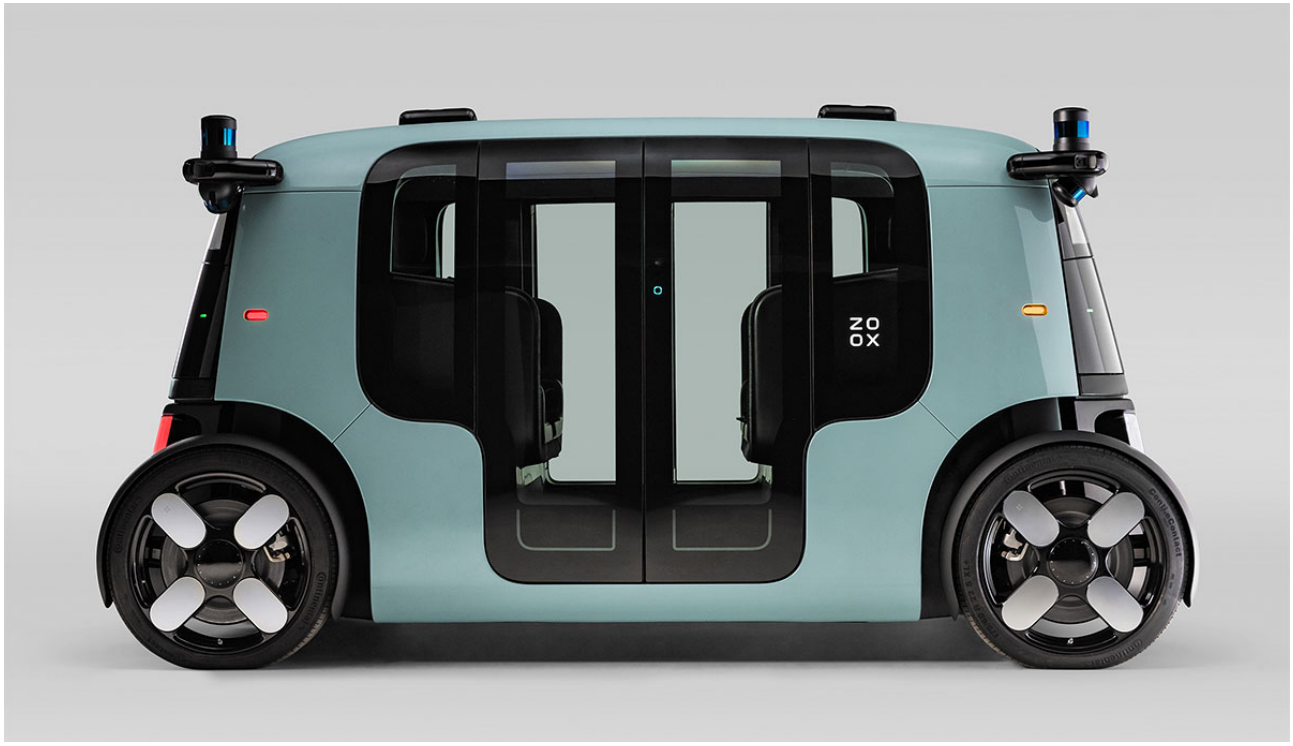
Software Engineering I - Teil 02

Projekt „Autonom fahrendes Auto“



https://zoox.com/ | https://www.youtube.com/watch?v=pQVpjuh6tZY

https://www.autox.ai/en/index.html | https://www.youtube.com/watch?v=7GVL9Na1_9Q

S01 | Komponenten und SOA

Das autonome Fahrzeug verfügt über eine zentrale Steuerungseinheit. Diese Einheit verwaltet einen EventBus. An den EventBus sind die Bauteile Elektromotor, LED-Scheinwerfer, Blinker, Bremsen, GPS, Kamera und Lidar angeschlossen.

Bauteil	Events
Elektromotor	EngineOn, EngineOff, IncreaseRPM(deltaRPM,seconds) DecreaseRPM(deltaRPM,seconds)
Batterie[1]	Der Elektromotor bezieht die benötigte Energie aus einer Batterie mit Zeichen aus dem Pool [0, 1].
LED-Scheinwerfer	LEDOn, LEDOff, LEDDimmed, LEDHighBeam
Bremslichter	BrakeLightOn, BrakeLightOff
Blinker	LeftIndicatorOn, LeftIndicatorOff, RightIndicatorOn, RightIndicatorOff, HazardWarningOn, HazardWarningOff
Bremsen	BrakeSet(percentage)
GPS	GPSOn, GPSOff, GPSConnectSatellite(String frequency)
Kamera	CameraOn, CameraOff
Lidar	LidarOn, LidarOff

Die Camera existiert in zwei Varianten CameraV1 und CameraV2, realisiert als dynamisch austauschbare Komponente. Das Lidar existiert in den zwei Varianten LidarNG und LidarXT. In einer zentralen Configuration – realisiert als Konfiguration im Format JSON – ist die eingesetzte Variante der Camera sowie des Lidar definiert. Basierend auf der Definition in der JSON-Datei wird die korrespondierende Komponente geladen.

Jedes Bauteil ist eine digital signierte Komponente.

1 Struktur ist definiert im Aufgabenteil „Composite“.

S02 - S07 | Entwurfsmuster

Themenbereich	Entwurfsmuster
Erzeugung und Struktur	**Builder**, Factory **Adapter**, **Bridge**, **Composite**, Decorator, **Facade**, Flyweight, Proxy
Verhalten	**COR**, **Command**, Filter, Iterator, **Mediator**, **Memento**, **Observer**, **State**, **Strategy**, Template, **Visitor**

S02 | Builder

- Amazon Zoox | 1 Chassis, 1 Elektromotor, 1 Batterie, 4 LED-Scheinwerfer, 4 Bremslichter, 4 Blinker, 4 Türen, 2 Sitzbänke, 4 Räder, 4 Bremsen, 2 GPS, 4 Kamera, 4 Lidar
- AutoX | 1 Chassis, 1 Elektromotor,1 Batterie, 2 LED-Scheinwerfer, 2 Bremslichter, 4 Blinker, 4 Türen, 6 Sitze, 4 Räder, 8 Bremsen, 2 GPS, 2 Kamera, 4 Lidar

S02 | Adapter

Ladestation hat 2-poligen Anschluss. Für das Laden der Batterien wird ein Adapter genutzt. Amazon Zoox | 4-polig; AutoX | 3-polig

S02 | Bridge

Der Elektromotor existiert in den Varianten EngineX und EngineNG. Der Elektromotor EngineX verbraucht 4 Energieeinheiten/RPM je Iteration. Der Elektromotor EngineNG braucht 3 Energieeinheiten/RPM je Iteration. Der Verbrauch einer Energieeinheit wird durch die Umwandlung von 1 nach 0 in einer Zelle der Batterie simuliert.

In einer zentralen Configuration – realisiert als Enumeration – ist die eingesetzte Variante des Elektromotors definiert.

S03 | Composite

Eine Batterie besteht aus 500 Hauptzellen. Eine Hauptzelle besteht aus 100 Subzellen. Eine Subzelle hat 5 Zellen. Das Attribut energy einer Zelle kann die Werte 0 (entladen) oder 1 (geladen) annehmen. Amazon Zoox | 8 Batterien; AutoX | 4 Batterien.

S03 | Facade

startup	EngineOn, LEDOn, GPSOn, GPSConnectSatellite(118.75), CameraOn, LidarOn, LeftIndicatorOff
move(deltaRPM,seconds)	LeftIndicatorOff, RightIndicatorOff, LEDDimmed, IncreaseRPM(deltaRPM,seconds), BrakeSet(0), BrakeLightOff
leftTurn(deltaRPM,seconds)	LeftIndicatorOn, DecreaseRPM(deltaRPM,seconds), BrakeSet(70), BrakeLightOn

rightTurn(deltaRPM,seconds)	RightIndicatorOff, DecreaseRPM(deltaRPM,seconds), BrakeSet(70), BrakeLightOn
stop	BrakeSet(100), BrakeLightOn
emergencyStop	BrakeSet(100), BrakeLightOn, HazardWarningOn, EngineOff, LEDHighBeam, CameraOff, LidarOff
shutdown	BrakeSet(100), EngineOff, BrakeLightOff LEDOff, HazardWarningOff, GPSOff, CameraOff, LidarOff

S03 | Command

Über einen elektronischen Schlüssel wird das autonome Fahrzeug aktiviert oder deaktiviert.
Auf dem elektronischen Schlüssel ist das mit AES verschlüsselte Passwort [i] ZooxSDC73 für den Amazon Zoox und [ii] AutoX23 für AutoX gespeichert. Die zentrale Steuerungseinheit im autonomen Fahrzeug ist mit einem Empfangsmodul für das Signal (verschlüsseltes Passwort) des elektronischen Schlüssels verbunden. Die zentrale Steuerungseinheit entschlüsselt das Signal. Bei korrektem Passwort wird das autonome Fahrzeug aktiviert oder deaktiviert.

Amazon Zoox | Auf der linken und rechten Seite existiert ein Taster mit einem Sensor zum Öffnen der Türen. AutoX | An den je zwei Türen auf der linken und rechten Seite existiert ein Sensor zum Öffnen. Der Sensor sendet die Kommandos Open und Close an den Elektromotor je Tür.

Beide Fahrzeuge verfügen über einen Notruf-Button. Durch Drücken des Notruf-Button wird das Servicecenter informiert.

S04 | Memento

Aus Aspekten der Vereinfachung wird das Verhalten des autonomen Fahrzeuges mit nachfolgend aufgeführten Einstellungen zentral konfiguriert. [i] rejectDrunkenPassenger = [true | false, default: true], [ii] stopByPoliceRequest = [true | false, default true], [iii] allowDriveByNight = [true | false, default: true], [iv] behaviorWithNaggingPassengers = [doNothing | stopAndWaitForExcuse, default stopAndWaitForExcuse], [v] musicDuringDrive = [ac/dc, helene fischer, default: ac/dc].

In einem Menü – realisiert als dediziert ausführbare Konsolen-Applikation – werden die Optionen [i] print, [ii] set parameter, [iii] undo und [iv] exit angeboten.

Für die zentrale Konfiguration wird die Hauptapplikation mit dem Parameter -config gestartet.

Bei Auswahl von [i] print wird eine Übersicht der Parameter mit den aktuellen Einstellungen angezeigt. Nach Anzeige der Übersicht erfolgt der Rücksprung zum Menü.

Bei Auswahl von [ii] set parameter wird eine Übersicht der Parameter mit den aktuellen Einstellungen angezeigt und die id des zu ändernden Parameters abgefragt. Der Benutzer gibt die id des zu ändernden Parameters ein und das System fragt den neuen Wert des Parameters ab, enter value for [name] | current [current value] | allowed [allowed values] > [new value]. Eine ungültige Eingabe ist solange zu wiederholen bis diese gültig ist.

S04 | Observer

- Amazon Zoox | Auf der linken und rechten Seite existiert ein Taster mit einem Sensor zum Öffnen der Türen. AutoX | An den je zwei Türen auf der linken und rechten Seite existiert ein Sensor zum Öffnen.
- Die Temperatur jeder Batterie wird mit einem Sensor durch die zentrale Steuerungseinheit überwacht und bei Änderungen automatisch informiert.
- Die zentrale Steuerungseinheit nutzt Ultraschallsensoren für die Messung von Abständen zu Objekten rund um das Fahrzeug. Bei Änderungen eines oder mehrerer Abstände wird die Zentraleinheit automatisch informiert.
  Amazon Zoox und AutoX | 8 Ultraschallsensoren.

S04 | State

Initial befindet sich die Tür im Status Closed. Bei einem Signal im Status Closed öffnet die Tür und wechselt in den Status Open. Bei einem Signal im Status Open schließt die Tür und wechselt in den Status Closed.

S05 | Servicecenter

Composite | Dem Supervisor sind zwei Teams [i] AutoX und [ii] Zoox unterstellt. Jedes Team hat drei Mitarbeiter. Zwei Mitarbeiter sind spezialisiert auf Wartung und einer auf Notfälle. Ein Mitarbeiter hat den Status verfügbar, beschäftigt oder abwesend. **CoR** | Nach Betätigung des Notrufkopfes wird ein verfügbarer und verantwortlicher Mitarbeiter im korrespondierenden Team informiert. **Mediator / Bridge / Strategy / Observer** | Für die Zutrittskontrolle verfügt das Servicecenter über eine Personenschleuse. Für die Authentifizierung in der Zutrittskontrolle ist entweder die Variante Iris oder Fingerabdruck verbaut. Fingerabdruck und Iris werden jeweils dargestellt als 10x10-Matrix mit zufällig gewählten Zeichen aus dem Pool [. | : | * | +). Aus Aspekten der Vereinfachung ist einem Mitarbeiter ein Fingerabdruck und eine Iris (realisiert als Komposition) zugeordnet. Die Information zu Fingerabdruck und Iris sind einheitlich verschlüsselt mit MD5 oder SHA256 auf der Magnetkarte zu dem Mitarbeiter gespeichert. Eine Magnetkarte ist einem Mitarbeiter zugeordnet. Der Mitarbeiter kennt die Magnetkarte. Die Magnetkarte kennt nicht den Mitarbeiter. Nach erfolgreicher Authentifizierung öffnet sich die erste Tür der Schleuse. Ein Sensor registriert den Zutritt, die erste Tür wird geschlossen und die zweite Tür geöffnet. Der Sensor registriert das Verlassen des Mitarbeiters und die zweite Tür wird geschlossen.

S06 | Ladestation

In einer geeigneten Datenstruktur werden 500.000 Einheiten Energie verwaltet. Die Ladestation hat einen 2-poligen Anschluss. Der Benutzer charakterisiert durch Name verfügt über eine Guthaben-/Treuekarte mit 5000 Euro. Eine Einheit Energie kostet 0,35 Euro. **Strategy** | Die Informationen auf der Guthaben-/Treuekarte sind mit SHA256 oder AES verschlüsselt. **State** | Bei der Guthaben-/Treuekarte werden vier Status Blau, Silber, Gold und Platin unterschieden. Ein ggf. möglicher Statuswechsel wir nach dem Tankvorgang geprüft/vollzogen. Für jede Einheit Energie erhält der Benutzer einen Treuepunkt gutgeschrieben. Ab 500 Treuepunkten wechselt der Status von Blau nach Silber und es werden zusätzlich bei jedem Ladevorgang 150 Treuepunkte gutgeschrieben. Ab 2000 Treuepunkten wechselt der Status von Silber nach Gold und bei jedem Ladevorgang wird die doppelte Anzahl an Treuepunkten gutgeschrieben. Ab 10000 Treuepunkten wechselt der Status von Gold nach Platin. Der erste nach Statuswechsel von Gold nach Platin durchgeführte Tankvorgang ist kostenfrei und es wird die einfache Anzahl an Treuepunkten gutgeschrieben. Bei nachfolgenden Tankvorgängen wird die doppelte Anzahl an Treuepunkten gutgeschrieben. Jeder fünfte nach Statuswechsel von Gold zu Platin durchgeführte Tankvorgang ist kostenfrei, jedoch werden in diesem Fall keine Punkte gutgeschrieben.

S07 | Autonome Werkstatt

Zu Simulationszwecken werden 5 Fahrzeuge (2x AutoX und 3x Amazon Zoox) in eine priorisierte Warteschlange eingereiht. Die Amazon Zoox werden vorrangig gegenüber den AutoX abgefertigt. **Mediator / Command / Observer / Visitor** | Der gesamte Prozess der Werkstatt wird über eine Zentraleinheit gesteuert. Die Zentraleinheit sendet ein Kommando an das erste Fahrzeug in der Warteschlange und dieses fährt in Werkstatt auf die Hebebühne. Ein Sensor an der Hebebühne registriert, sobald sich das Fahrzeug korrekt auf dieser befindet. Zur linken und rechten Seite befindet je ein Roboter. Jeder Roboter ist für die auf seiner Seite befindlichen Bauteile LED-Scheinwerfer, Blinker, Bremslichter, Bremsen, GPS, Kamera und Lidar zuständig. Auf jeder Seite des Roboters existiert zu jedem Bauteil ein Stapel mit 5 Einheiten. Zu jedem Bauteil verfügt der Roboter über ein dediziertes Prüfverfahren. Mit einer Wahrscheinlichkeit von 5% ist ein Bauteil defekt und wird getauscht. Für die defekten Bauteile existiert eine Ablage, realisiert als Stack.

Wichtige Hinweise für die Bearbeitung

- **Studium** der **Struktur** und **Funktionsweise** der beteiligten **Design Patterns**.
- **Programmiersprache** | **Java 17.0.5 (LTS)**
  https://www.oracle.com/de/java/technologies/downloads/#java17
- **IntelliJ IDEA** Community **2022.3.2**
- Verwendung geeigneter **englische**r Begriffe für **Namen** und **Bezeichnungen**.
- **Modellierung** Klassendiagramm in **Visual Paradigm 17**.
  Lizenzschlüssel | 7PK32-37N77-7W6PB-94CH3-4W596
  https://ap.visual-paradigm.com/duale-hochschule-badenwurttemberg-mosbach
  Bitte
  - achten Sie auf ein geordnetes Gesamtbild.
  - benennen Sie die **Datei** mit **komplexaufgabe_autonomes_fahrzeug.vpp**.
  - benennen Sie das Klassendiagramm mit [task_id], z.B. s04_observer.
  - löschen Sie die *.bak-Dateien von Visual Paradigm.
- **Implementierung** einer technisch einwandfrei lauffähigen Applikation.
  Bitte
  - erstellen Sie ein **IntelliJ-Projekt** im Verzeichnis [autonomes_fahrzeug].
  - erstellen Sie für jede Aufgabe ein **Paket** mit der Bezeichnung [task_id], z.B. s04observer.
  - nutzen Sie die **camelCase-Notation**, um die Lesbarkeit zu vereinfachen.
  - stellen Sie sicher, dass Modellierung und Implementierung harmonieren.
- **Je Aufgabe** sind **aussagekräftige** und **dedizierte Tests in JUnit** zu erstellen.
- **Je Aufgabe** ist ein **aussagekräftiges Test-Szenario auf Basis BDD** zu erstellen.
  https://jgiven.org/userguide/
- **Clean-Up** und **Formatierung** des **Source Code**.
  (Code ► Reformat Code [Optimize imports, Rearrange entries, Cleanup code])
- Erstellung einer **7-Zip-Datei komplexaufgabe_autonomes_fahrzeug.7z** mit den UML-Diagrammen als vpp-Dateien, dem vollständigen IntelliJ-Projekt und der Readme-Datei. Der 7-Zip-Datei ist eine **Readme-Datei beizufügen**, aus dem eindeutig die **Zuordnung** der **Aufgaben zu** den Studierenden/**Matrikelnummern** hervorgeht.
- **Zeitansatz:** 30 Stunden je Aufgabeblock/Studierenden
- **Abgabetermin:** **Upload in Moodle bis** spätestens Sonntag, **05.03.2023**.
- **Bewertung:** **30 Punkte / Aufgabenblock / Studierenden**