Praktikum Programmierung von Komponentenarchitekturen (I267)
Prof. Dr. Mario Neugebauer
Dipl.-Medieninf. Alexander Wülfing

1. Anforderungen

Es soll ein Flottenmanagementsystem (FMS) für UGVs (Unmanned Guided Vehicles) entwickelt werden. Das erfordert eine Backend-Software, die mehrere Anforderungen erfüllen muss, um einen zuverlässigen Betrieb der UGV-Flotte zu gewährleisten. Im Folgenden werden die Anforderungen an das Flottenmanagement grob beschrieben.

1.1. Fahrzeugverfolgung und -verwaltung

Die Software sollte in der Lage sein, die genaue Position jedes UGVs in Echtzeit zu verfolgen.

Es sollte möglich sein, die Flotte zu verwalten, neue Fahrzeuge hinzuzufügen, defekte Fahrzeuge auszusondern und die Zuweisung von Fahrzeugen zu Aufgaben zu steuern.

1.2. Routenplanung und Aufgabenzuweisung

Die Software muss die Planung von Routen für die UGVs unterstützen. Entweder über eine Benutzerschnittstelle, über eine Mobile App oder über eine Konfigurationsdatei sollen Routen vorgegeben werden können. Es ist davon auszugehen, dass die Routenplanung wesentlich durch einen Menschen unterstützt und nicht automatisch durchgeführt werden soll.

Es sollte die Möglichkeit geben, Aufgaben dynamisch den verfügbaren UGVs zuzuweisen und dabei verschiedene Kriterien zu berücksichtigen, wie z. B. die aktuelle Position, die Ladungskapazität und die Batterielaufzeit der Fahrzeuge. Eine Aufgabe könnte dann aus dem Befahren einer oder mehrerer Routen bestehen.

1.3. Kommunikation und Koordination

Die Software muss eine zuverlässige Kommunikation zwischen den UGVs und dem Backend-System gewährleisten. Als Schnittstellentechnologie wird REST vorgegeben.

Es sollte Mechanismen für die Koordination und Zusammenarbeit zwischen den UGVs geben, um Kollisionen zu vermeiden und Aufgaben effizient zu erledigen. Diese Mechanismen dürfen über das Backend als zentrale Stelle für die Koordination laufen. Es müssen keine individuellen, lokal auf dem UGV befindlichen Algorithmen zur Koordination entwickelt werden.

1.4. Leistungsüberwachung und Diagnose:

Die Software sollte Leistungsdaten von jedem UGV sammeln und analysieren, um Betriebsprobleme frühzeitig zu erkennen. Das können zum Beispiel der Batterieladestand, Fehlercodes oder Abweichungen vom normalen Betriebsverhalten sein.

Es sollte Funktionen zur Diagnose und Fehlerbehebung bieten, um Probleme wie Batterieausfälle, Sensorausfälle oder Navigationsprobleme zu identifizieren und zu lösen.

1.5. Sicherheit und Authentifizierung:

Die Software muss Sicherheitsmechanismen implementieren, um unbefugten Zugriff auf das Backend-System und die UGVs zu verhindern.

Es sollte eine robuste Authentifizierung und Autorisierung für Benutzer und Administratoren geben, um sicherzustellen, dass nur autorisierte Personen auf das System zugreifen können.

1.6. Skalierbarkeit und Flexibilität:

Die Software muss skalierbar sein, um mit einer wachsenden Anzahl von UGVs, entsprechend zugewiesenen Aufgaben und einer zunehmenden Komplexität der Betriebsabläufe umgehen zu können.

Es sollte flexibel genug sein, um verschiedene Arten von UGVs zu unterstützen und sich an sich ändernde Anforderungen und Technologien anzupassen.

1.7. Datenmanagement und Analyse:

Die Software sollte eine zentrale Datenbank für die Speicherung von Fahrzeugdaten, Betriebsprotokollen und Leistungsstatistiken bereitstellen.

Es sollte Funktionen für die Analyse von Betriebsdaten bieten, um Einblicke in die Leistung der Flotte zu gewinnen und Verbesserungspotenziale zu identifizieren.

2. Konzept und Datenmodellierung

2.1. Systemkonzeption

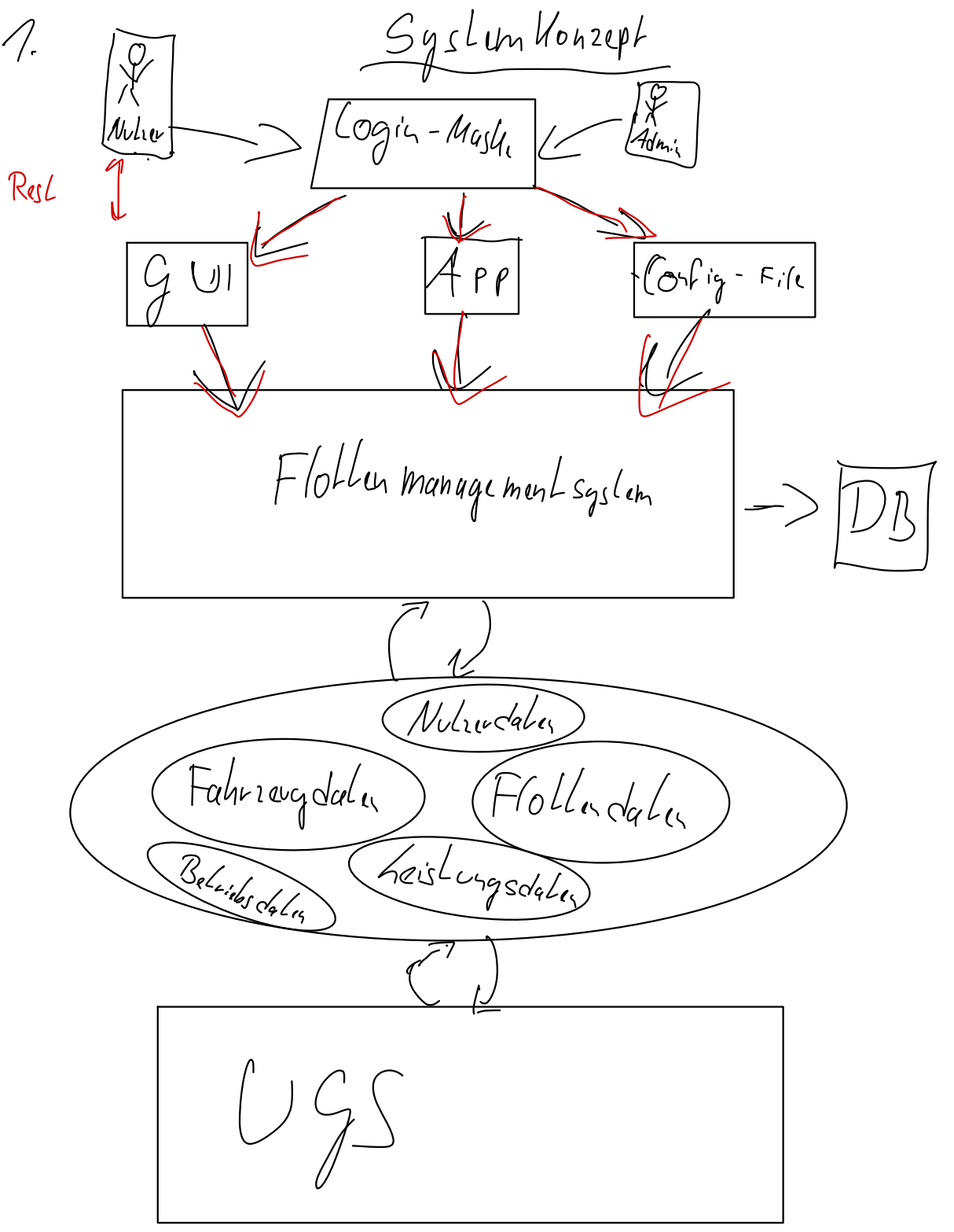
Erstellen Sie ein Systemkonzept und beschreiben Sie es mit Hilfe von FMC (http://www.fmc-modeling.org).

2.2. Anforderungen differenzieren

Differenzieren Sie die Anforderungen in funktional und nicht-funktional.

2.3. Datenmodell

Erstellen Sie ein Datenmodell für das FMS. Fokussieren Sie zunächst auf einfacheren Anforderungen. Bauen Sie es später mit den komplexeren Anforderungen aus.


1.
System Konzept
Nutzer
Login - Maske
Admin
REST
GUI
APP
Config - File
Flotten management system
DB
Nutzerdaten
Fahrzeugdaten
Flottendaten
Betriebsdaten
Leistungsdaten
UGS

2.

funktionale Anforderung

=> was das System tut

- muss eine Loginmaske geben
- muss ein DB geben
- muss mehrere Schnittstellen haben
  Nutzeroberfläche

nicht funktionale Anford.

=> wie das System tut

- Skalierbarkeit -> muss mit wachsender Anzahl von UAs klar kommen
- über REST kommunizieren
- Daten in Echtzeit übertragen
- kein Fremder darf auf das System drauf
- muss mit vielen Daten umgehen

3.

Fahrzeugdaten

ID
Aktuelle Position
Ladungskapazität
Batterielaufzeit

Leistungsdaten

Fahrzeug_ID
Batterieladestand
Fehlercodes
Abweichung Betriebsverhalten

Nutzerdaten

Name
PW -> nicht in DB
  -> verschlüsselt
Recht

Flottendaten

ID
Anzahl Fahrzeuge
Anzahl defekter Fahrzeuge
Auslastung