 Template

Januar 2023

**Über arc42**

arc42, das Template zur Dokumentation von Software- und Systemarchitekturen.

Template Version 8.2 DE. (basiert auf AsciiDoc Version), Januar 2023

Created, maintained and © by Dr. Peter Hruschka, Dr. Gernot Starke and contributors. Siehe <https://arc42.org>.

# Einführung und Ziele

Konzipiert wird ein System, was die Einhaltung der Gesetzgebung zu CO² für schwere Nutzfahrzeuge gemäß EU-Verordnung 2017/2400 ermöglich. Die Verordnung gilt für Fahrzeuge, die folgende Kriterien erfüllen:

* Batterieelektrisch angetrieben
* Gehören zu Fahrzeugklasse 5,9,10,12
* Produktionsdatum nach 01.01.2023

Der Dokumentdruck soll auf Basis eingegebenen Konfigurationsnummer des Fahrzeuges ein Simulationsergebnis erhalten. Diese wird auf Basis von Daten durchgeführt, die bei Logistik und Fahrzeugentwicklung angefragt werden.

Es wird ein reibungsloser Prozess angestrebt und auf Basis der „Pipes und Filter“-Architektur eine schnelle Distribution der Daten.

## Aufgabenstellung

Um einen reibungslosen Ablauf und vor allem eine schnelle Bearbeitung insbesondere so, dass die Triftigkeit der Daten sichergestellt ist, müssen wir ein robustes und leistungsfähiges System bzw. eine geeignete Software-Architektur für das Szenario „Fahrzeugsimulation für E-Autos“ entwerfen. Hierfür müssen alle beteiligten Akteure und Systeme berücksichtigt werden. Entsprechend soll die Entscheidung implementiert werden.

## Qualitätsziele

Wir wollen eine sichere und effiziente Erfüllung der EU-Verordnung gewährleisten.

## Stakeholder

| Rolle |  | Erwartungshaltung |
| --- | --- | --- |
| *Dokumentendruck* |  | *Minimaler Aufwand, Korrektheit der Simulation* |
| *Logistik*  *Fahrzeugentwicklung*  *Unser Team* |  | *Minimaler Aufwand*  *Minimaler Aufwand*  *Reibungsloser Austausch zwischen den verschiedenen Instanzen, Korrekt der Simulation und Distribution* |

# Kontextabgrenzung

## Fachlicher Kontext

Aufgrund der Änderungsverordnung der EU muss der Dokumentdruck rechtlich korrekte Dokumente für Fahrzeuge drucken, die nach dem 01.01.2023 produziert wurden, zu einer der Fahrzeugklassen 5,9,10,12 und batterieelektrisch angetrieben sind. bestehen zwischen Dokumentendruck und Software, Logistik und Software und zwischen Fahrzeugentwicklung und Software. Dabei liefert der Dokumentendruck die notwendige Konfigurationsnummer, auf Basis derer Logistik und Fahrzeugentwicklung die notwendigen Berechnungen durchführen. Anschließend führt die Software die benötigte Simulation durch und überliefert Sie zurück an den Dokumentendruck.

## Technischer Kontext

Schnittstelle bestehen zwischen Dokumentendruck und Software, Logistik und Software und zwischen Fahrzeugentwicklung und Software. Dabei liefert der Dokumentendruck die notwendige Konfigurationsnummer, auf Basis derer Logistik und Fahrzeugentwicklung die notwendigen Berechnungen durchführen. Anschließend führt die Software die benötigte Simulation durch und überliefert Sie zurück an den Dokumentendruck. Dies ist auf der folgenden Abbildung zu sehen, die einen ersten Entwurf des technischen Kontexts darstellt:

Ein Bild, das Diagramm, Plan enthält.

Automatisch generierte Beschreibung

# Bausteinsicht

## Whitebox Gesamtsystem

Begründung

Für die Wahl einer geeigneten Architektur ist es unabdingbar, dass wir die einzelnen beteiligten Instanzen gut verstehen und im Einzelnen betrachtet haben, damit wir das System akkurat aufstellen können.

Enthaltene Bausteine

### Logistik

Berechnet auf Basis der Konfigurationsnummer, die er von uns (Software) bekommt, das Gewicht und die Abmessung eines gesamten Fahrzeuges. Diese zwei Ergebnisse werden dann zurück an die Software geleitet.

### Dokumentdruck

Stellt die Konfigurationsnummer und benötigt von der Software das Simulationsergebnis um die Fahrzeugdokumente gemäß den Richtlinien zu drucken.

Benötigt das Simulationsergebnis unserer Software. Stellt die Konfigurationsnummer, auf Basis derer die notwendigen Berechnungen von Logistik und Fahrzeugentwicklung durchgeführt werden.

### Fahrzeugentwicklung

Stellt auf Basis der Konfigurationsnummer die passenden Messdaten für die jeweiligen Komponenten bereit. Leitet diese zurück an die Software.

### Software

Ermöglicht die Distribution der Schlüsseldaten der involvierten Instanzen und liefert das Simulationsergebnis, was für den Dokumentendruck erforderlich ist.

### VECTO-Library

Enthält alle notwendigen Funktionen, die für die Durchführung des Simulationsergebnisses erforderlich sind. Stellt die Basis der zu programmierenden Software bereit.

Relevante Schnittstellen

*Dokumentendruck 🡪 Software*

*Software 🡪 Logistik*

*Logistik 🡪 Software*

*Software 🡪 Fahrzeugentwicklung*

*Fahrzeugentwicklung 🡪 Software*

*Software 🡪 Dokumentendruck*

*Ein Bild, das Diagramm enthält.

Automatisch generierte Beschreibung*

Modellierung in C4:

Wie in auf den folgenden Abbildungen zu sehen, haben wir drei Ebenen: die erste Ebene ist sehr abstrakt gehalten und stellt den Zusammenhang dar. Die zweite Ebene ist vertiefter und ergänzt die erste Ebene um jegliche Container. Die dritte Ebene wird ebenso ergänzt um die Komponenten. Die Pfeile beschreiben die Abhängigkeiten bzw. Verhältnisse und Verwendungen.

Ein Bild, das Diagramm enthält.

Automatisch generierte Beschreibung

Ein Bild, das Diagramm enthält.

Automatisch generierte Beschreibung

Ein Bild, das Diagramm enthält.

Automatisch generierte Beschreibung

# Laufzeitsicht

Der standardmäßige Workflow verläuft wie folgt:

1. Dokumentendruck stellt die Konfigurationsnummer an die Software
2. Weiterleitung der Konfigurationsnummer an Logistik und Fahrzeugentwicklung
3. Logistik berechnet anhand der Konfigurationsnummer das Gewicht und Abmessung
4. Eingabe der Ergebnisse in die Software
5. Parallel dazu stellt die Fahrzeugentwicklung die Messdaten pro Komponenten
6. Eingabe der Ergebnisse in die Software
7. Simulation auf Basis der Eingaben
8. Zustellung des Simulationsergebnisses an Dokumentendruck

# Architekturentscheidung

Wir haben uns für das Pattern „Pipes & Filter“ entschieden.

Ein Bild, das Diagramm, Plan enthält.

Automatisch generierte Beschreibung

Für die Umsetzung dieses Simulationsergebnisses, haben wir uns für das Architektur-Pattern "Pipes und Filter" entschieden, da dieses Pattern unseren Vorstellungen des Programmablaufs entspricht. Außerdem hat das Pattern den Vorteil, durch seine Modularität, die Wartung und Erweiterung der Software zu erleichtern. Es ist flexibel, sodass man neue Filter mit neuen Funktionen leicht hinzufügen kann, sollte es beispielsweise zu einer erneuten Veränderung der Gesetzeslage kommen. Jeder Filter kann für sich getestet werden, und es gibt eine klare Trennung der Verantwortlichkeiten

In der obigen Abbildung ist folgendes zu sehen: in der ersten Pipe wird die Konfigurationsnummer des Dokumentendrucks an das System übergeben und dort dekodiert. Daraufhin werden dann zuerst über die Schnittstelle von Logistics die Gewichte und Abmessungen abgefragt und anschließend gespeichert. Die nächsten Pipes beinhalten die Abfrage der Messdaten über die Schnittstelle zu Vehicle-Data und ebenfalls das anschließende Speichern. Abschließend wird dann aus diesen Daten das Simulationsergebnis mit Vecto4J erstellt und in der letzten Pipe an den Dokumentendruck zurückgeliefert.

# Entscheidung – Programmiersprache

Unsere Entscheidungsmöglichkeiten dazu, welche Programmiersprache wir für die programmiertechnische Umsetzung des Simulationsergebnisses anwenden, sind relativ begrenzt. Das liegt daran, dass wir bisher nur mit der Programmiersprache "Java" programmiert haben. Somit haben wir uns auch für Java entschieden, und andere Programmiersprachen sind nicht sehr stark in Frage gekommen.

Als Entwicklungsumgebung haben wir uns für "Eclipse" entschieden. Dort haben wir "Packages" erstellt, um den Code in logisch zusammenhängende Gruppen zu organisieren und um eine gewisse Struktur zu schaffen. In diesen befinden sich die jeweiligen Klassen, die benötigt werden, um die Architektur auf programmiertechnische Weise umzusetzen, wie beispielsweise die "Pipes und Filter"-Architektur.

Neben der Tatsache, dass wir bisher nur mit Java programmiert haben, bietet die Programmiersprache auch viele weitere Vorteile. Java ist vor allem plattformunabhängig und kann so auf unterschiedlichen Betriebssystemen ausgeführt werden, ohne dafür den Code selbst ändern zu müssen. Ein weiterer Grund liegt in der Robustheit von Java: dadurch, dass der Compiler die Korrektheit des Codes während der Kompilierung überprüft, werden Fehler bereits zur Kompilierzeit erkannt und nicht erst zur Laufzeit. Außerdem verfügt Java über eine sehr umfangreiche Standardbibliothek. Diese enthält viele nützliche Klassen und Methoden, die uns die Arbeit erleichtern können. Darüber hinaus gilt Java allgemein als eine der beliebtesten und weit verbreitetsten Programmiersprache, sodass es entsprechend viele Entwickler und Ressourcen dazu gibt, die uns bei der Entwicklung und/oder bei Fehlerbehebungen helfen können.

# Qualitätsanforderungen

Es herrschen strenge Qualitätsanforderungen, da der Gesetzgeber involviert ist.

Das System muss sicher sein und es darf keine Datenlecks nach außen geben. Unbefugten bleibt der Zugriff verwehrt.

Des Weiteren sind Laufzeitschwierigkeiten auszuschließen, um zeitnahe Abwicklung des Prozesses zu gewährleisten. Das System muss dahingehend auch ausfallsicher sein. Darüber hinaus ist eine fehlerhafte Simulation durch die Software inakzeptabel. Ebenso ist eine übersichtliche Darstellung bzw. Benutzeroberfläche vorausgesetzt, um Mitarbeitern die Nutzung zu erleichtern.

# Risiken und technische Schulden

Allem voran besteht bei fehlerhafter oder unzuverlässiger Ausführung der Software stets ein rechtliches Risiko. Darunter ordnen sich die verschiedenen potenzielle Gründe für das Eintreten dieses Risikos: Allem voran ein inkorrektes Simulationsergebnis, eine unverständliche grafische Oberfläche die zu menschlichen Fehlern bzw. Fehleingaben führt und ein totales Ausfallen des Systems.