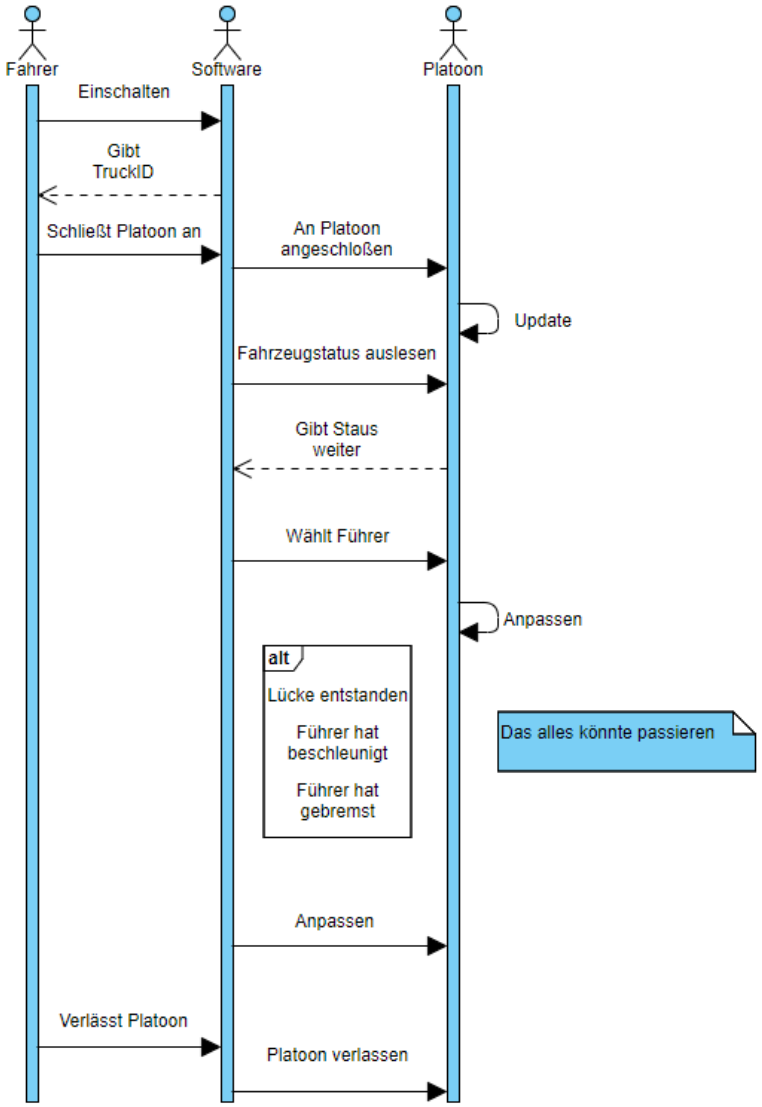


- truckId: Eine Eindeutige Truck ID
- leadTruck: Zeigt den aktuellen führenden Truck
- truckInFront: Zeigt den Truck davor
- truckBehind: Zeigt den Truck dahinter
- PID: Eine Individuelle Prozess ID (anders als truckID)

- joinPlatoon(): Wird ausgeführt um den Platoon bei zu treten
- leavePlatoon(): Wird ausgeführt um den Platoon zu verlassen
- accelerate(): Wird ausgeführt um die Geschwindigkeit zu erhöhen
- brake(): Wird ausgeführt um die Geschwindigkeit zu verringern
- stop(): Wird ausgeführt um stehen zu bleiben
- getTruckId(): Liest die Truck ID aus
- getLeadTruck(): Liest den führenden Truck aus
- setTruckId(): Belegt eine Truck ID
- setLeadTruck(): Belegt den führenden Truck
- getSpeed(): Liest die Geschwindigkeit aus
- getTruckInFront(): Liest den vor fahrenden Truck aus
- getTruckBehind(): Liest den hinterher fahrenden Truck aus
- setSpeed(): Belegt die Geschwindigkeit
- setTruckInFront(): Belegt den vor fahrenden Truck
- setTruckBehind(): Belegt den hinterher fahrenden Truck

(Geschrieben von Florian Eifert + Manuel Künzel)

5 Kommunikationsfluss



Zuerst startet der Fahrer die Software. Die Software gibt daraufhin eine TruckID. Der Fahrer darf nun dem Platoon beitreten. Das Platoon updated sich. Die Software fragt die Fahrzeugstatus (PID, TruckID,…). Das Platoon gibt die Information an Software weiter. Die Software wählt mit der Information einen Führer. Das Platoon passt sich dem Führer Fahrzeug an. Es können folgende Events eintreten: Lücke entstanden, Führer hat beschleunigt oder Führer hat gebremst. Falls ein Event eintreten sollte, sagt die Software dem Platoon, wie diese sich anpassen sollen. Zum Schluss kann der Fahrer sich dazu entscheiden, das Platoon zu verlassen.
(Geschrieben von Calvin Kluk + Manuel Künzel)

6 Wahlalgorithmen

Bully-Algorithmus:
Jeder Truck bekommt beim Platoon eintreten eine individuelle PID (Prozess ID). Die PID wird größer für jeden Truck hinter diesem. Die PID startet bei 0. Der Truck mit der Höchsten PID ist der Führer.

Beispiel:
Unser Platoon besteht aus 3 Trucks, Rot(PID 2), Blau(PID 1) und Grün(PID 0). Jetzt fährt ein neuer Truck zwischen Blau und Grün. Das ist der Orangene Truck. Weil ein neuer Truck eingetreten ist, bekommen alle Trucks eine neue PID. Rot(PID 3) ist immer noch ganz vorne. Blau(PID 2) hat nun eine erhöhte PID aber ist nicht der Führer. Orange(PID 1) ist vor Grün eingefahren und erhält somit eine 1. Grün(PID 0) ist weiterhin ganz hinten am Platoon.

Somit wissen wir stets wer unser Führer ist.

Beispiel 2:
Angenommen Truck Rot verlässt das Platoon. Orange führt jetzt eine Auswahl durch. Orange fragt nach allen PIDs. Rot gibt keine Antwort weil er verlassen hat. Grün gibt keine PID weil seine kleiner ist. Blau gibt seine PID weiter. Orange erkennt dass seine PID kleiner ist als Blau also hört Orange jetzt auf Blau. Ebenfalls erkennt Blau dass er jetzt die höchste PID hat und gibt allen anderen Trucks im Platoon die Nachricht weiter, dass er jetzt der Führer ist.
(Geschrieben von Lemine El Agheb + Florian Eifert)

7 Quellen

- Backend-Technologien Vorlesungsunterlagen
- https://www.spring.io
- https://martinfowler.com/articles/microservices.html
- https://online.visual-paradigm.com/drive/#diagramlist:proj=0&new=SequenceDiagram