UNIVERSITY OF APPLIED SCIENCES
ASCHAFFENBURG

Software Design Specification

Project: SPOTY

Phase: Design

Authors:
Mehmet Ali Özcevik
Isam Hamid
Velat Perver Irmak
Zübeyir Eser
Mohamed Elsobky

Documentname: AB-SPOTY-DS-3

Version: 1.5

Creation date: 17.06.2024

File: https://github.com/MemoAli03/LiDAR-SDS

Modifications – Document Status

Version	Status	Creation Date	Editor	Modifications
1.0	Planung	05.06.2024	Mehmet Ali Özcevik	Ausgangsdokument
1.1	Unter Bearbeitung	06.06.2024	Mehmet Ali Özcevik	Hauptthemen sind absolviert
1.2	Unter Bearbeitung	08.06.2024	Mehmet Ali Özcevik	Alle Themen absolviert
1.3	Unter Bearbeitung	11.06.2024	Mehmet Ali Özcevik	Feedback Verbesserungen
1.4	Unter Bearbeitung	16.06.2024	Mehmet Ali Özcevik	Weitere Feedback Verbesserungen
1.5	Abgabe	17.06.2024	Mehmet Ali Özcevik	-

(Status ::= planned, under construction, presented, accepted)

This document is based on:

A Software Design Specification Template

by Brad Appleton <bradapp@enteract.com>
http://www.enteract.com/~bradapp

TABLE OF CONTENTS

1 Einleitung

1.1 Zielsetzung:

Der Zweck dieses Dokuments besteht darin, das Softwaredesign des Produkts explizit zu beschreiben. Das Dokument basiert auf der Systemanforderungsspezifikation für dieses Projekt [SRS-SPOTY-1]. Somit werden die Informationen und das Verständnis für die Projektimplementierung vermittelt.

1.2 Definitionen, Akronyme und Abkürzungen:

SRS	Software Requirement Specification (Software-Anforderungsspezifikation)
SDS	Software Design Specification (Software-Desgin-Spezifikation)
LiDAR	Light Detection and Ranging ist eine optische Fernerkundungstechnik, bei der Laserlicht für ein dichtes Abtasten der Erdoberfläche verwendet wird
Point Cloud	Eine dreidimensionale Ansammlung von Punkten, die durch LiDAR-Sensoren erstellt wird, indem sie Laserstrahlen aussenden und die reflektierten Signale messen, um die genaue Position von Oberflächen in der Umgebung zu bestimmen
GUI	Grafische Benutzeroberfläche
ML	Maschinelles Lernen
CVAT	Computer Vision Annotation Tool ist ein Open-Source-Tool zur Markierung und Klassifikation von Bild- und Videodaten für maschinelles Lernen und Computer Vision
VSC	Visual Studio Code
SPOTY	Produktname

1.3 Referenzen:

SRS-SPOTY-1	Software-Requirement-Specification Dokument vom Cooperative Object Detection Vehicle Projekt, Version 1.1 (24.04.2024), alle Autoren

1.4 Überblick:

Abschnitt 2 dieses Dokuments gibt einen Überblick über das System. In Abschnitt 3 werden Überlegungen zum Entwurf beschrieben. Die Systemarchitektur wird in Abschnitt 4 erläutert. Der detaillierte Systementwurf mit jeweiligen Systemkomponenten wird in Abschnitt 5 beschrieben.

2 Übersichtsbeschreibung

2.1 Funktionalität:

Die Software ist in der Lage, mit Hilfe von LiDAR-Sensoren detektierten Point Clouds in verschiedene Klassen zu gliedern. So werden im Verkehrsfluss statische Daten erfasst, welche anschließend während der Trainings-Pipeline im ML-Prozess mit vorher gelabelten Informationen verglichen werden. Dadurch kann die Software bewegte Objekte in die folgenden Klassen unterteilen:

- Autos
- Zweiräder (Motorräder, E-Scooter, Fahrräder)
- Trucks (LKWs, Busse)
- Sonstige Fahrzeuge

2.2 Systemdesign:

Das Architekturkonzept basiert auf einer klassischen Drei-Schichten-Architektur, welche sich in eine Anwendungs-, Fachkonzept- und Datenhaltungsschicht einteilt. Eine detaillierte Erläuterung der Architektur erfolgt in Kapitel 4.

3 Entwurfsbetrachtungen

3.1 Annahmen und Abhängigkeiten:

Vor der Nutzung der Software ist sicherzustellen, dass Python installiert ist.
Die Software wird im Microsoft Visual Studio Code unter Verwendung der Programmiersprache Python entwickelt und getestet.

3.2 Entwurfsrestriktionen:

Wie in Kapitel 3.1 erläutert, ist es ausreichend, wenn auf dem System die Programmiersprache Python installiert ist.

3.3 Ziele und Richtlinien:

Die Software basiert auf maschinellem Lernen (ML). Ihre Aufgabe besteht in der schnellen und präzisen Klassifikation der Input-Daten sowie der Ausgabe der entsprechenden Output-Daten. Zu diesem Zweck wurde eine Trainings-Pipeline mit einer geeigneten Struktur entwickelt.

3.4 Entwicklungsmethodik:

Folgende Tools wurde in der Softwareentwicklung verwendet:

- CVAT
    - Labeln der Rohdaten in die oben erwähnten Klassen
- Microsoft VSC
    - Entwicklung des Codes für das ML-Prozess

4 Systemarchitektur

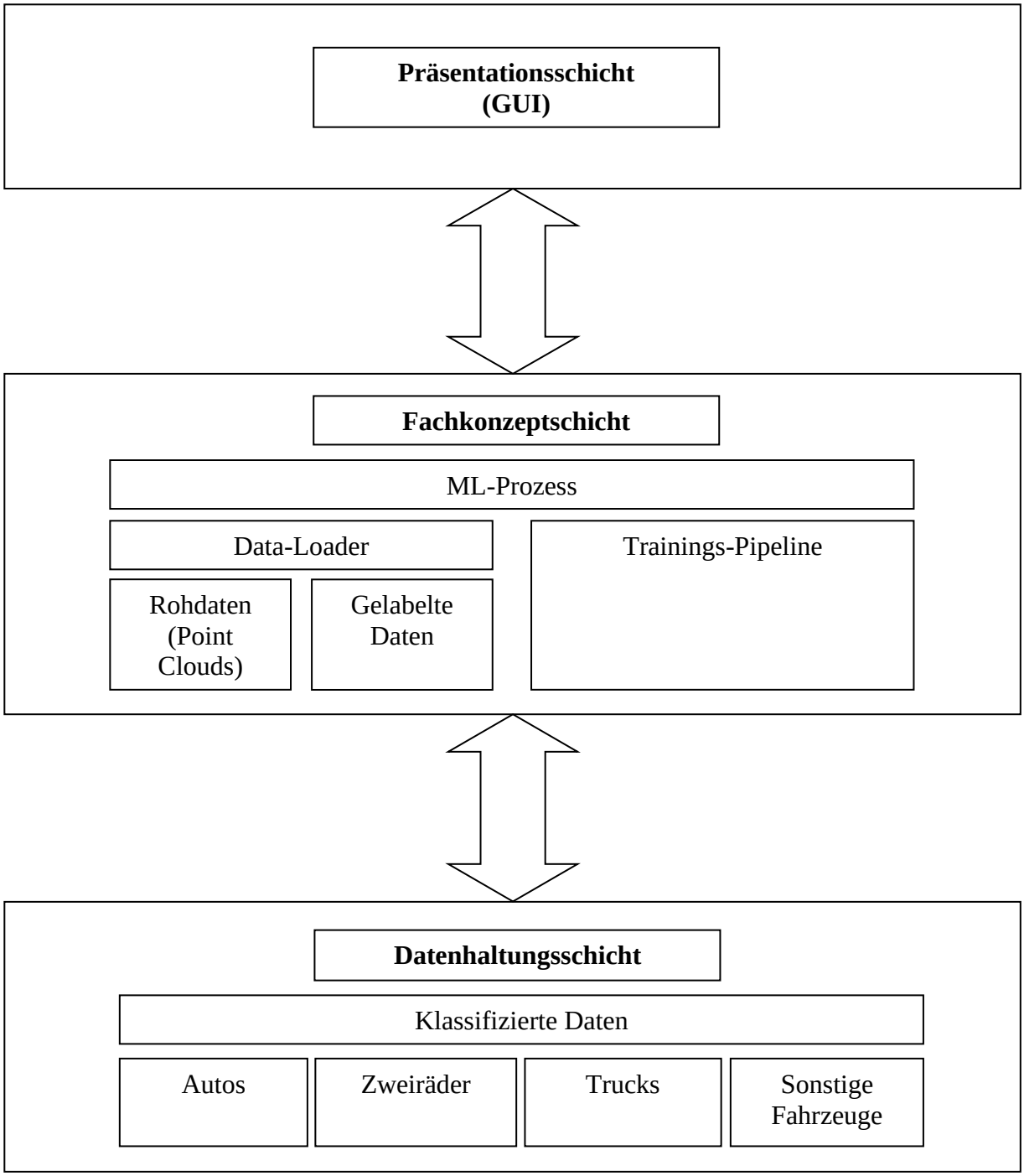


Bild 1 stellt die Systemarchitektur grafisch dar. Wie bereits erwähnt, handelt es sich hierbei um eine klassische Drei-Schichten-Architektur die jeweils aus einer Anwendungs-, Fachkonzept- und Datenhaltungsschicht besteht. Des Weiteren kann man in der Grafik erkennen, dass die einzelnen Schichten ebenfalls in kleine Subkomponenten unterteilt sind, welche zusammen eine Gesamtunterteilung ergeben.

4.1 Beschreibung der Subkomponenten:

4.1.1 Anwendungsschicht (GUI):

- Die Funktionalität wird von Microsoft-VSC implementiert und realisiert

4.1.2 Fachkonzeptschicht:

- **ML-Prozess:**
  - **Data-Loader:** dieser ist dafür zuständig die Rohdaten als Point-Clouds und die von uns vorgelabelten Daten in das Programm zu laden
  - **Trainings-Pipeline:** die reingeladenen Daten werden mit einem Trainingsalgorithmus in die vorgegeben Klassen unterteilt

4.1.3 Datenhaltungsschicht:

- Die in der Trainings-Pipeline sortierten Daten, werden nun in den vier Klassen (Autos, Zweiräder, Trucks und sonstige Fahrzeuge) gespeichert und als Output-Daten festgehalten

5 Systementwurf

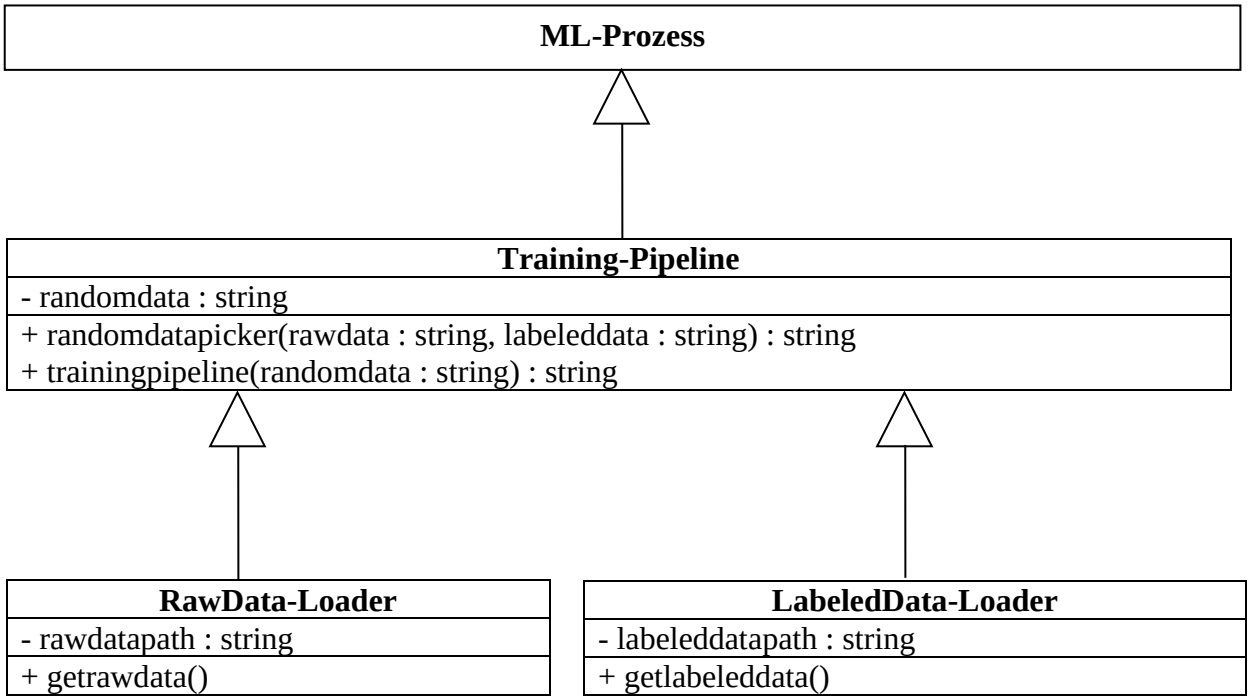
5.1 Präsentationsschicht:

Wie bereits erwähnt, wird die Funktionalität von Microsoft-VSC implementiert und realisiert.

5.2 Fachkonzeptschicht:

5.2.1 Klassendiagramm der Fachkonzeptschicht:

Die folgende Grafik stellt das Klassendiagramm der Fachkonzeptschicht des Spotys dar:



Das Klassendiagramm der Fachkonzeptschicht beschreibt den ML-Prozess des Produkts. Es empfiehlt sich, die Betrachtung mit den unteren Kindsklassen zu beginnen. Die Klasse „RawData-Loader“ ist dafür zuständig, im Dateipfad liegende Rohdaten in das Programm hineinzuladen. Der „LabeledData-Loader“ arbeitet analog dazu, lädt stattdessen aber die von uns gelabelten Daten hinein. Im Anschluss werden beide Datenarten in der Methode „randomdatapicker“ zufällig für die anstehende Training-Pipeline ausgewählt. Dies hat den Vorteil, dass im Trainingsprozess unterschiedliche Daten bearbeitet werden, wodurch die Qualität und Richtigkeit der klassifizierten Output-Daten erhöht wird. Andernfalls würde das Programm immer dieselbe Datenreihenfolge im Trainingsprozess bearbeiten, was bei der realen Nutzung, also bei unvorhersehbaren Input-Daten, zu Komplikationen führen kann. Wie auch bereits angedeutet, ist die Methode „trainingpipeline“ für das Trainieren der Daten zuständig. Alle diese Kindsklassen sind der Elternklasse ML-Prozess zugeordnet.

5.3 Datenhaltungsschicht:

5.3.1 Klassendiagramm der Datenhaltungsschicht:

Die folgende Grafik stellt das Klassendiagramm der Fachkonzeptschicht des Spotys dar:

Print-Output-Data
- car : string - bicycle : string - truck : string - other vehicle : string
+ print(output : string) : string

In dieser Klasse erfolgt die Ausgabe der in Klassen gegliederten Daten. Die Attributnamen repräsentieren die einzelnen Klassen, welche die folgenden Kategorien umfassen: „Autos", „Zweiräder", „Trucks" und „andere Fahrzeuge". Die Ausgabe der Daten erfolgt mit Hilfe der print-Methode.