

Modul: Business Process Automation

Anwendungsdokumentation Thema:
Visuelle Inspektion in der Produktion mit Hilfe von Computer Vision

Bearbeiter:
Jakob Laufer, Tom Marinovic, Konrad Adamski, Tom Bischopink, Fabian Kuhne, Jessica Knick

Abgabe: 27.06.2024
Betreuender Professor: Prof. Dr. Reichelt

Inhaltsverzeichnis

Abbildungsverzeichnis	3
1 Architekturkonzept.....	4
1.1 Beschreibung Architektur.....	4
1.2 Schnittstellen	5
1.2.1 MQTT.....	5
1.2.2 HTTP	5
1.3 Sequenzdiagramm.....	6
1.4 Geschäftsprozessmodel	7
2 Auto-ID.....	8
3 AAS	9
3.1 Entscheidung	9
3.2 Aufbau der AAS Shell	11
3.3 Prüfplan	12
4 Entwickelte Artefakte	17
4.1 Architektur	17
4.2 Sequenzdiagramme.....	18
5 Einrichtung und Inbetriebnahme	20
5.1 Anleitung Inbetriebnahme.....	20
5.2 Anleitung Erweiterung.....	20
5.3 Hinzufügen neuer Shells	20
6 Eigenständigkeitserklärung.....	22

Abbildungsverzeichnis

Abbildung 1 Architekturkonzept	4
Abbildung 2 Sequenzdiagramm Entwurf Komponenten Kommunikation	6
Abbildung 3 BPMN-Geschäftsprozess	7
Abbildung 4 Gewählter ECLASS Standard	10
Abbildung 5 Beispiel Aufbau AAS	11
Abbildung 6 Prüfplan im JSON Format	13
Abbildung 7 Schema der Antwort	14
Abbildung 8 Antwort der optischen Prüfeinheit	15
Abbildung 9 Klassendiagramm der entwickelten Artefakte	17
Abbildung 10 Sequenzdiagramm Verbindungsaufbau	18
Abbildung 11 Sequenzdiagramm Implementierte Komponenten Kommunikation	19
Abbildung 12 Beheben von doppeltem ID Fehler	21

1 Architekturkonzept

1.1 Beschreibung Architektur

Es gilt, ein Konzept zur Integration der I4.0-Verwaltungsschale mit Industrial Computer Vision Lösungen zu untersuchen. Im ersten Schritt wurde die hierfür notwendige Struktur geplant. Die folgende Grafik zeigt die gewählte Architektur des Projektes:

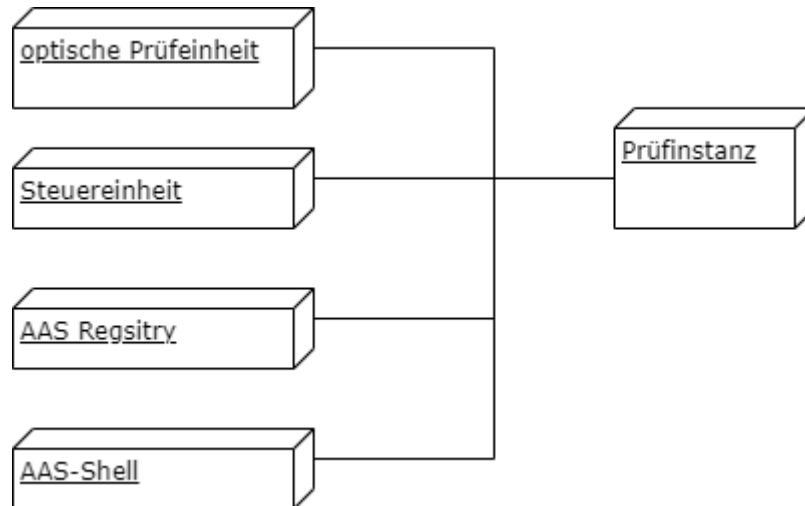


Abbildung 1 Architekturkonzept

Die Prüfinstanz, welche es im Zuge des Projektes zu entwickeln galt, agiert als Middleware zwischen der Steuereinheit, der optischen Prüfeinheit, der AAS-Registry und den Asset Administration Shells.

Die Steuereinheit ist verantwortlich, die Prüfinstanz über das Eintreffen eines neuen Fahrzeugs auf dem Prüfplatz zu informieren, sowie den RFID-Chip des Autos auszulesen. Die Kommunikation über die Steuereinheit ist als OPC-UA-Server umgesetzt. Das Eintreffen eines neuen Autos wird durch Setzen einer Variablen „start_scan“ oder das Drücken des Knopfes am OPC-UA-Server simuliert. Wenn eine Änderung der Variablen auf true registriert wird, startet der RFID-Reader und liest den Chip aus.

Die optische Prüfeinheit hat die Aufgabe das Auto auf dem Prüfplatz zu prüfen und das Prüfergebnis an die Prüfinstanz zu melden. Die Kommunikation mit der optischen Prüfeinheit wird über MQTT abgebildet.

Die Registry wird initial bei jedem Auto mit einem http GET-Request angesprochen. Die ID des Autos aus dem RFID-Chip stimmt mit der ID der jeweiligen Shell überein. Mithilfe der Registry wird die, für das aktuelle Auto, passende Shell identifiziert. Mit dieser identifizierten Shell läuft dann die weitere Kommunikation ab.

Die Prüfinstanz holt sich wieder über einen http GET-Request den Prüfplan des Autos aus der Shell und lädt nach abgeschlossener Prüfung das Prüfergebnis als JSON-Datei wieder in die Shell.

1.2 Schnittstellen

Die Kommunikation der entwickelten Prüfinstanz mit der Steuereinheit, der optischen Prüfeinheit, der AAS-Registry und den AAS-Shells erfolgt entweder über OPC-UA, MQTT oder HTTP.

1.2.1 MQTT

Der MQTT-Broker ist unter der IP-Adresse 141.56.180.177 mit dem Port 1883 erreichbar. Das Grundtopic ist bpa24. Die Topics für die Kamera sind für den Request: bpa24/cv/request und die Response: bpa24/cv/result.

1.2.2 HTTP

Wir verwenden die Open Source Plattform BaSyxs für das Konzept der Asset Administration Shells als digitalen Zwilling der Autos. BaSyxs hostet die AAS-Registry und AAS-Shells auf einem Tomcat-Webserver. In unserem Aufbau wird die Plattform BaSyxs in einer Linux VM (IP: 141.56.180.118) in einem Docker Container gehostet.

Durch diesen Aufbau können die Registry und die Shells mittels HTTP-Requests erreicht werden und die Daten, wie den Prüfplan, die Shell-ID und den Port abfragen. Die AAS-Registry ist unter dem Port 8080 und die 3 Shells über jeweils 8081, 8082 und 8083.

1.3 Sequenzdiagramm

Zur Visualisierung der Kommunikation wurde ein Sequenzdiagramm erstellt. Diese visualisiert den umgesetzten Kommunikationsablauf in der Lösung.

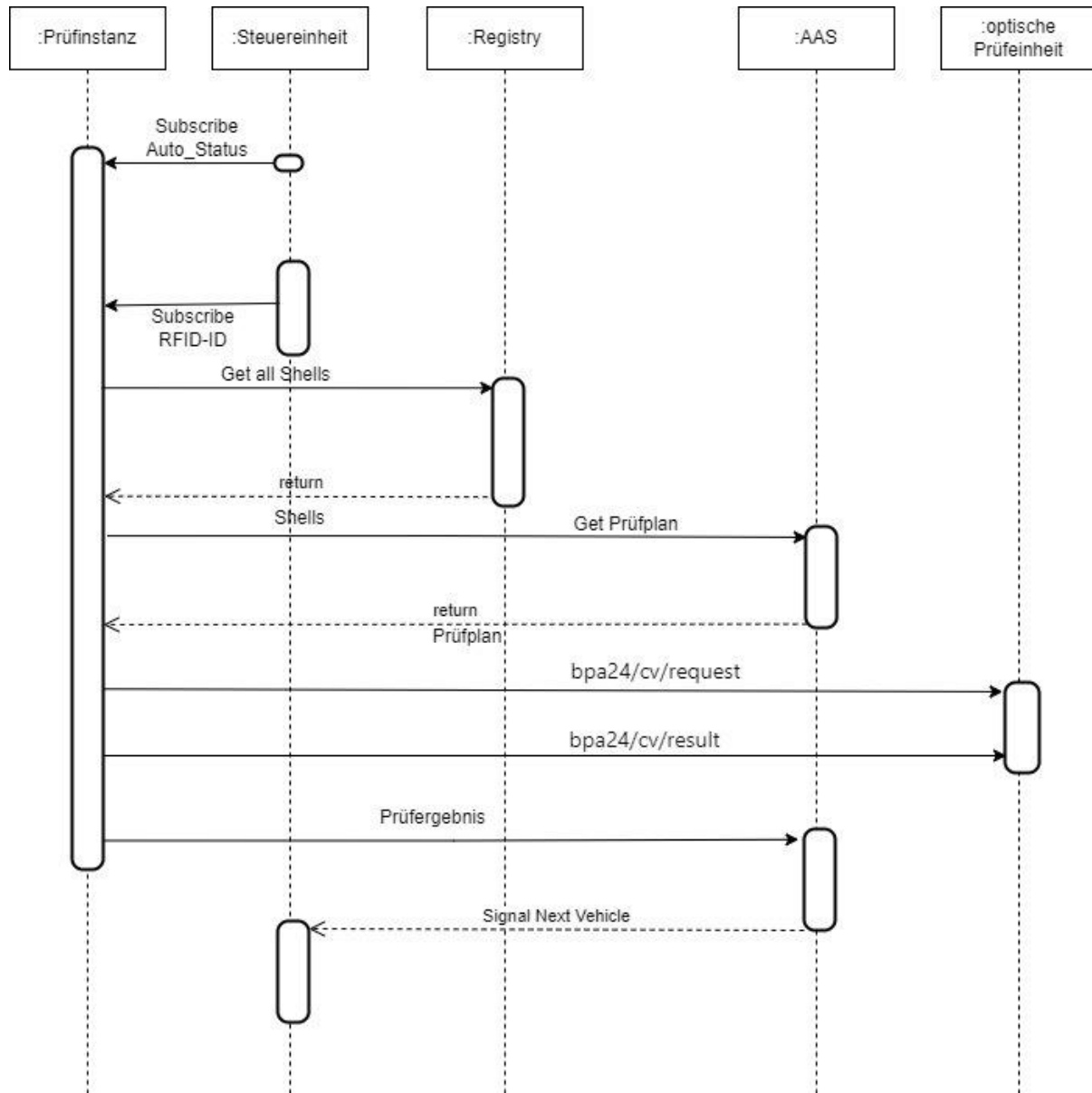


Abbildung 2 Sequenzdiagramm Entwurf Komponenten Kommunikation

1.4 Geschäftsprozessmodell

Zum besseren Verständnis der Kommunikation und Abläufe zwischen den Komponenten wurde ein Geschäftsprozess in BPMN-Notation erstellt. Zusätzlich konnte so die zu entwickelnde Prüfinstanz geplant werden.

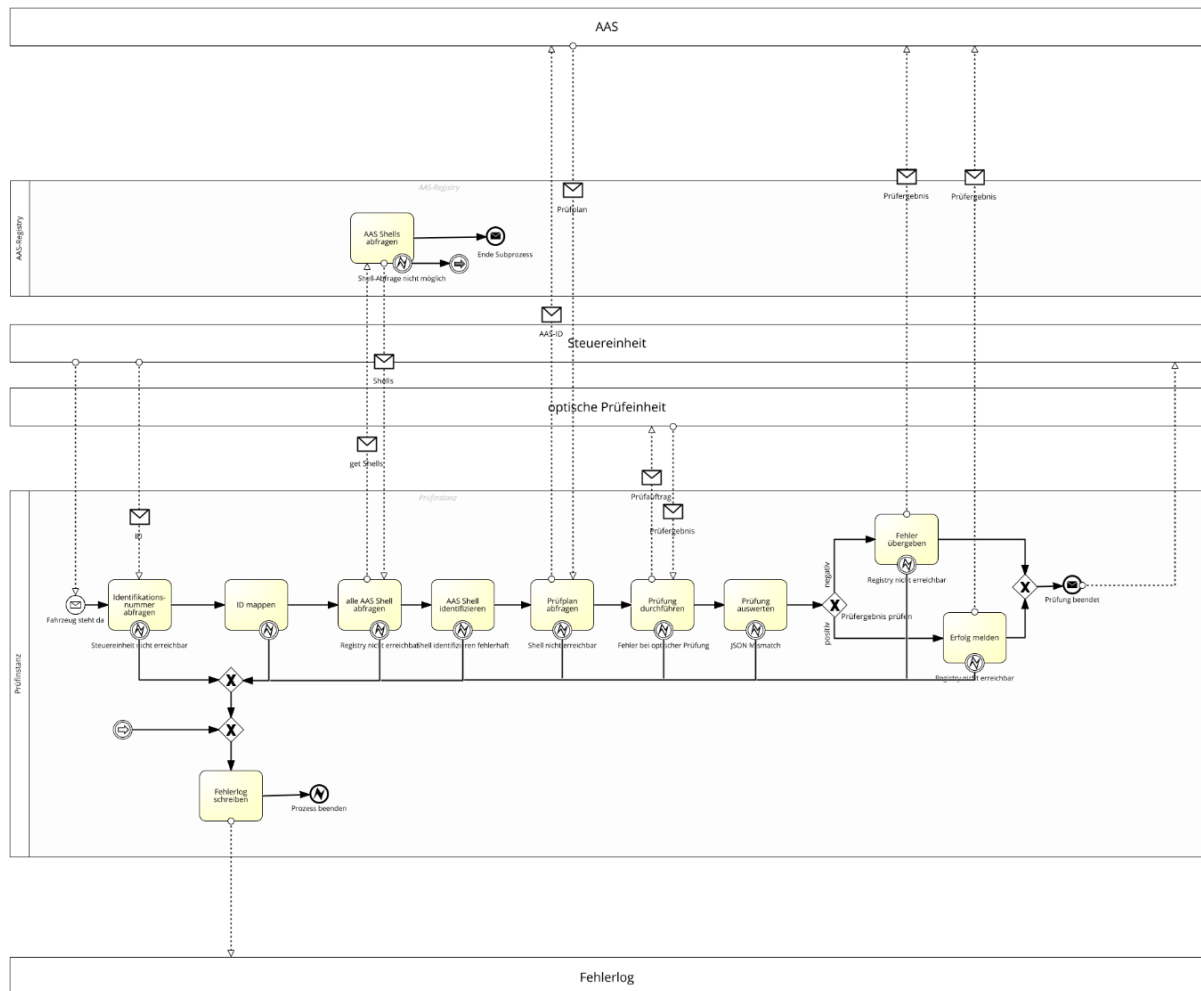


Abbildung 3 BPMN-Geschäftsprozess

Eine größere Darstellung des Geschäftsprozesses befindet sich im Anhang.

2 Auto-ID

Auto-ID steht für **Automatische Identifikation und Datenerfassung**. Das bedeutet für uns, dass definierte Objekte mit einer spezifischen Codierung versehen werden, die von einem Gerät automatisiert ausgelesen werden können. In unserem konkreten Anwendungsfall ist das Objekt ein Modellfahrzeug.

In der Praxis konnten sich verschiedene Möglichkeiten zur Identifikation und Datenerfassung etablieren:

Mobile Daten

Die konkrete Umsetzung von mobilen Daten wäre im Kontext unseres Projektes unklar.

Barcodes

Barcodes ermöglichen eine optische Kontrolle. Diese Methode ist kostengünstig, erfordert jedoch die Berücksichtigung eines speziellen Prozesses zur Barcode-Zuordnung.

QR-Code / Datamatrix

QR-Codes und Datamatrix-Codes ermöglichen ebenfalls eine optische Kontrolle. Diese Codes setzen jedoch Sichtkontakt zum Objekt voraus, wobei die Entfernung zu beachten ist. Zudem muss auch hier ein spezieller Prozess zur QR-Code-Zuordnung berücksichtigt werden.

RFID

RFID-Tags ermöglichen die Identifikation auch ohne Sichtkontakt. Zudem gibt es HF-Tags, die für den Nahbereich gut geeignet sind. Der Tag müsste nicht am Modellfahrzeug befestigt werden (da die Befestigung auf Metall ungünstig ist), sondern könnte an Seitenträgern montiert werden. Ein Scanner könnte dabei an der Seite befestigt werden.

Daher wurde RFID als Verfahren zur automatischen Identifizierung von Objekten über Funk im Kontext unseren Anwendungsfalls als die beste Alternative eingeschätzt.

3 AAS

3.1 Entscheidung

Eine AAS soll Eigenschaften wie Abstraktion, Identifizierbarkeit und Kommunikationsfähigkeit ermöglichen. Das geschieht beispielsweise durch die Trennung zwischen Typ und Instanz eines Gegenstandes oder Prozesses. Während der Typ mit einer Vorlage verglichen werden kann, bildet die Instanz eine konkrete Ausprägung ab.

Bezogen auf den Produktionsprozess eines Fahrzeugs, wurden folgende Teilgebiete innerhalb der Produktionslinie als relevant eingestuft:

- Allgemeine Informationen zum Fahrzeug
- Komponenten eines Fahrzeugs
- Produktionslinie
- Prüfplan

Zudem haben in den Teilgebieten detaillierte Informationen ausgearbeitet.

- **Allgemeine Informationen**
 - Herstellerartikelnummer: 12345
 - Marke: BMW / Rolls Royce / Mini
 - Herstellerproduktbezeichnung: Auto / Motorrad
 - Herstellerprodukttyp: M4 Coupe / ...
 - Global Location Number des Herstellers: 1234567898765
 - Global Trade Item Number: 9876543212345
- **Fahrzeugkomponenten**
 - Karosserie (Größe in mm): 4671 x 1870 x 1383
 - Motor: 6 (Zylinder)
 - Türen: 2 / 4
 - Spiegel: 2
 - Scheinwerfer: Xenon / ...
 - Reifentyp (Zoll): 18"/19" oder 19"/20"
 - Lackierungsfarbe: Mineralweiß Metallic / M Portimao Blau Metallic
 - Cabrio-Verdeck: Ja/Nein
 - Logo: Ja/Nein
 - Schild: Ja/Nein
 - Felge: 20'

- **Produktionslinie**
 - Fertigungsschritt 1: Presswerk
 - Zeitstempel: 2021-01-01 00:00:00 +0000
 - Fertigungsschritt 2: Karosseriebau
 - Zeitstempel: 2021-01-01 00:00:00 +0000
 - Fertigungsschritt 3: Lackierung
 - Zeitstempel: 2021-01-01 00:00:00 +0000
 - Fertigungsschritt 4: Montage
 - Zeitstempel: 2021-01-01 00:00:00 +0000
 - Fertigungsschritt 5: Endmontage
 - Zeitstempel: 2021-01-01 00:00:00 +0000
 - Fertigungsschritt 6: Optische Prüfung
 - Zeitstempel_Beginn: 2021-01-01 00:00:00 +0000
- **Prüfplan**
 - Vgl. Kapitel 3.3

In der Praxis bieten sogenannte AAS-Templates standardisierte Vorlagen zur Verwaltung von Informationen über physische Assets. Beispielsweise stellt der Verein ECLASS e.V. Standards in der Form von Templates zur erleichterten Implementierung und Nutzung der Verwaltungsschalen bereit.

Wir haben uns anschließend für die Arbeit mit den AAS-Templates entschieden, weil sie sowohl für unser Projekt als auch für potenzielle weiterführende Anbindungen zahlreiche Vorteile bieten würde. Die standardisierten Vorlagen ermöglichen eine schnellere und effizientere Verwaltung von Informationen, da sie bewährte Strukturen und Prozesse vorgeben. Weiterhin wird auch eine hohe Konsistenz sichergestellt, da Informationen einheitlich und strukturiert erfasst und verwaltet werden können. Weiterführend wird durch die Einhaltung dieser etablierten Standards die Qualität der Daten und der Prozesse verbessert, was zu einer höheren Zuverlässigkeit und Genauigkeit führt.

Daher haben wir den ECLASS Content durchsucht und für unsere Teilgebiete nach passenden AAS-Templates gesucht. Für unser Teilgebiet “Allgemeine Informationen” entschieden wir uns für [28-01-01-01 PKW, geschlossen](#), da der Standard alle detaillierten Informationen enthielt.

≡

ECLASS BASIC 14.0 (de)

⊟

28 Fahrzeug (Gesamtfahrzeug)

⊟

28-01 Kraftfahrzeug ⓘ

⊟

28-01-01 Personenkraftwagen (PKW) ⓘ

⊟

28-01-01-01 PKW, geschlossen

📄

Klassifikation: 28010101 [AGN838005]

Bevorzugte Benennung	28-01-01-01 PKW, geschlossen
Definition	Personenkraftwagen (abgekürzt Pkw oder PKW), in der Schweiz Personenwagen (PW), sind mehrspurige Fahrzeuge mit eigenem Antrieb zum vorwiegenden Zwecke der Personenbeförderung. Es sind Kraftfahrzeuge, die nach ihrer Bauart und Ausstattung zur Beförderung von nicht mehr als neun Personen (inkl. Fahrzeugführer) geeignet und bestimmt sind. Im Alltag werden sie auch Auto (kurz für Automobil) bzw. technisch Kraftwagen genannt
IRDI	0173-1#01-AGN838#005

🔽 AUF MERKLISTE

Abbildung 4 Gewählter ECLASS Standard

3.2 Aufbau der AAS Shell

Der Aufbau unserer AAS enthält folgende Informationen und ist wie folgt strukturiert:

- Im **Package** sind alle Daten und Metadaten organisiert, die im **Environment** verwaltet werden
- **Administration Shells** dienen als zentrale digitale Repräsentationen der Modellfahrzeuge und umfassen verschiedene Submodels
- **Submodels** sind spezialisierte Teile der Administration Shell, die spezifische Aspekte oder Eigenschaften der Modellfahrzeuge detailliert beschreiben

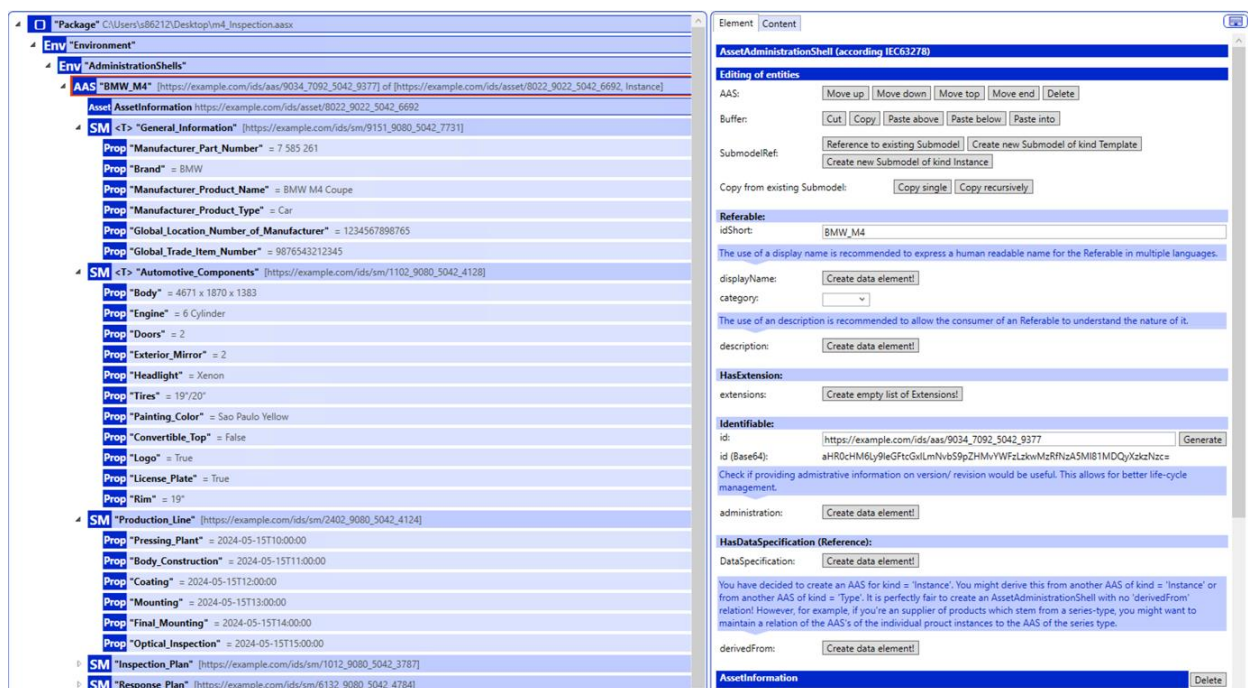
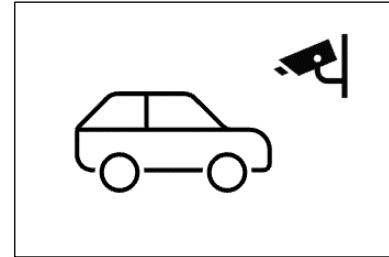


Abbildung 5 Beispiel Aufbau AAS

3.3 Prüfplan

In dem von uns bearbeiteten Szenario ist die Kamera der Prüfstation oberhalb vor dem Auto montiert. Auf Basis dieser vorläufigen Kameraposition wurde der Prüfplan aufgebaut. Es wurden aus allen Elementen, welche die optische Prüfeinheit prüfen kann, jene ausgewählt, die von einer Kamera an besagter Position erkannt werden können. Der Prüfplan wurde als JSON-Datei erstellt, da die Response der optischen Prüfeinheit im JSON-Format kommt. Weiterhin kann durch ein einheitliches Format ein leichteres Zuordnen der Daten gewährleistet werden kann.



Im Prüfplan werden momentan die Elemente abgebildet, welche von der Kameraperspektive aus erfasst werden können. Das betrifft folgende Elemente des Autos:

- Frontscheibe (vorhanden)
- Spiegel
- Markenzeichen/Logo
- Kühlergrill
- Scheinwerfer
- Dach
- Motorhaube
- Scheibenwischer
- Frontschürze

Momentan kann die Prüfeinheit nur die generelle Existenz der Elemente überprüfen und nicht deren Zustand (Schäden). Um diese Werte in dem Prüfplan angeben zu können, werden Werte für die Prüfeinheit zum Test per Zufall generiert. Diese künstlich erzeugten Werte geben an, ob Bauteile wie die Karosserie, die Spiegel und die Scheinwerfer frei von Schäden sind.

JSON-Datei des Prüfplans:

```
{
  "Inspection_Plan": {
    "Car_Roof": {
      "Car_Roof_in_place": [true, false],
      "Car_Roof_free_of_damage": [true, false]
    },
    "Engine_Hood": {
      "Engine_Hood_in_place": [true, false],
      "Engine_Hood_is_free_of_damage": [true, false]
    },
    "Front_Window": {
      "Front_Window_in_place": [true, false],
      "Front_Window_is_free_of_damage": [true, false]
    },
    "Windshield_Wiper": {
      "Windshield_Wiper_in_place": [true, false],
      "Windshield_Wiper_is_free_of_damage": [true, false]
    },
    "Mirrors": {
      "Mirrors_in_place": [true, false],
      "Mirrors_free_of_damage": [true, false]
    },
    "Headlights": {
      "Headlights_in_place": [true, false],
      "Headlights_free_of_damage": [true, false]
    },
    "Radiator_Grill": {
      "Radiator_Grill_in_place": [true, false],
      "Radiator_Grill_is_free_of_damage": [true, false]
    },
    "Front_Bumper": {
      "Front_Bumper_in_place": [true, false],
      "Front_Bumper_is_free_of_damage": [true, false]
    },
    "Brand_Badge": {
      "Brand_Badge_in_place": [true, false],
      "Brand_Badge_is_free_of_damage": [true, false]
    }
  }
}
```

Abbildung 6 Prüfplan im JSON Format

Schema der Antwort:

```
{
  "response": {
    "classes": {
      "0": "partA",
      "1": "partB",
      "...": "...",
      "N": "partN"
    },
    "detections": {
      [
        {
          "confidence",
          "free_of_damage",
          "class_id"
        }
      ]
    }
  }
}
```

Abbildung 7 Schema der Antwort

Erklärung der Antwort:

Die Paare, wie "0" : "partA" sind als Dictionary zu verstehen. In der Antwort der Prüfinstanz steht die Zahl "0" stellvertretend für das Bauteil "partA". Unter dem Punkt detections wird diesem Bauteil eine "confidence" zugeordnet. Dieser Wert gibt an, wie sicher sich das System ist das Teil richtig erkannt zu haben. Ab einem Wert größer als 0.6 gilt das Bauteil als korrekt erkannt. Der Wert "free_of_damage" ist ein Boolean Wert, welcher Momentan noch simuliert wird. Darin gibt das System an, ob das Bauteil Frei von Schäden ist oder nicht. Der letzte Punkt "class_id" enthält die Id von dem Bauteil. In unserem Beispiel handelt es sich um "partA". Dementsprechend ist die "class_id": "0".

Konkret sieht die Antwort wie folgt aus:

```
{
  "classes": [
    "0": "mirror",
    "1": "a_pillar",
    "2": "window_side_front",
    "3": "cooler",
    "4": "front_door",
    "5": "wheel_rear",
    "6": "front_bumper",
    "7": "roof",
    "8": "window_front",
    "9": "front_light",
    "10": "wipper",
    "11": "hood",
    "12": "window_side_rear",
    "13": "licence_plate",
    "14": "window_roof",
    "15": "brand",
    "16": "wheel_front",
    "17": "back_body",
    "18": "front_body",
    "19": "back_door",
    "20": "door_handle",
    "21": "c_pillar",
    "22": "back_bumper",
    "23": "back_light",
    "24": "trunc",
    "25": "window_back",
    "26": "exhaust"
  ],
  "detections": [
    [
      9.6826e-01,
      true,
      1.8000e+01
    ],
    [
      9.6729e-01,
      false,
      1.6000e+01
    ],
    [
      9.6680e-01,
      true,
      9.0000e+00
    ]
  ]
}
```

Abbildung 8 Antwort der optischen Prüfeinheit

Ausblick:

Mit der Weiterentwicklung der Prüfinstanz und des Kamerasystems soll es in Zukunft möglich sein, weitere Bauteile in die Prüfelemente aufzunehmen. Konkret könnten Kameras seitlich und hinter dem Auto dabei helfen die Türgriffe, die Seitentüren oder auch den Kofferraum optisch zu erkennen. Ziel muss es sein alle Elemente des Autos optisch zu erfassen. Weiterhin ist wünschenswert, wenn das Kamerasystem in Zukunft in der Lage ist reale Beschädigungen am Auto zu erkennen. Abschließend ist es auch wünschenswert die Farbe des Autos durch das Kamerasystem überprüfen zu können.

4 Entwickelte Artefakte

Für dieses Projekt wurde eine code generierte Dokumentation mithilfe des Sphinx Framework gewählt. Diese Dokumentation ist im GitLab des Projektes ([BPA-ss2024](#)) als PDF zu finden.

Es wurde diese Art der Dokumentation des Codes gewählt, da dies ein modernes Vorgehen einer Codedokumentation eines Projektes ist, der viele Vorteile bietet. Dieses Vorgehen ist automatisiert, professionell, einfach und anpassbar an verschiedene Release-Versionen.

4.1 Architektur

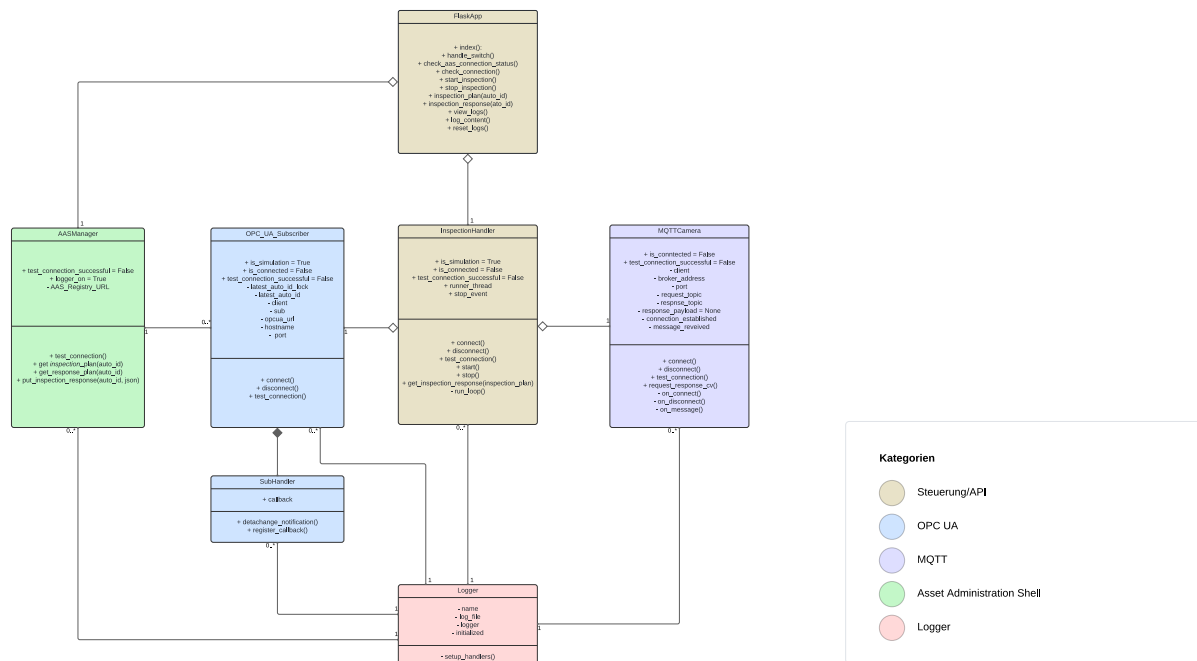


Abbildung 9 Klassendiagramm der entwickelten Artefakte

Mithilfe von objektorientierten Ansätzen wurde die Code-Base so entwickelt, dass es modular anpassbar ist. Damit kann eine einfache Erweiterbarkeit der existierenden Funktionalitäten der bisherigen Klassen und Funktionen realisiert werden.

Die Flask-App (UI für den Nutzer) kann als zentrale Steuerung aller Kommunikationsmethoden für das Projekt dienen, wobei die Klassen **AASManager**, **OPC_UA_Subscriber**, **InspectionHandler** und **MQTTCamera** getrennt aufgerufen und ausgeführt werden können. Über die Umgebungsvariablen in der **.env** Datei können die MQTT, OPCUA und AAS-URLs der Server definiert werden.

Die Klasse **SingletonLogger** ist verantwortlich für eine einheitliche und standardisierte Protokollierung aller Events, die zwischen den Klassen, also der Kommunikation mit AAS, OPCUA und MQTT, stattfinden. Diese Events werden in einer **app.log** Datei gespeichert und in der Flask-App für den Nutzer ausgegeben. Dabei wurde das sog. Singleton-Designmuster angewendet, welches dafür sorgt, dass die Klasse nur einmal instanziiert und anschließend referenziert wird. Damit wird sichergestellt, dass alle Log-Events in eine **app.log** Datei geschrieben werden. Ansonsten würde jede Klasse ihre eigene app.log Datei haben.

Die Anleitung der Inbetriebnahme der App ist in der README zu finden.

4.2 Sequenzdiagramme

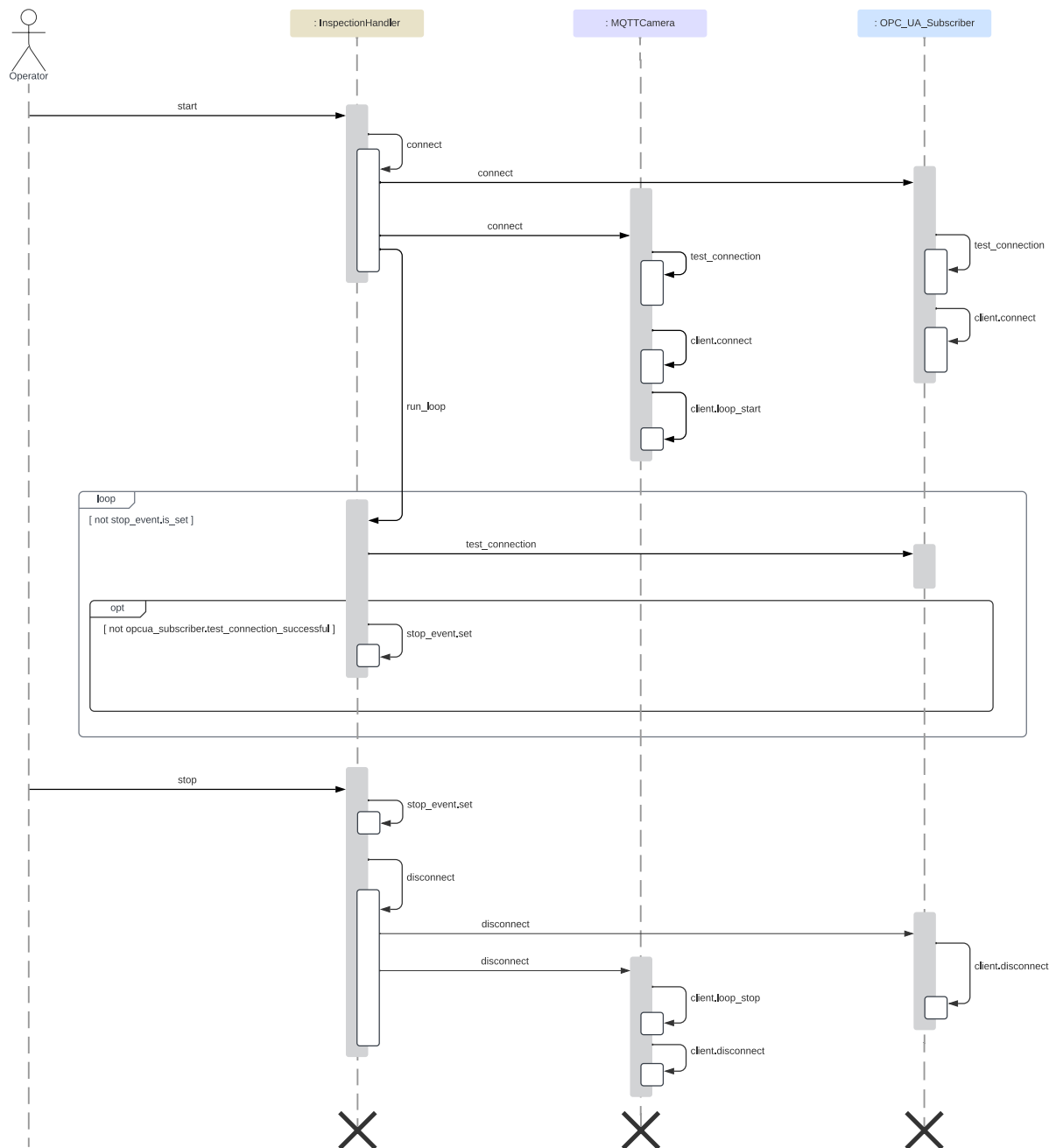


Abbildung 10 Sequenzdiagramm Verbindungsaufbau

In diesem Sequenzdiagramm wird der Verbindungsaufbau und das Beenden der Inspektionsanwendung dargestellt.

Beim Verbindungsaufbau finden zunächst Verbindungsüberprüfungen statt. Beim Erfolg startet eine Dauerschleife, die die OPCU-Verbindung kontinuierlich überprüft. Da über OPC UA neue Fahrzeuguntersuchungen ausgelöst werden, ist diese Verbindung für die Anwendung essenziell.

Beim Verbindungsabbau wird die Dauerschleife beendet und jede Verbindung sauber getrennt.

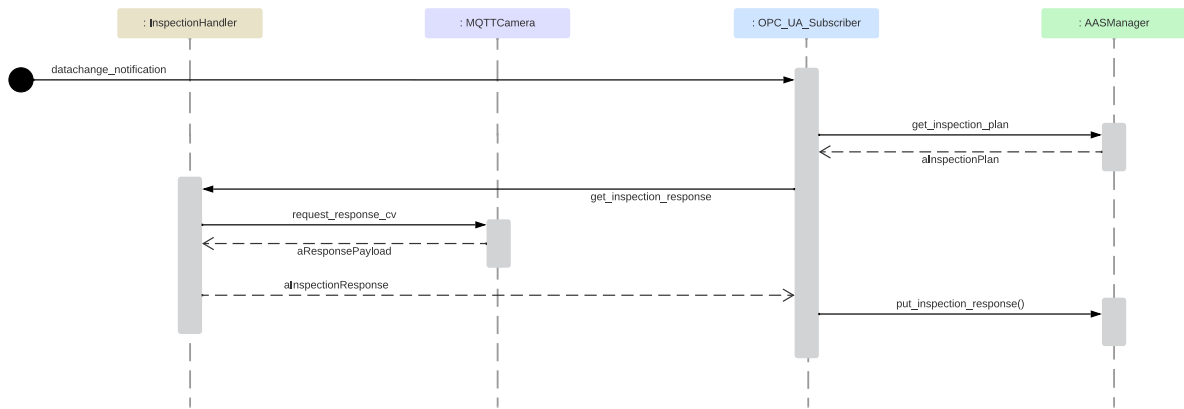


Abbildung 11 Sequenzdiagramm Implementierte Komponenten Kommunikation

Das Sequenzdiagramm zeigt, wie durch eine Datenänderungsbenachrichtigung in OPC UA der Prozess der Verarbeitung des Inspektionsplans aus der AAS und die Antwort der Kameraeinheit aus MQTT zu einer Inspektionsantwort für die AAS führen.

5 Einrichtung und Inbetriebnahme

5.1 Anleitung Inbetriebnahme

Schritte: vm/container/basyxs starten anschließend AAS starten:

`docker compose up -d registry` - Befehl startet basyxs registry

`docker compose up -d 'AAS'` - z.B. `docker compose up -d X7`

5.2 Anleitung Erweiterung

Schritte zum Hinzufügen einer neuen Shell

Hochladen der AAS Shell auf den Server

Hinzufügen der AAS Shell zum Dockercompose File mit eigenem Port

Erstellen und Konfigurieren einer neuen .properties Datei mit eigenem Port, entsprechend den Bestandsshells.

Konfiguration und Test der Steuereinheit und des RFID-Readers erfolgten auf dem Testserver, der über dieselbe IP-Adresse verfügt wie der Simulator.

5.3 Hinzufügen neuer Shells


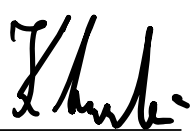


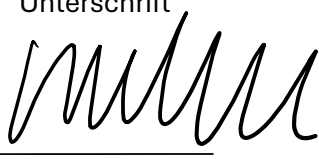
Wenn für das Erstellen einer neuen Shell eine bestehende kopiert und abgeändert wurde, führt das zu folgendem Fehler:

```
Loading AAS Environment (0/1) from file 'mini_cooper_Inspection.aasx'
Mini | 2024-05-29T07:39:27.991Z WARN 1 --- [          main]
ConfigServletWebServerApplicationContext : Exception encountered during context initialization -
cancelling refresh attempt: org.springframework.beans.factory.BeanCreationException: Error
creating bean with name 'preconfigurationLoaderInitializer' defined in URL
[jar:file:/application/basyxExecutable.jar!/BOOT-INF/lib/basyx.aasenvironment-core-2.0.0-
SNAPSHOT.jar!/org/eclipse/digitaltwin/basyx/aasenvironment/preconfiguration/PreconfigurationLo
aderInitializer.class]: Unable to link the element with id
'https://example.com/ids/sm/6132_9080_5042_3784' with the Registry.
```

Der Ursprung des Fehlers ist die Shell-ID, durch das Kopieren bleibt die Shell-ID gleich, das führt zu einem Fehler, da die Shells unique IDs benötigen. Im AASX-Package Explorer kann die ID geändert werden.

6 Eigenständigkeitserklärung

Wir versichern hiermit, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die Angegebenen Quellen und Hilfsmittel benutzen haben. Wir reichen sie erstmals als Prüfungsleistung ein. Uns ist bekannt, dass ein Betrugsversuch mit der Note „nicht ausreichend“ (5.0) geahndet wird und im Wiederholungsfall zum Ausschluss von der Erbringung weiterer Prüfung

27.06.2024	Tom Bishopink	
Datum	Name	Unterschrift
27.06.2024	Konrad Adamski	
Datum	Name	Unterschrift
27.06.2024	Jessica Knick	
Datum	Name	Unterschrift
27.06.2024	Jakob Laufer	
Datum	Name	Unterschrift
27.06.2024	Fabian Kuhne	
Datum	Name	Unterschrift
27.06.2024	Tom Marinovic	
Datum	Name	Unterschrift

