Template

Inhaltsverzeichnis

Über arc42

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

Erstellt von Dr. Gernot Starke, Dr. Peter Hruschka und Mitwirkenden.

Template Revision: 7.0 DE (asciidoc-based), January 2017

Diese Version des Templates enthält Hilfen und Erläuterungen. Sie dient der Einarbeitung in arc42 sowie dem Verständnis der Konzepte. Für die Dokumentation eigener System verwenden Sie besser die *plain* Version.

1. Einführung und Ziele

Dieser Abschnitt führt in die Aufgabenstellung ein und skizziert die Ziele, die Apollo Auto verfolgt.

1.1. Aufgabenstellung

1.1.1. Was ist Apollo Auto?

Apollo ist eine hochleistungsfähige, flexible Architektur, die die Entwicklung, das Testen und den Einsatz von autonomen Fahrzeugen beschleunigt. Apollo Auto bietet unter Andrem Lösungen für Valet Parking, V2X-Kommunikation und intelligente Lichtsignalanlagen.

1.1.2. Wesentliche Features:

- Valet Parking
  - Software- und Hardware-Integrationslösung. Multifusionslösung bestand aus Fahrzeug, Cloud, HD-Karte und Parkplätzen
  - Bietet hochwertige Dienstleistungen, wie automatische Parkplatzerkennung und autonomes Parken, für Kunden.
- V2X-Kommunikation
  - Interaktionslösung für intelligente Fahrzeuginfrastruktur
  - Apollo V2X umfasst ein intelligentes Transportsystem für Fahrzeug Straßendatenerfassung und intelligente Verarbeitungsanalyse, Verkehrssicherheit und -effizienz
  - Wahrnehmung aller Verkehrsteilnehmer im Sichtfeld und die bereitgestellten straßenseitigen Sensorinformationen können für die Entscheidungsfindung beim autonomen Fahren auf hohem Niveau verwendet werden
  - Wahrnehmung von Verkehrteilnehmern ausserhalb des Sichtfeldes

- Bietet einen vollständigen, kontinuierlichen, multimodalen Datendienst mit niedriger Latenz für L4-Autopilot-Fahrzeuge, die in mehreren Szenarien getestet wurden
- Durch die permanente dynamische Erfassung von Verkehrsinformationen und die Cloud-Integration, wird eine weltweite optimale kollaborative Steuerungsfunktionen für Verkehrsteilnehmer und Verkehrsmanagement erreicht
    - Smart Traffic Signals
- Holographisches Wahrnehmen und Verstehen, basierend auf dem holografischen Wahrnehmungs- und Erkennungssystem
- Status von Fußgängern und Fahrzeugen auf jeder Fahrspur genau erkennen und die Leistung des aktuellen Verkehrsflusses wie Volumen, Warteschlangenlänge, Verspätungen usw.
- Vollständige raum-zeitliche Ableitung und Entscheidungsfindung
- Echtzeitsteuerung der gesamten Szene
- Reduzierung der durchschnittlichen Wartezeit um 20-30% während der Rush Hour
    - Robotaxi
- Die Robotaxis, die aus Chinas erstem werkseitig installierten L4-Passagier-Fahrzeug sind zur Zeit auf öffentlichen Straßen im Testbetrieb
- Sie werden in Kooperation von Baido und FAW an einer gemeinsamen Produktionsline hergstellt
    - Minibus
- Die Minibusse ermöglichen ebenfalls autonomes Fahren der Stufe 4
- Funktionen sind unter Anderem Hinderniserkennung und -vermeidung, zu einem Zielort Fahren und Kreuzungen überqueren

1.2. Qualitätsziele

Die folgende Tabelle beschreibt die zentralen Qualitätsziele von DokChess, wobei die Reihenfolge eine grobe Orientierung bezüglich der Wichtigkeit vorgibt.

Qualitätsziel	Motivation und Erläuterung
Zugängliches Beispiel (Analysierbarkeit)	Apollo Auto ist eine offene Plattform, daher ist es wichtig, dass sich neue Entwickler möglichst schnell in die Architektur, Entwurf und Implementierung einarbeiten können
Echzeitsteuerung von einzelnen Fahrzeugen und Verkehrströmen	Apollo Auto übernimmt zuverlässig und sicher die autonome Steuerung von Fashrzeugen auf Level 4
Echtzeit Umfelderkennung	Für die Steuerung von Fahrzeugen wird ein Modell des Umfelds benötigt. Aus den Sensoprdaten wird ein digitales Abbild des Fahrweges, von beweglichen und unbeweglichen Hindernissen und von Signalen geschaffen
Prediction	Um die Fahrsicherheit weiter zu erhöhen, wird auf die Sensor- und Zustandsdaten von anderen Verkehrsteilnehmern in Echtzeit zugegriffen

1.3. Stakeholder

Die folgende Tabelle stellt die Stakeholder von Apollo Auto und ihre jeweilige Intention dar.

Rolle	Interesse, Bezug
Softwarearchitekten	Wollen ein Gefühl bekommen, wie Architekturdokumentation für ein konkretes System aussehen kann. Möchten sich Dinge (z.B. Form, Notation) für Ihre tägliche Arbeit abgucken. Gewinnen Sicherheit für Ihre eigenen Projekte.
Entwickler	Nehmen Architekturaufgaben im Team wahr. Brauchen ein generelles Verständnis für die Architektur.
OEM & Lieferanten	Entwickeln neue Produkte auf Grundlage von Apollo Auto. Wollen Anregungen für eigene Produkte finden.
Gesetzgeber & Genehmigungsbehörden	Entwickeln einen gesetzlichen Rahmen zur Zulassung von fahrerlosen Fahrzeugen im öffentlichen Straßenverkehr. Etablieren Prüfvorschriften und Tests für Genehmigungsverfahren.
Universitäten	Entwickeln eigene Forschungsprojekte auf Grundlage von Apollo Auto. Wollen Anregungen für weitere Forschungsprojekte und studentische Arbeiten finden.
Studenten	Interessieren sich aufgrund ihres Studiums für die verschiedenen Aspekte einer Architekturdokumentation. Setzen eigene Projekte (z.B. Masterarbeit) zum Thema autonomes Fahren mit Apollo Auto um. Schreiben eine Architekturdokumentation zu Apollo Auto.

2. Randbedingungen

Beim Einsatz von Apollo sind verschiedene Randbedingungen zu beachten. Dieser Abschnitt stellt sie dar und erklärt auch – wo nötig – deren Motivation.

2.1. Technische Randbedingungen

- Für den Einsatz von Apollo Auto wird eine anspruchsvolle Hardwareausstattung benötigt. Eine Umsetzung mit einem marktüblichen Standard-Notebook allein ist nicht möglich.
- Es wird ein Fahrzeug benötigt, dass mit By-Wire-Systemen ausgestattet ist, zum Beispiel Brake-by-Wire, Steering-by-Wire, Throttle-by-Wire oder Shift-by-Wire (Apollo wird derzeit auf Lincoln MKZ getestet).
- Ein Rechner mit einem 4-Kern-Prozessor und mindestens 8 GB Speicher (16 GB für Apollo 3.5 und höher)
- Ubuntu 18.04
- Zusätzlich wird eine umfangreiche Sensorik benötigt die Bild- und Abstandsinformationen aus dem Umfeld aufnehmen
- Arbeitskenntnisse über Docker

2.2. Organisatorische Randbedingungen

Randbedingung	Erläuterungen, Hintergrund
github	*Quellcode ist über github verfügbar.*
Bereitstellung von Daten	*Alle Daten müssen in einem Format hochgeladen werden, das den Apollo-Datenspezifikationen entspricht.*
Speicherung von Daten	*Daten, die in China gesammelt wurden, dürfen nur auf Servern in China gespeichert werden. Daten, die in anderen Ländern und Regionen erhoben werden, unterliegen den Beschränkungen der Datenspeicherung, die durch die Gesetze der jeweiligen Länder festgelegt sind.*
Bereitstellung von Daten	*Als Initiator dieser Plattform stellt Baidu die Ausgangsdaten für diese Plattform bereit. Die Daten stehen allen Partnern dieser Plattform offen. Das Prinzip der fairen Daten stellt sicher, dass Partner mit größeren eigenen Beiträgen mehr Daten und Dienste von dieser Plattform erhalten.*
Datenschutz	*Jeder Partner kann seine eigenen Daten anzeigen und die Datenschutzeigenschaften der Daten als privat oder öffentlich festlegen. Die von Partnern hochgeladenen Daten gelten standardmäßig als privat.*
Endscheidungsführung	*Grundsätzlich handelt es sich bei Apollo Auto um eine offene Plattform. Allerdings wirde Baidu um die architektonische Integrität, die Systemzuverlässigkeit und die schnelle Entwicklung von Apollo zu gewährleisten, bei Bedarf wichtige Entscheidungen treffen, während die aktive Beteiligung der breiteren Gemeinschaft erhalten bleibt.*

2.3. Konventionen

Konvention	Erläuterungen, Hintergrund
Dokumentation	*Terminologie und Gliederung nach dem deutschen arc42-Template in der Version 6.0*
Spezifische Datenformate und Frameworks für autonomes Fahren	*Verwendung etablierter Standards für autonomes Fahren. zum Beispiel sind alle Softwaremodule als ROS(Robot Operating System)-Knoten zu behandeln.*

3. Kontextabgrenzung

Dieser Abschnitt beschreibt das Umfeld von Apollo Auto. Für welche Benutzer ist es da, und mit welchen Fremdsystemen interagiert es?

3.1. Fachlicher Kontext

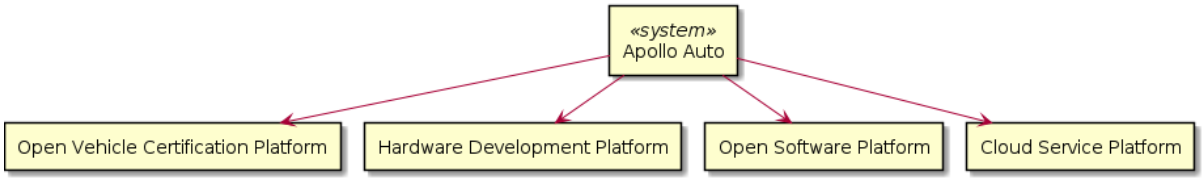


Abbildung 1. Fachlicher Kontext von Apollo Auto

3.1.1. Open Vehicle Certification Platform

Die Open Vehicle Certification Platform schlug eine standardisierte Schnittstelle zwischen dem Autonomen Fahrsystem und dem Fahrzeug vor. Durch diese Plattform können die Fahrzeuganbieter das Fahrzeug einfach mit der offenen Plattform von Apollo verbinden, mehr Entwickler für autonomes Fahren abdecken und die Entwicklung beschleunigen.

3.1.2. Hardware Development Platform

Unter der Hardware Development Platform werden einige durch Apollo Auto unterstütze Hardwarekomponenten zusammengefasst.

Diese sind beispielsweise Sensoren wie Kamera und Lidar sowie Rechen- und Adaptersysteme.

3.1.3. Open Software Platform

Die Open Software Platform fasst alle Softwaremodule zusammen, die von Apollo Auto verwendet werden, um ein autonomes Fahren zu ermöglichen.

Hierzu zählen beispielsweise Module zur Routenplanung, Vorhersage und Wahrneh

3.1.4. Cloud Service Platform

Die Cloud Service Platform fasst ergänzende cloud-basierte Module zusammen die von Apollo Auto verwendet werden.

Hierzu zählen beispielsweise Security, V2X und OTA.

3.2. Technischer Kontext

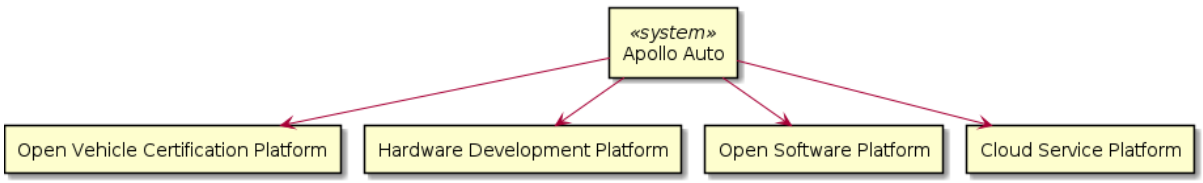


Abbildung 2. Technischer Kontext von Apollo Auto

4. Lösungsstrategie

Dieser Abschnitt enthält einen stark verdichteten Architekturüberblick. Eine Gegenüberstellung der wichtigsten Ziele und Lösungsansätze.

irgendwas zu interoperabilität weil versch. sensoren etc verwendet werden können

5. Bausteinsicht

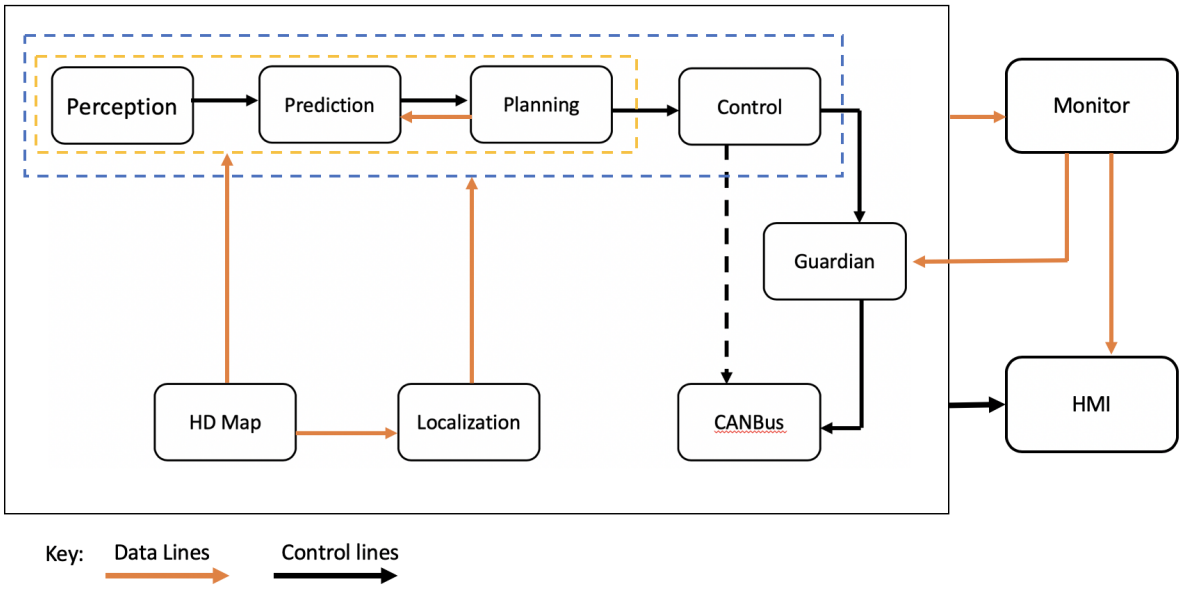
Dieser Abschnitt beschreibt die Zerlegung von Apollo Auto in Module, wie sie sich auch in der Paketstruktur des Java-Quelltextes widerspiegelt. Module der ersten Zerlegungsebene bezeichnen wir in DokChess als Subsysteme. Die → Bausteinsicht, Ebene 1 stellt sie inklusive ihrer Schnittstellen dar.

Für das Subsystem Engine enthält dieser Überblick auch eine detailliertere Zerlegung in → Ebene 2.

5.1. Whitebox Gesamtsystem

DokChess zerfällt wie in Bild unten dargestellt in vier Subsysteme. Die gestrichelten Pfeile stellen fachliche Abhängigkeiten der Subsysteme untereinander dar ("x → y" für "x ist abhängig von y"). Die Kästchen auf der Membran des Systems sind Interaktionspunkte mit Außenstehenden (→ 3.2 Kontextabgrenzung).

Subsystem Kurzbeschreibung XBoard-Protokoll Realisiert die Kommunikation mit einem Client mit Hilfe des XBoard-Protokolls. Spielregeln Beinhaltet die Schachregeln und kann z.B. zu einer Stellung alle gültigen Züge ermitteln. Engine Beinhaltet die Ermittlung eines nächsten Zuges ausgehend von einer Spielsituation. Eröffnung Stellt Züge aus der Eröffnungsliteratur zu einer Spielsituation bereit. Tabelle: Überblick über Subsysteme von DokChess



Begründung

<Erläuternder Text>

Enthaltene Bausteine

Name	Verantwortung
Guardian	Ein Sicherheitsmodul, das die Funktion eines Aktionszentrums übernimmt und eingreift, wenn Monitor einen Fehler erkennt.
Monitor	Das Überwachungssystem aller Module im Fahrzeug inklusive Hardware.
HMI/Dreamview	Human Machine Interface oder DreamView in Apollo ist ein Modul zur Anzeige des Fahrzeugstatus, zum Testen anderer Module und zur Steuerung der Funktion des Fahrzeugs in Echtzeit.
CANBus	Der CanBus ist die Schnittstelle, die Steuerbefehle an die Fahrzeughardware weitergibt. Außerdem gibt er die Fahrwerksinformationen an das Softwaresystem weiter.
Control	Das Control-Modul führt die geplante räumlich-zeitliche Trajektorie aus, indem es Steuerbefehle wie Gas, Bremse und Lenkung erzeugt.
Planning	Das Planning-Modul plant die räumlich-zeitliche Trajektorie, die das autonome Fahrzeug fahren soll.
Routing	Das Routing Modul sagt dem autonomen Fahrzeug, wie es sein Ziel über eine Reihe von Fahrspuren oder Straßen erreichen kann.
Localization	Das Localization-Modul nutzt verschiedene Informationsquellen wie GPS, LiDAR und IMU, um zu schätzen, wo sich das autonome Fahrzeug befindet.

Name	Verantwortung
Prediction	Das Prediction-Modul antizipiert die zukünftigen Bewegungstrajektorien der wahrgenommenen Hindernisse.
Perception	Das Perception-Modul identifiziert die Welt, die das autonome Fahrzeug umgibt. Innerhalb Perception gibt es zwei wichtige Teilmodule: Hinderniserkennung und Ampelerkennung.

Wichtige Schnittstellen

<Beschreibung wichtiger Schnittstellen>

5.2. Guardian

5.2.1. Zweck/Verantwortung

Ein Sicherheitsmodul, dass die Funktion eines "Action Centers" übernimmt und eingreift, wenn das Modul Monitor einen Fehler erkennt. ==== Schnittstelle(n) ==== Qualitäts-/Leistungsmerkmale

- Alle Module funktionieren einwandfrei - Guardian lässt den Steuerfluss normal funktionieren. Steuersignale werden an den CANBus gesendet, als ob Guardian nicht vorhanden wäre.
- Wenn ein Fehler von Monitor erkannt wird, verhindert Guardian, dass Steuersignale den CANBus erreichen und bringt das Auto zum Stillstand. Es gibt 3 Möglichkeiten, wie Guardian entscheidet, wie das Fahrzeug anzuhalten ist. Dazu verwendet Guardian die Ultraschallsensoren
  - Wenn der Ultraschallsensor einwandfrei funktioniert, ohne ein Hindernis zu erkennen, bringt Guardian das Auto langsam zum Stehen
  - Reagieren die Ultraschallsensoren nicht, bremst Guardian hart ab, um das Auto sofort zum Stillstand zu bringen.
  - Ein Sonderfall liegt vor, wenn das HMI den Fahrer über einen drohenden Unfall informiert und der Fahrer 10 Sekunden lang nicht

eingreift, führt Guardian eine Vollbremsung durch, um das Fahrzeug sofort zum Stehen zu bringen.

5.2.2. Ablageort/Datei(en)

https://github.com/ApolloAuto/apollo/tree/r5.5.0/modules/guardian

5.3. Monitor

5.3.1. Zweck/Verantwortung

Das Überwachungssystem für alle Module im Fahrzeug einschließlich der Hardware. Monitor empfängt Daten von verschiedenen Modulen und leitet sie an die HMI weiter, damit der Fahrer sie sehen und sicherstellen kann, dass alle Module ohne Probleme funktionieren. Im Falle eines Modul- oder Hardwarefehlers sendet Monitor eine Warnung an Guardian, das dann entscheidet, welche Maßnahmen ergriffen werden müssen, um einen Unfall zu verhindern.

5.3.2. Schnittstelle(n)

5.3.3. Qualitäts-/Leistungsmerkmale

Dieses Modul enthält Software auf Systemebene, wie z. B. Code zur Überprüfung des Hardwarestatus und zur Überwachung des Systemzustands. In Apollo 5.5 führt das Monitor-Modul nun u. a. die folgenden Prüfungen durch:

- Status der laufenden Module
- Überwachung der Datenintegrität
- Überwachung der Datenfrequenz
- Überwachung des Systemzustands (z. B. CPU-, Speicher-, Festplattennutzung usw.)
- Erzeugen eines End-to-End-Latenz-Statistikberichts

5.3.4. Ablageort/Datei(en)

https://github.com/ApolloAuto/apollo/tree/r5.5.0/modules/monitor

5.4. HMI/Dreamview

5.4.1. Zweck/Verantwortung

Human Machine Interface oder DreamView in Apollo ist eine Web-Anwendung, die: - die aktuelle Ausgabe relevanter autonomer Fahrmodule visualisiert, z. B. Planung der Trajektorie, Fahrzeuglokalisierung, Fahrwerkstatus usw. - eine Mensch-Maschine-Schnittstelle bietet, über die der Benutzer den Hardwarestatus einsehen, Module ein- und ausschalten und das autonom fahrende Auto starten kann. - bietet Debugging-Tools, wie z. B. PnC-Monitor, zur effizienten Verfolgung von Modulproblemen.

5.4.2. Schnittstelle(n)

5.4.3. Qualitäts-/Leistungsmerkmale

5.4.4. Ablageort/Datei(en)

https://github.com/ApolloAuto/apollo/tree/r5.5.0/modules/dreamview

5.5. CANBus

5.5.1. Zweck/Verantwortung

Der CanBus ist die Schnittstelle, die Steuerbefehle an die Fahrzeughardware weitergibt. Außerdem gibt er die Fahrwerksinformationen an das Softwaresystem weiter.

5.5.2. Schnittstelle(n)

- OnControlCommand - ein ereignisbasierter Publisher mit einer Callback-Funktion, die ausgelöst wird, wenn das CANBus-Modul Steuerbefehle

empfängt

- OnGuardianCommand - ein ereignisbasierter Publisher mit einer Callback-Funktion, die ausgelöst wird, wenn das Guardian-Modul Steuerbefehle empfängt

5.5.3. Ablageort/Datei(en)

https://github.com/ApolloAuto/apollo/tree/r5.5.0/modules/canbus

5.6. Control

5.6.1. Zweck/Verantwortung

Die Control nimmt die geplante Trajektorie als Eingabe und generiert den Steuerbefehl zur Weitergabe an den CANBus.

5.6.2. Schnittstelle(n)

- OnPad
- OnMonitor
- OnChassis
- OnPlanning
- OnLocalization

OnPad und OnMonitor sind Routineinteraktionen mit der PAD-basierten Benutzeroberfläche und Simulationen.

5.6.3. Ablageort/Datei(en)

https://github.com/ApolloAuto/apollo/tree/r5.5.0/modules/control

5.7. Planning

5.7.1. Zweck/Verantwortung

Apollo 3.5 verwendet mehrere Informationsquellen, um eine sichere und kollisionsfreie Trajektorie zu planen, daher interagiert das Planning-Modul mit fast jedem anderen Modul.

Zunächst nimmt das Planning-Modul die Vorhersageausgabe. Da die Vorhersageausgabe das ursprünglich wahrgenommene Hindernis umschließt, abonniert das Planning-Modul die Ausgabe der Ampelerkennung und nicht die Ausgabe der wahrgenommenen Hindernisse.

Dann nimmt das Planning-Modul die Routing-Ausgabe. In bestimmten Szenarien kann das Planning-Modul auch eine neue Routing-Berechnung auslösen, indem es eine Routing-Anforderung sendet, wenn der aktuellen Route nicht treu gefolgt werden kann.

Schließlich muss das Planning-Modul den Standort (Lokalisierung: wo bin ich) sowie die aktuellen autonomen Fahrzeuginformationen (Fahrwerk: wie ist mein Status) kennen.

5.7.2. Schnittstelle(n)

5.7.3. Qualitäts-/Leistungsmerkmale

5.7.4. Ablageort/Datei(en)

https://github.com/ApolloAuto/apollo/tree/r5.5.0/modules/planning

5.8. Routing

5.8.1. Zweck/Verantwortung

Das Routing-Modul muss den Start- und Endpunkt des Routings kennen, um die Durchfahrtsspuren und Straßen zu berechnen. Normalerweise ist der Startpunkt der Standort des autonomen Fahrzeugs. Die RoutingResponse wird wie unten gezeigt berechnet und veröffentlicht.

5.8.2. Schnittstelle(n)

5.8.3. Qualitäts-/Leistungsmerkmale

5.8.4. Ablageort/Datei(en)

https://github.com/ApolloAuto/apollo/tree/r5.5.0/modules/routing

5.9. Localization

5.9.1. Zweck/Verantwortung

Das Lokalisierungsmodul aggregiert verschiedene Daten, um das autonome Fahrzeug zu lokalisieren. Es gibt zwei Arten von Lokalisierungsmodi: OnTimer und Multiple SensorFusion.

Die erste Lokalisierungsmethode ist RTK-basiert, mit einer Timer-basierten Callback-Funktion OnTimer, wie unten gezeigt.

Die andere Lokalisierungsmethode ist die Multiple Sensor Fusion (MSF)-Methode, bei der eine Reihe von ereignisgesteuerten Callback-Funktionen registriert werden, wie unten gezeigt.

5.9.2. Schnittstelle(n)

5.9.3. Qualitäts-/Leistungsmerkmale

5.9.4. Ablageort/Datei(en)

https://github.com/ApolloAuto/apollo/tree/r5.5.0/modules/localization

5.10. Prediction

5.10.1. Zweck/Verantwortung

Das Prediction-Modul schätzt die zukünftigen Bewegungstrajektorien für alle

wahrgenommenen Hindernisse. Die ausgegebene Vorhersagemeldung beinhaltet die Informationen zur Hinderniserkennung. Prediction abonniert Lokalisierungs-, Planungs- und Wahrnehmungs-Hindernis-Nachrichten wie unten dargestellt. Wenn ein Lokalisierungsupdate empfangen wird, aktualisiert das Prediction-Modul seinen internen Status. Die eigentliche Vorhersage wird ausgelöst, wenn Perception ihre Perception-Hindernismeldung aussendet.

5.10.2. Schnittstelle(n)

5.10.3. Ablageort/Datei(en)

https://github.com/ApolloAuto/apollo/tree/r5.5.0/modules/prediction

5.11. Perception

5.11.1. Zweck/Verantwortung

Das Perception-Modul verfügt über die Fähigkeit, 5 Kameras (2 vorne, 2 seitlich und 1 hinten) und 2 Radare (vorne und hinten) zusammen mit 3 16-Linien-LiDARs (2 hinten und 1 vorne) und 1 128-Linien-LiDAR zu verwenden, um Hindernisse zu erkennen und ihre individuellen Spuren zu einer endgültigen Spurliste zu verschmelzen. Das Hindernis-Submodul erkennt, klassifiziert und verfolgt Hindernisse. Dieses Teilmodul sagt auch die Bewegung und Positionsinformationen des Hindernisses voraus (z. B. Richtung und Geschwindigkeit). Für die Fahrspur werden Fahrspurinstanzen durch Nachbearbeitung von Fahrspur-Parsing-Pixeln konstruiert und die relative Position der Fahrspur zum Ego-Fahrzeug berechnet (L0, L1, R0, R1, usw.).

5.11.2. Ablageort/Datei(en)

https://github.com/ApolloAuto/apollo/tree/r5.5.0/modules/perception

5.11.3. <Name Schnittstelle 1>

...

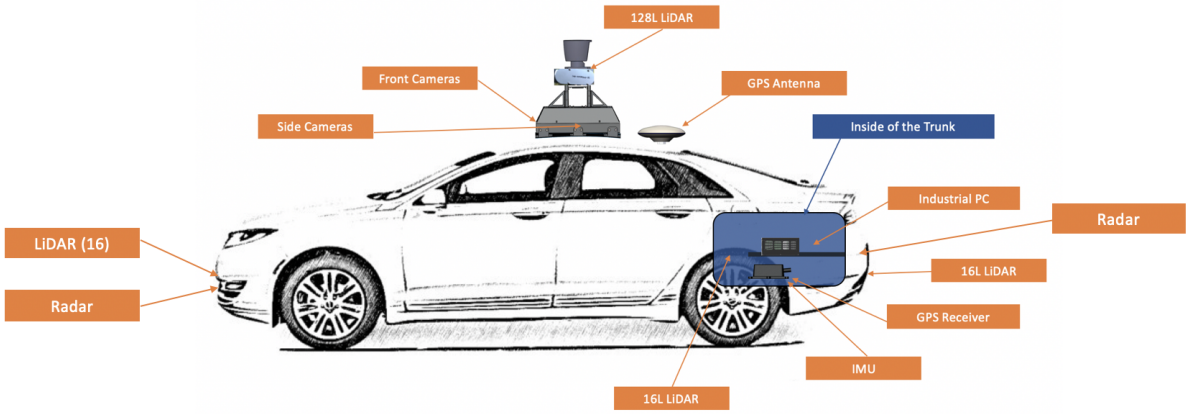
5.11.4. <Name Schnittstelle m>

6. Laufzeitsicht

TBD

7. Verteilungssicht

7.1. Infrastruktur Ebene 1



Begründung

<Erläuternder Text>

Qualitäts- und/oder Leistungsmerkmale

<Erläuternder Text>

Zuordnung von Bausteinen zu Infrastruktur

<Beschreibung der Zuordnung>

7.2. Infrastruktur Ebene 2

7.2.1. *<Infrastrukturelement 1>*

<Diagramm + Erläuterungen>

7.2.2. *<Infrastrukturelement 2>*

<Diagramm + Erläuterungen>

...

7.2.3. *<Infrastrukturelement n>*

<Diagramm + Erläuterungen>

8. Querschnittliche Konzepte

Dieser Abschnitt beschreibt allgemeine Strukturen und Aspekte, die systemweit gelten. Darüber hinaus stellt er verschiedene technische Lösungskonzepte vor.

8.1. Unter-der-Haube

Die Dynamik von autonomen Fahrzeugen (AV) wird von der Planungs-Engine über den Controller Area Network-Bus (CAN-Bus) gesteuert. Die Software liest Daten aus Hardware-Registern und schreibt sie zurück, genau wie wir es in Assembler tun würden. Für hochpräzise Berechnungen arbeiten die Lokalisierungs-, Wahrnehmungs- und Planungsmodule als unabhängige Eingabequellen, während die Ausgabequellen über das Peer2Peer (P2P)-Protokoll zusammenarbeiten. P2P wird durch die RPC-Netzwerkanwendung unterstützt.

Apollo Auto verwendet ROS1 als zugrunde liegendes Netzwerk, was bedeutet, dass Apollo Auto das Master-Nodes-Framework von ROS1 entlehnt. Da xmlRPC von ROS1 wirklich alt ist (im Vergleich zu den neueren brpc und grpc), hat Baidu seine eigene protobuf-Version von RPC entwickelt.

8.2. *<Konzept 2>*

<Erklärung>

...

8.3. *<Konzept n>*

<Erklärung>

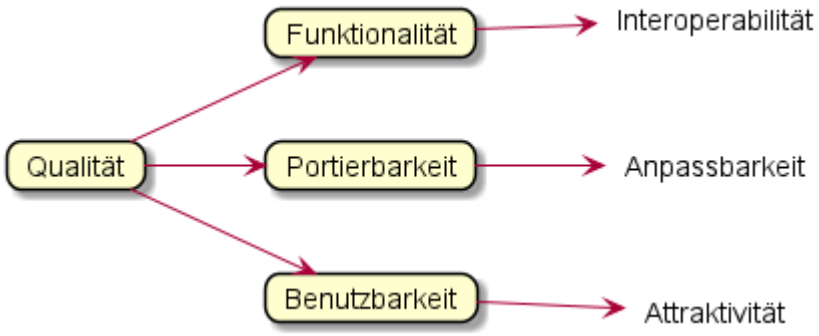
9. Entwurfsentscheidungen

10. Qualitätsanforderungen

Dieser Abschnitt beinhaltet konkrete Qualitätsszenarien, welche die zentralen Qualitätsziele, aber auch andere geforderte Qualitätseigenschaften besser fassen. Sie ermöglichen es, Entscheidungsoptionen zu bewerten.

10.1. Qualitätsbaum

Das folgende Bild gibt einen Überblick über die relevanten Qualitätsmerkmale und den ihnen jeweils zugeordneten Szenarien.



10.2. Qualitätsszenarien

11. Risiken und technische Schulden

Die folgenden Risiken wurden zu Beginn des Vorhabens identifiziert. Sie beeinflussten die Planung der ersten drei Iterationen maßgeblich. Seit Abschluss der dritten Iteration werden sie beherrscht. Dieser Architekturüberblick zeigt die Risiken inklusive der damaligen Eventualfallplanung weiterhin, wegen ihres großen Einflusses auf die Lösung.

12. Glossar

Das folgende Glossar erklärt Begriffe aus dem Bereich Autonomes Fahren.

Begriff	Definition
By-Wire	*Bezeichnung für (zumindest partielles) Fahren oder Steuern von Fahrzeugen ohne mechanische Kraftübertragung der Bedienelemente zu den entsprechenden Stellelementen wie etwa Drosselklappen. Das By-Wire-Konzept umfasst dabei zumindest zwei oder mehr der „X-by-Wire“- Systeme wie etwa Brake-by-Wire (Bremssteuerung) und Steer-by-Wire (Lenkung)..*
LIDAR	*ist eine dem Radar verwandte Methode zur optischen Abstands- und Geschwindigkeitsmessung sowie zur Fernmessung. Statt der Radiowellen wie beim Radar werden Laserstrahlen verwendet.*
RADAR	*ist die Bezeichnung für verschiedene Erkennungs- und Ortungsverfahren und -geräte auf der Basis elektromagnetischer Wellen im Radiofrequenzbereich (Funkwellen).*
P2P	*Peer-to-Peer und Rechner-Rechner-Verbindung sind synonyme Bezeichnungen für eine Kommunikation unter Gleichen, hier bezogen auf ein Rechnernetz.*
RPC	*Remote Procedure Call ist eine Technik zur Realisierung von Interprozesskommunikation. Sie ermöglicht den Aufruf von Funktionen in anderen Adressräumen. Im Normalfall werden die aufgerufenen Funktionen auf einem anderen Computer als das aufrufende Programm ausgeführt.*

13. Anhang

13.1. Apollo

13.2. Arc42

- die Dokumentation muss bereits zu Beginn der Arbeiten an der Architektur/am Quellcode erfolgen um stets den aktuellen Stand abbilden zu können.
- die Dokumentation soll mit dem Projekt wachsen!
- die nachträgliche Dokumentation ist eine Sysiphusarbeit
    - Beispiel:
        - Diagramme in der Laufzeitsicht sind effizienter zu gestalten während ein Modul im Entstehen ist. Änderungen können mit jeder Entwicklungsstufe eingepflegt werden.
        - Ein fertiges Modul in ein Diagramm zu überführen erfordert viel Zeit, da alle beteiligten Module und deren Funktionsweise erst identifiziert werden müssen.

13.3. docToolChain

- plantUML ist zwar bei sehr einfachen Diagrammen eine Erleichterung, kommt aber an beispielsweise draw.io oder tikz (in Latex) nicht heran wenn es um komplexe und detailierte Diagramme geht
    - teilweise werden bei der inline-Erstellung bei plantUML Diagrammeinstellungen nicht übernommen
- Schriftgrößen sind nicht einstellbar, was an manchen Stellen - z.B Tabellen nötig gewesen wäre
- Ansonsten ist docToolChain eine interessante Alternative, besonders in Bezug auf den CI-Task in github.
    - damit nicht bei jedem Commit die Dokumentation neu erstellt wird,

bietet github die Möglichkeit über Commitnachrichten, z.B "[no ci]" oder "[skip ci]" den CI-Task auszusetzen. Hier wäre es sinnvoller über Commitnachrichten explizit die Generierung zu starten.