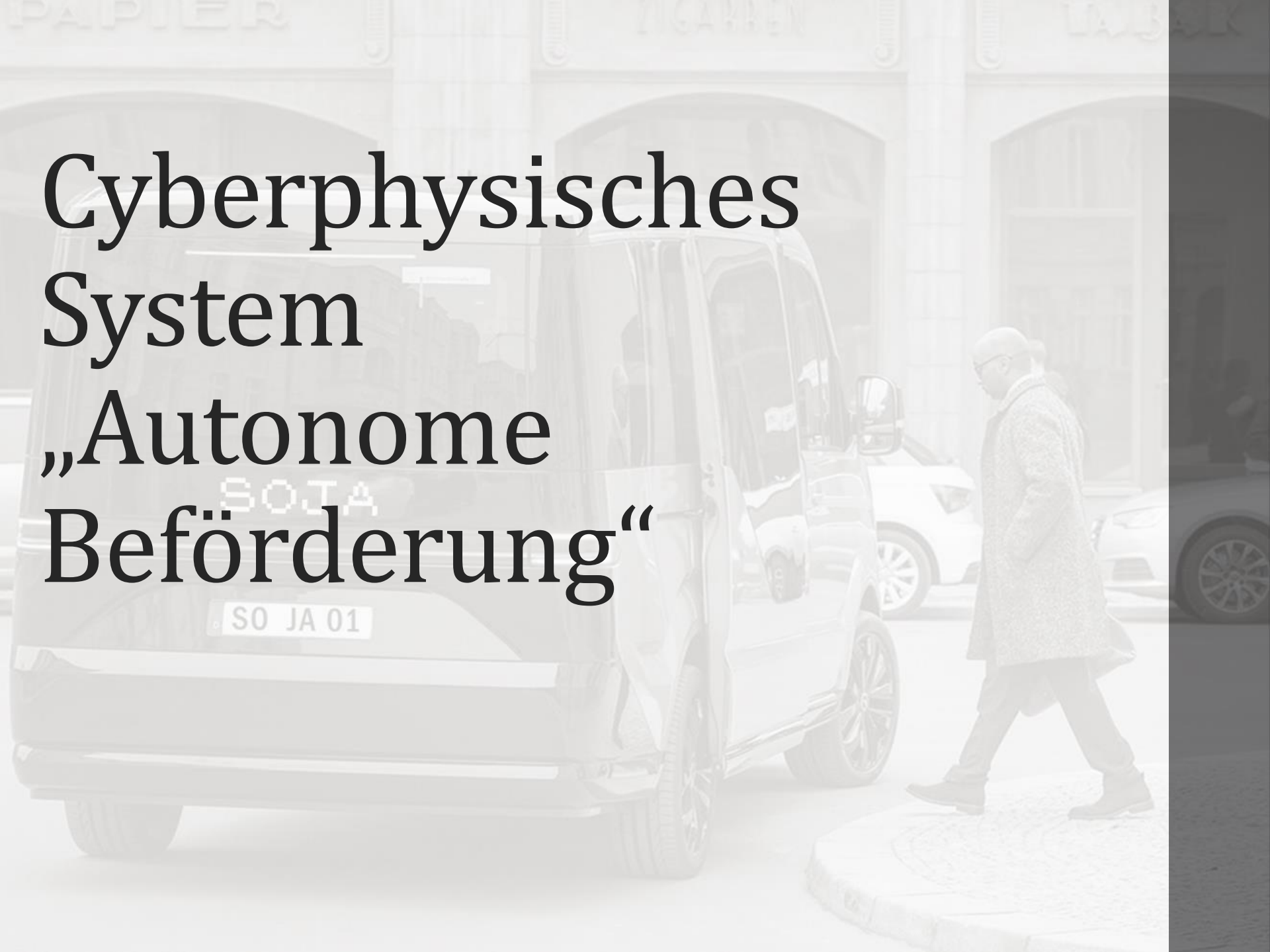
Cyberphysisches System „Autonome Beförderung“



Inhalt

- Einleitung & Präsentation
  - Was
  - Warum
  - Ziele
  - Konzept
  - Demonstration
- Eigenschaften des Systems
  - Zeitliche Anforderungen
  - Funktionale Anforderungen
- Modellierung
  - Uppaal
  - Architektur
- Testen
- Fazit
- Evaluation
- Diskussion

Einleitung
- Worum geht es und warum wird es gebraucht

- Idee: Beförderungs-Dienstleistung als Cyberphysisches System
- Problem: Kann man schneller Personen befördern als öffentl. Verkehrsmittel und das günstiger als Taxis?
- Vorteil durch Autonome Fahrzeuge als „Taxis“: Einsparen von Fahrern und Routen-Adaptivität
- Eventuelle Alternative zu teurem, aber schnellen Taxi und günstigen, aber langsamen öffentlichen Verkehrsmitteln

Einleitung
- Vorteile von CPS für diese Idee

- Geeigneter Ansatz für Mensch-System-Kooperation
- CPS-Schema deckt Ansprüche ab:
  - Selbstorganisation
  - Erheben von Daten der Komponenten zur Analyse
  - Und nachfolgende Anpassung/Rekonfiguration des Systems im laufenden Betrieb
- Verteilbarkeit des Systems
  - Hohe globale Verfügbarkeit
  - Vermeidung von typischen Bottlenecks und Single-Point-of-Failure (v.a. Server-Anwendungen)
- Skalierung der Komponenten

Einleitung
- Warum gibt es das noch nicht

- Straßenverkehrsgesetz: Sieht Fahrzeugführer als Notwendigkeit vor …
- Kann sich mit der Entwicklung sicherer Systeme ändern
- Hohe Sicherheitsanforderungen an Vollautonomie, denn Betreiber haftet i.d.R.
- Scheint erst seit kurzem realistisch
  - Steigende Rechenleistung (Vorteilhaft für größere Programme in Embedded Systems, z.B. in Fahrzeugen)
- Ähnliche Idee wird momentan von VW getestet und entwickelt (MOIA)

Einleitung
- Zentrale Ziele der Entwicklung

- Selbstorganisation: Betrieb soll möglichst ohne menschliche Überwachung funktionieren
- Skalierung: System soll ausbaubar sein und Betrieb mit vielen Teilnehmern stabil bleiben
- Adaptivität: System soll Fahrzeuge „smart“ verteilen (Load Balancing)
- Nutzerkomfort: Schnittstelle für Endnutzer soll leichtgewichtig und einfach nutzbar sein, Endnutzer soll schnell befördert werden
- Robustheit: Kommunikationsausfall und Komponentenausfall soll kompensierbar sein

Einleitung
- Sonstige wünschenswerte Eigenschaften

- Sicherheit der Nutzer (Verantwortung des Autoherstellers)
- Geschützte Privatsphäre (sicherer Kommunikationskanal und ausreichende Verschlüsselung)

Konzept

- Aktoren-Modell für zentrale Schnittstelle
  - Vermeidung von Bottlenecks/Denial-of-Service
  - Garantierte Verfügbarkeit
- Nebenläufigkeit durch Nachrichtenbasierte Kommunikation
- Vorprogrammierte Routen und „Kommandozeilen-Implementierung“ erlauben Fokus auf Kernziele
  - Routenplanung ist Wissenschaft für sich
- Gruppenkommunikation über MQTT:
  - Leichtgewichtiges, weit verbreitetes Protokoll
  - Erlaubt One-to-Many, Many-to-One und One-to-One (über UID)
  - Grundlegende Sicherheit durch Verschlüsselung
- Single-Purpose Komponenten für Übersicht und Wartbarkeit

Demonstration

- Szenario: 2 Routen
  - Große Rundroute in Hamburg
  - Kleine Rundroute in Lübeck Altstadt
- Großteil Fahrzeuge befindet sich in Hamburg
  - Hamburg: 20 Fahrzeuge
  - Lübeck: 5 Fahrzeuge
- … Aber Weihnachtsmarkt in Lübeck sorgt für unüblich hohes Aufkommen an Nutzern
  - Hamburg: 1 Anmeldungen pro Minute
  - Lübeck: 5 Anmeldungen pro Minute

Eigenschaften des Systems -Zeitliche Anforderungen

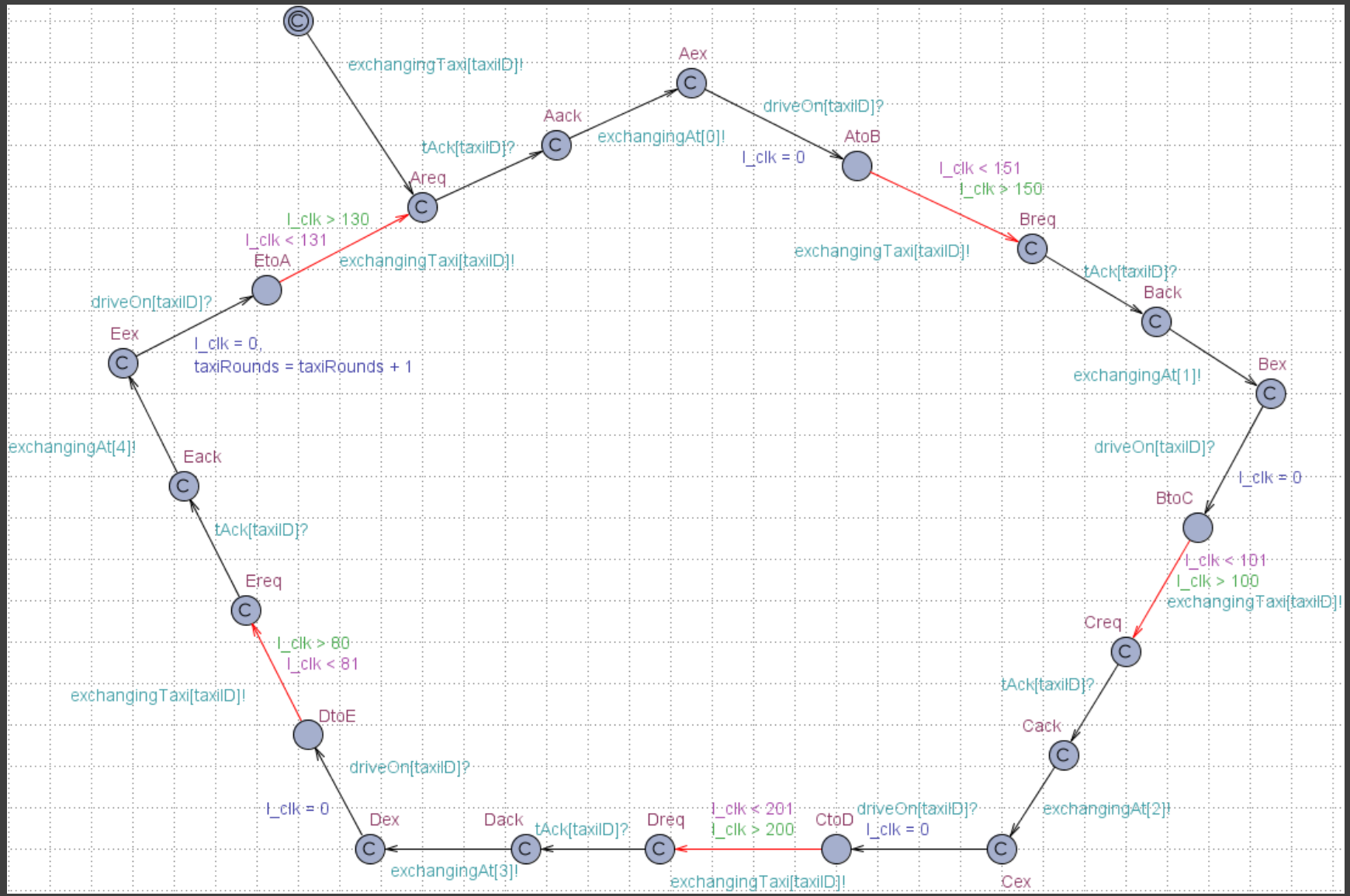
- Nutzer-orientiert (gleichzeitig Qualitätsanforderung):
  - Schnelle Aufnahme
  - Schnelle Transportation
- Reaktionen im Sekundenbereich werden von Datenfluss toleriert
  - Erlaubt komplexe Berechnungen und träge Kommunikation (vgl. Safety bei Autonomen Autos an sich)

Eigenschaften des Systems
-Funktionale Anforderungen

- Jeder Nutzer kommt irgendwann an
- Nutzer kann das erste Fahrzeug mit freien Plätzen nehmen
- Hinzufügen und Entfernen von Komponenten (Fahrzeuge/Nutzer) muss möglich sein

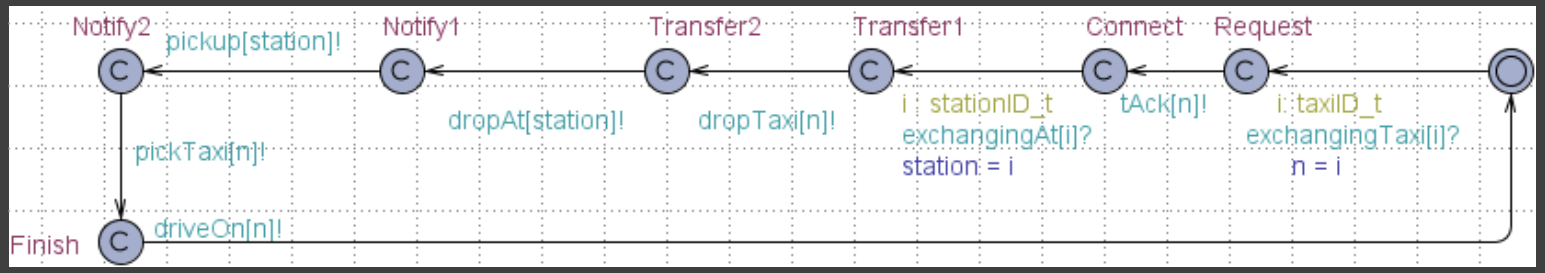
Modellierung
-Uppaal

Auto

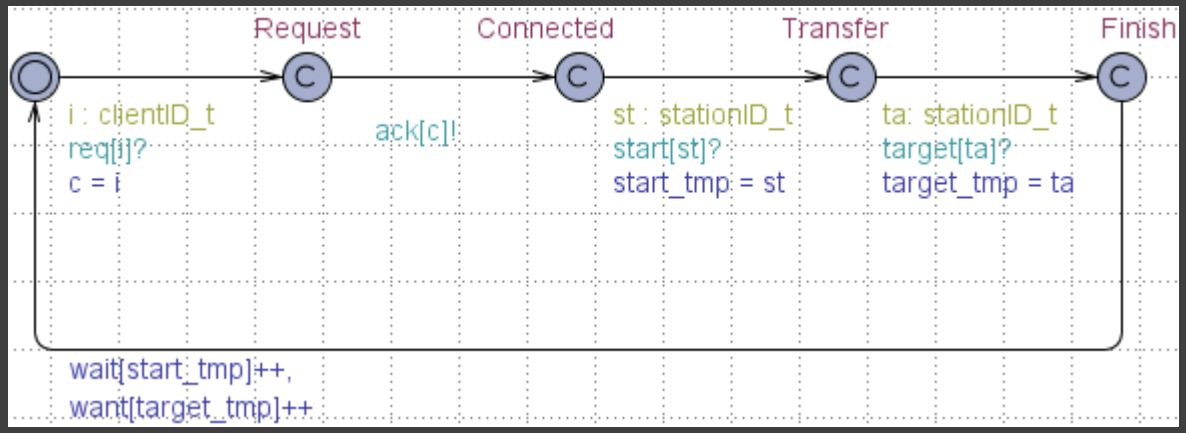


Modellierung
-Uppaal

Server für gebündelten Austausch an Station

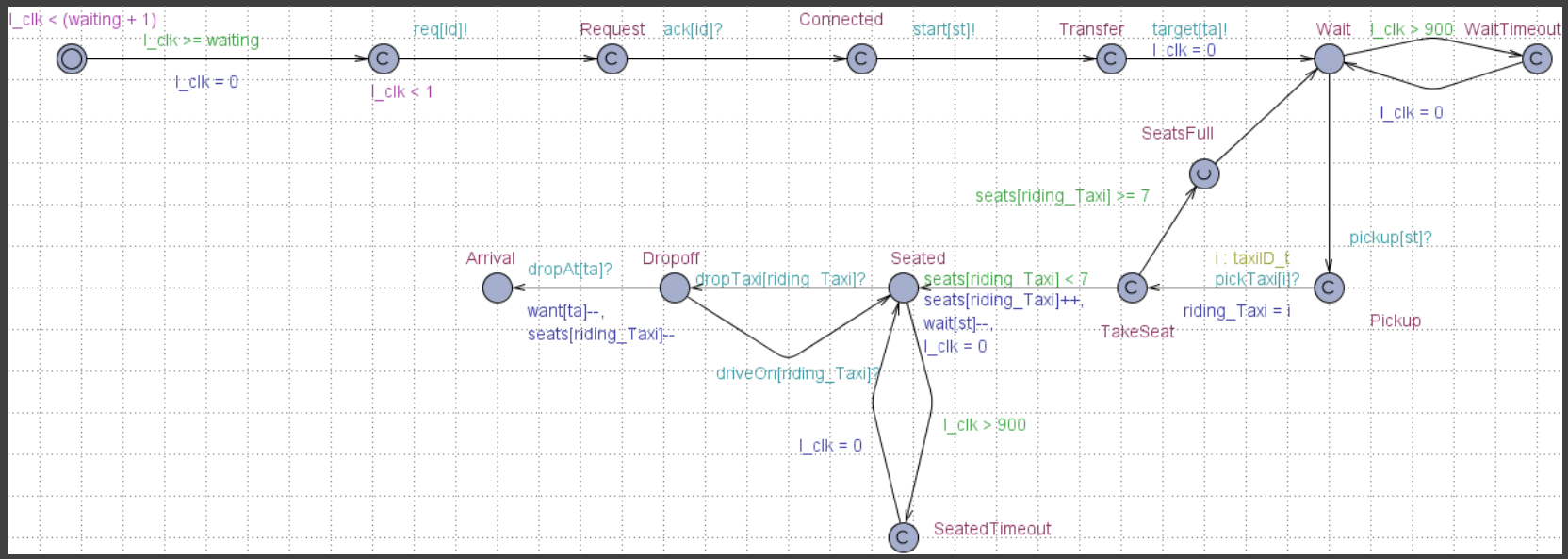


Server für Anmeldung der Nutzer



Modellierung
-Uppaal

Nutzer



Modellierung
-Uppaal: Verifikationsziele

- Folgende Verifikationsziele wurden erfolgreich mit sinnvollen und unterschiedlichen Umgebungsvariablen getestet:
  1. Es bleibt kein Passagier für immer im Fahrzeug sitzen
  2. Es gibt mehr Passagiere als Fahrzeuge und jeder kommt am Ziel an
  3. Passagiere nehmen das erste Fahrzeug falls ein Platz frei ist
  4. Kein Passagier steht im Fahrzeug (es werden nicht mehr Passagiere mitgenommen, als das Fahrzeug Plätze hat)

- Folgende Verifikationsziele wurden unter angemessenen Umgebungsvariablen erfolgreich getestet, schlugen aber bei hohen Kardinalitäten/Werten fehl:
  1. Eine Maximale Wartezeit wird nicht überschritten
  2. (Eine Maximale Fahrzeit wird nicht überschritten)

Modellierung
-Uppaal: Fazit

- Verifikationsziele zeigen Grenzen des Systems auf, die bei der Implementierung berücksichtigt werden müssen, z.B.
- Kommunikationsengpass Server-seitig
  - Betroffene Komponenten im Aktoren-Modell umsetzen
- Prüfen der Werte auf Integrität
  - z.B. Fahrzeug das letzte Wort bei der Aufnahme lassen

Modellierung

No Code

No code is the best way to write secure and reliable applications. Write nothing; deploy nowhere.

Getting Started

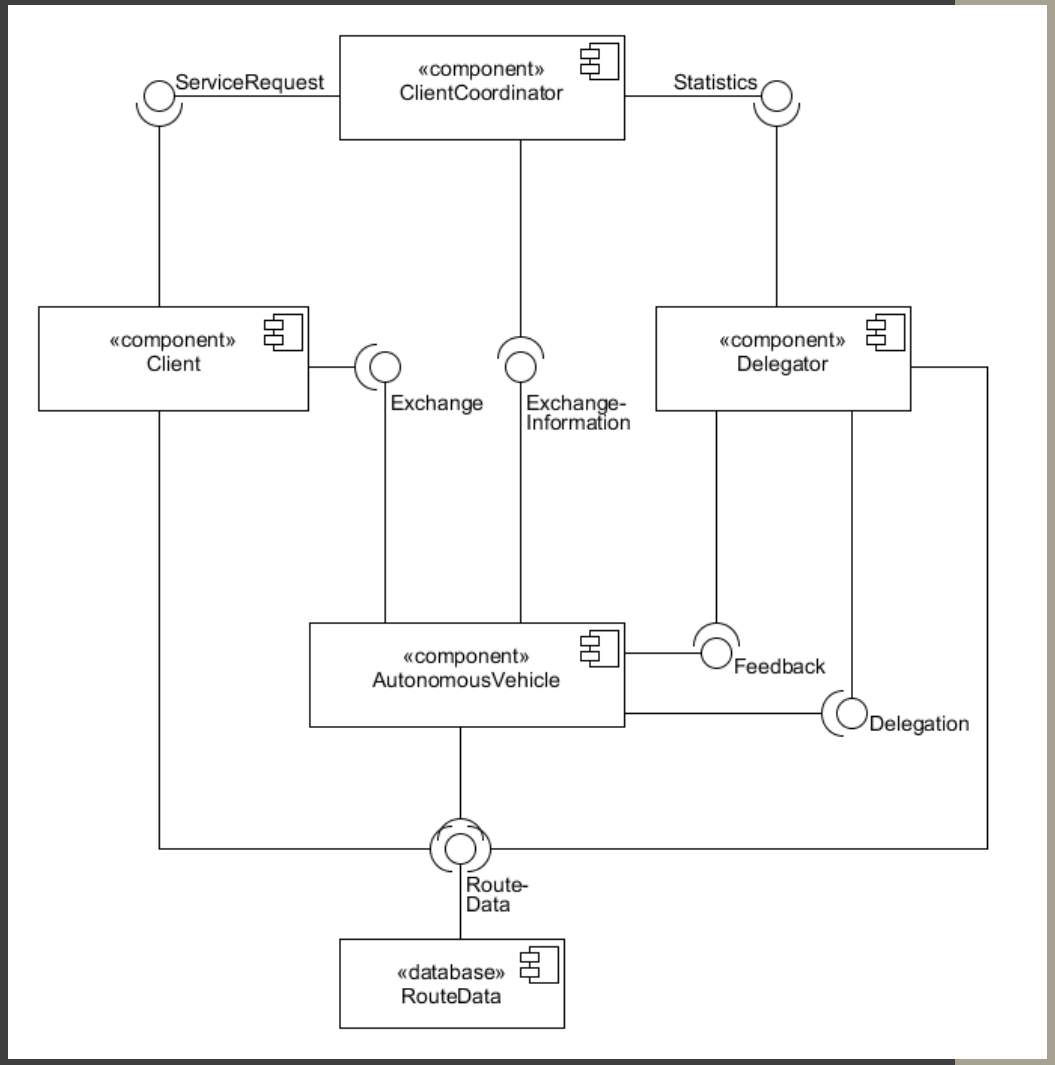
Start by not writing any code.

Adding new features is easy too:

The possibilities are endless.

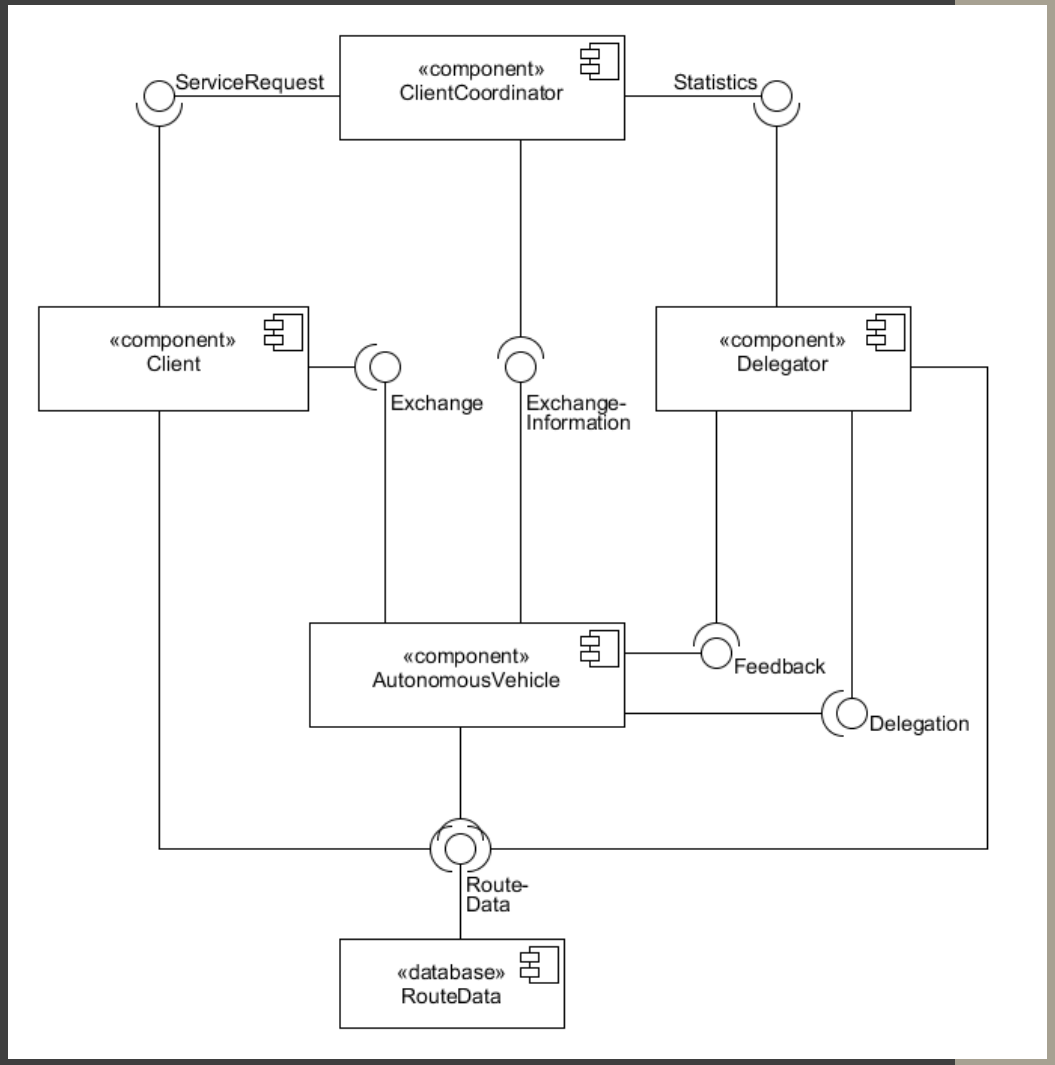
Modellierung -Komponenten

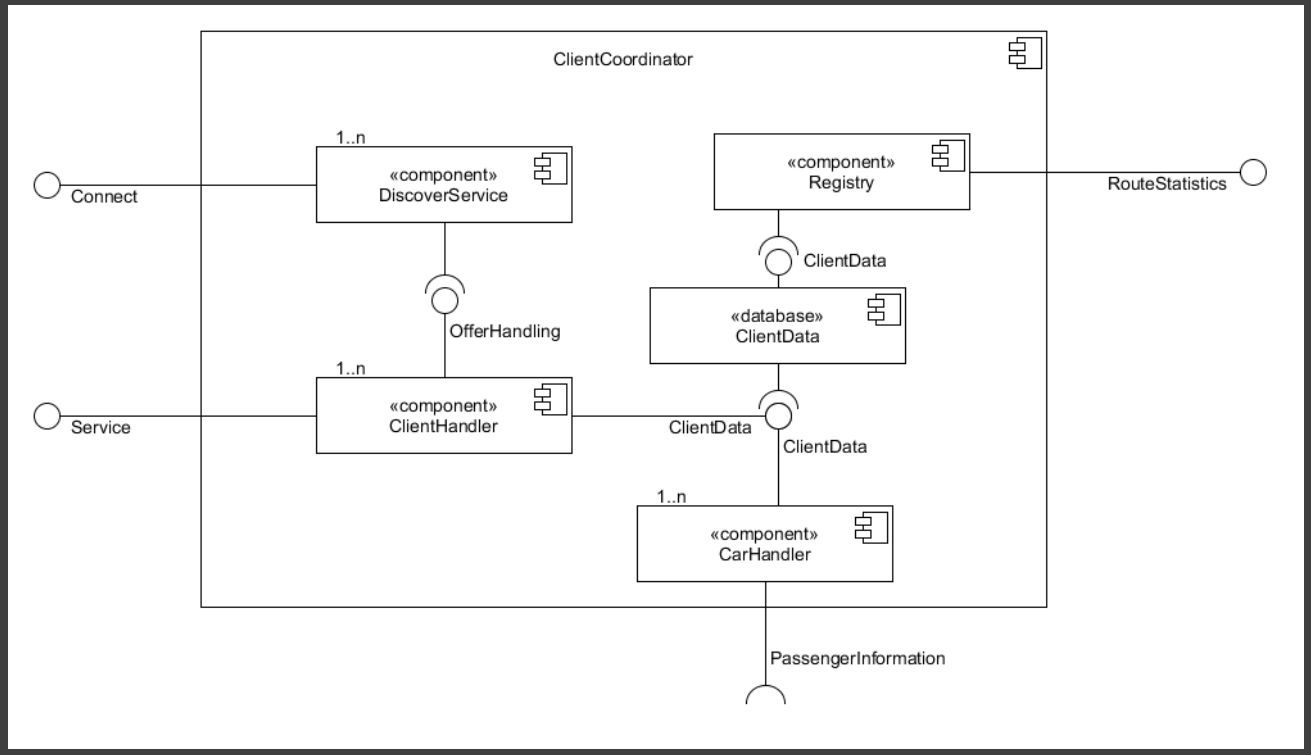
- ClientCoordinator:
  - Verwaltung von Nutzern
  - Sammeln der Nutzerzustände
  - Zusammenfassen der Nutzerdaten für Delegator
- Delegator
  - Erfährt Nutzerverteilung auf Routen vom ClientCoordinator
  - Erfährt Zustände der AutonomousVehicles
  - Fasst Daten in Statistik zusammen und teilt diese den AV mit
  - Kontrolliert Routenwechsel-Anfragen der AV, um Oszillation und Aushungern zu vermeiden



Modellierung -Komponenten

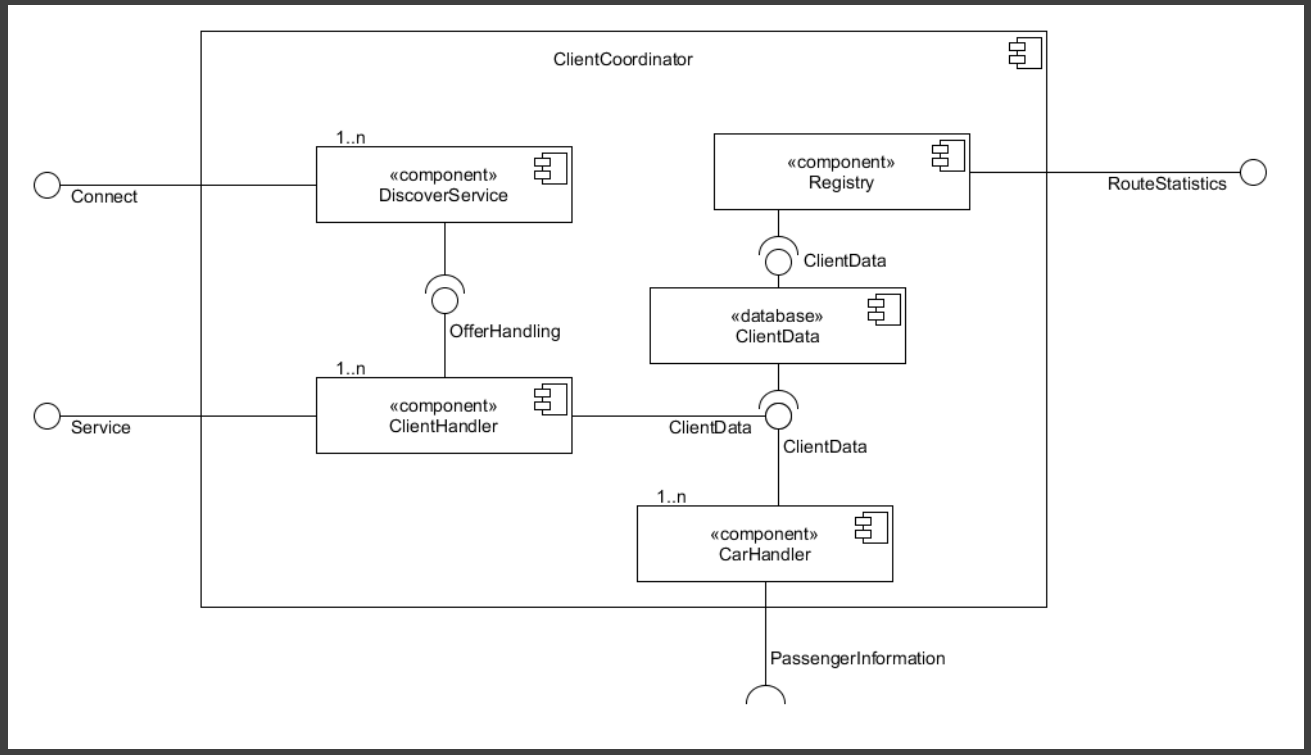
- Client:
  - Nutzer-Anwendung
  - Teilt sich ClientCoordinator mit und Schickt Service-Anfragen an diesen
  - Interagiert über direkte Kommunikation mit Fahrzeug
- AutonomousVehicle (AV)
  - Nimmt Nutzer auf und befördert sie
  - Teilt ClientCoordinator Mitfahrer-Daten mit
  - Teilt Delegator Zustand mit
  - Erfährt Routen-Statistiken von Delegator
  - Analysiert und prüft Statistiken auf mögliche Routenwechsel (Futterquelle)
  - Wechselt mit Einverständnis von Delegator Route, falls Kriterien erfüllt sind





Modellierung
-ClientCoordinator

- DiscoverService:
  - First-Discovery von Clients
  - Weiterleitung an ClientHandler
- ClientHandler:
  - Antwortet Clients
  - Empfängt Service-Anfragen und speist Client-Daten in Datenbank ein



Modellierung
-ClientCoordinator

- CarHandler:
  - Empfängt Client-Daten-Updates für Nutzer, die von AV eingesammelt oder abgesetzt wurden
- Registry:
  - Liest Client-Daten aus und bereitet Nutzer-Verteilung auf den Routen statistisch auf
  - Schickt regelmäßig Routen-Statistiken an Delegator
  - Archiviert abgelieferte Clients

Architektur
-Allgemein

- Umgesetzt in Java
- MQTT-Wrapper für alle Komponenten
  - Benutzt open-source hivemq-mqtt-client (1)
- Datenbank-Wrapper für alle Komponenten
- Definition von global gültigen Richtwerten und MQTT-Konstanten (Topics/Instructions)
- Client und AutonomousVehicle SM analog zu Uppaal
- Kommunikation angelehnt an Uppaal Channel (Deckung mit MQTT publish-subscribe)

(1) https://github.com/hivemq/hivemq-mqtt-client

Architektur
-AutonomousVehicle Adaptivität

- Routen hat num. Wert $cost$ in Abhängigkeit von Wartenden Clients c und AVs v auf der Route

$$cost = c/v$$

- $cost$ Bedeutung ähnlich Futterquellen: Je größer der Wert, desto attraktiver ist eine Route für den Wechsel

- Bedingung für Routenwechsel
  - Aktuelle Statistiken vorhanden
  - Kein Mitfahrer auf aktueller Route
  - Kein wartender Nutzer an der nächsten Station
  - $cost_{currentRoute} < cost_{otherRoute}$
  - $\frac{c}{v-1} < treshold \rightarrow$ Kosten meiner Route ohne mich kleiner als Schwellwert (experimentell ermittelt)

Architektur
-AutonomousVehicle Adaptivität

- Benötigte Distanz für Wechsel wird gegen *cost* gerechnet (erst ab einem Schwellwert z.B. 20km)
- Bei Erfüllung der Bedingungen Routenwechsel für Route mit höchster *cost* bei Delegator beantragen
- Bei Ablehnung nächsthöchste cost
- Anträge werden von Delegator zusammen mit Statistiken geprüft und bei positiv erwarteter Auswirkung angenommen
- Der Delegator achtet erneut darauf, dass bei einem Wechsel kein Aushungern der bisherigen Route die Folge ist

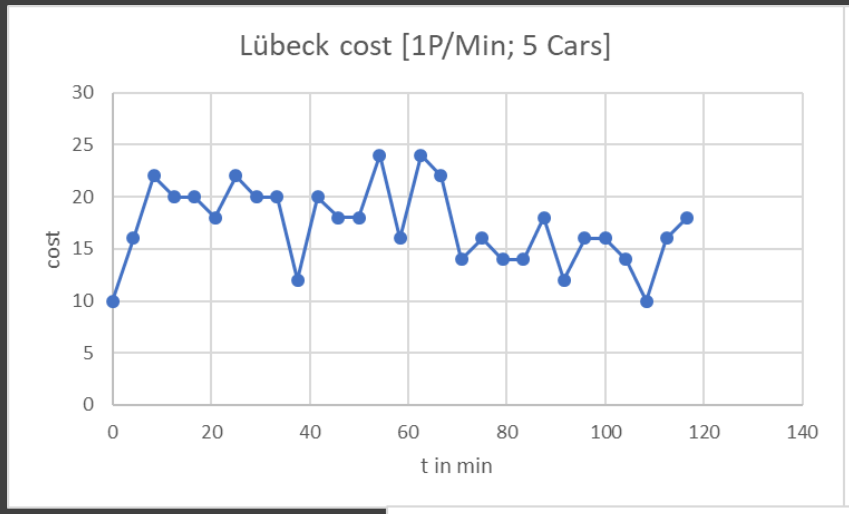
Testen
-Funktional

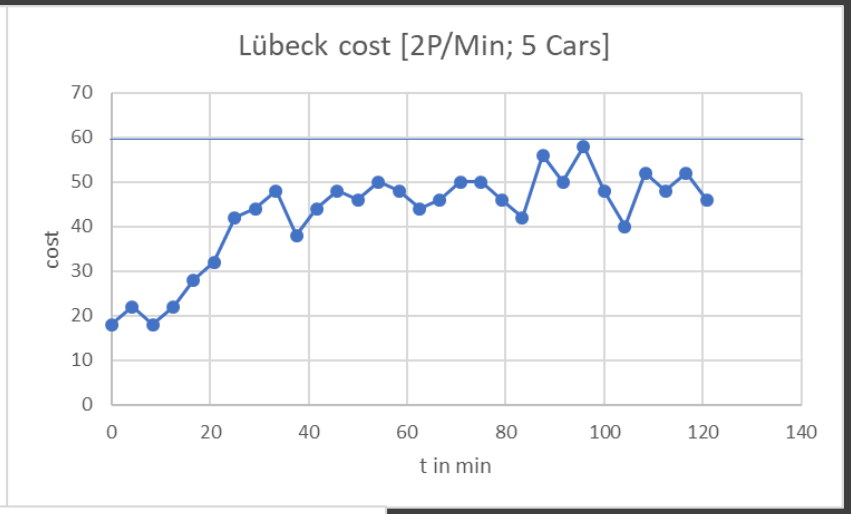
- Ausgabe von unerlaubten Kombinationen aus Zustand und IO
- Log Debugging (Wichtige Events, Zeitstempel)
- Ausgabe von gesammelten Daten (ClientCoordinator/Delegator) in Tabellen
  - Vergleich Anzahl instanziierte Komponenten mit beteiligten Komponenten

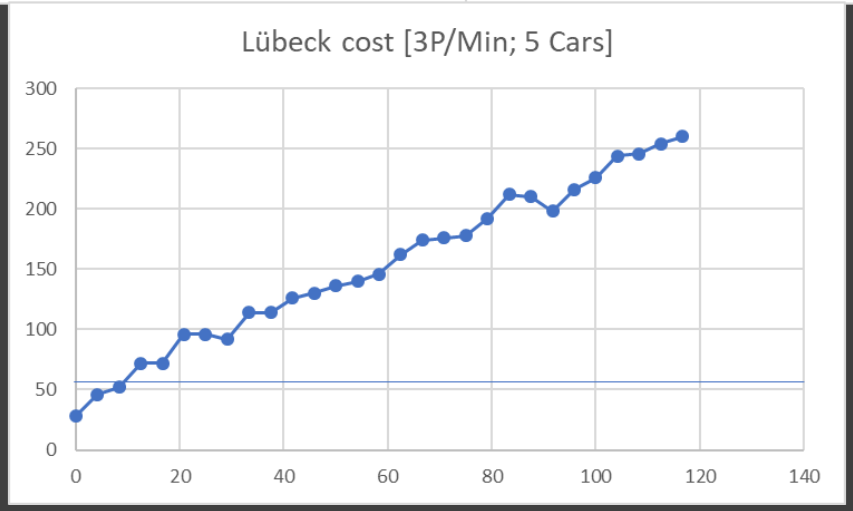
Testen
-Quantitativ

- Verschiedene Arten von Tests
  - Feste Anzahl an Teilnehmern
    - Randomisierte Verteilung
    - Faire Verteilung
    - Unfaire Verteilung
    - Unrealistische Verteilung
  - Variable Anzahl an Teilnehmern
    - Erzeugen von Passagieren über Zeit verteilt → Realität am nähesten

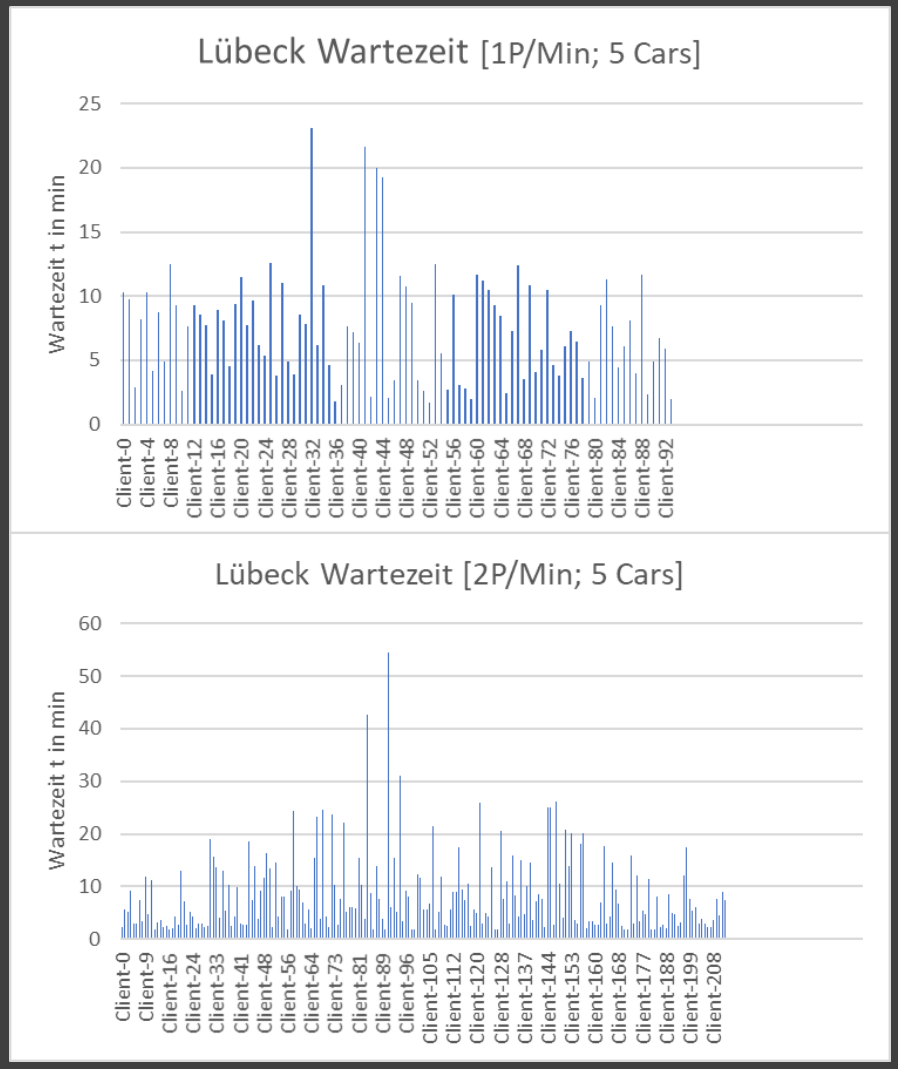
Testen
-Lübeck (kurze Entf.), 5 Autos



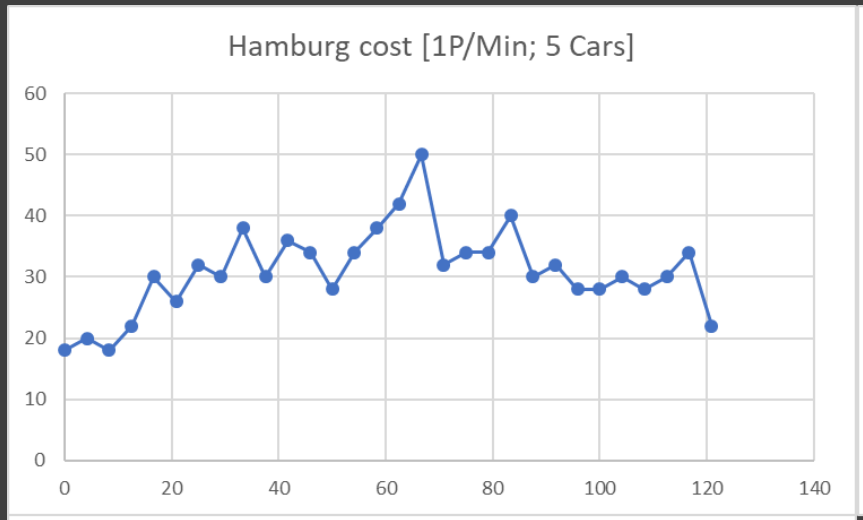


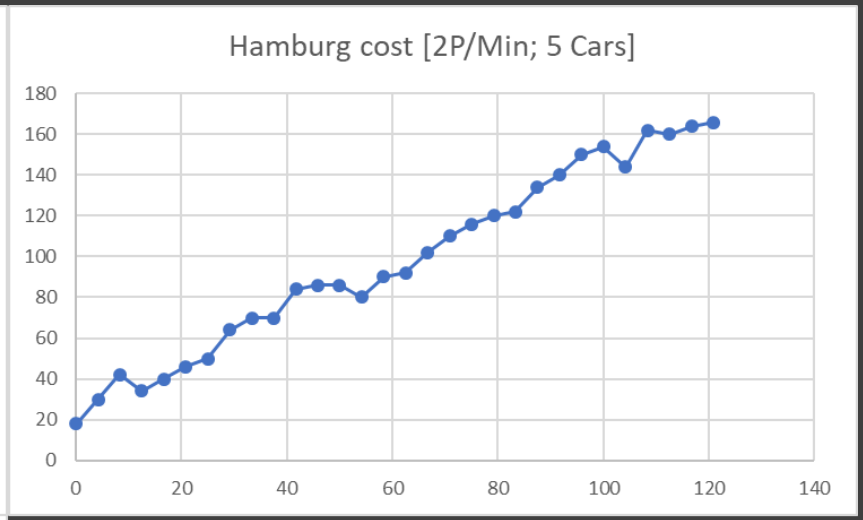


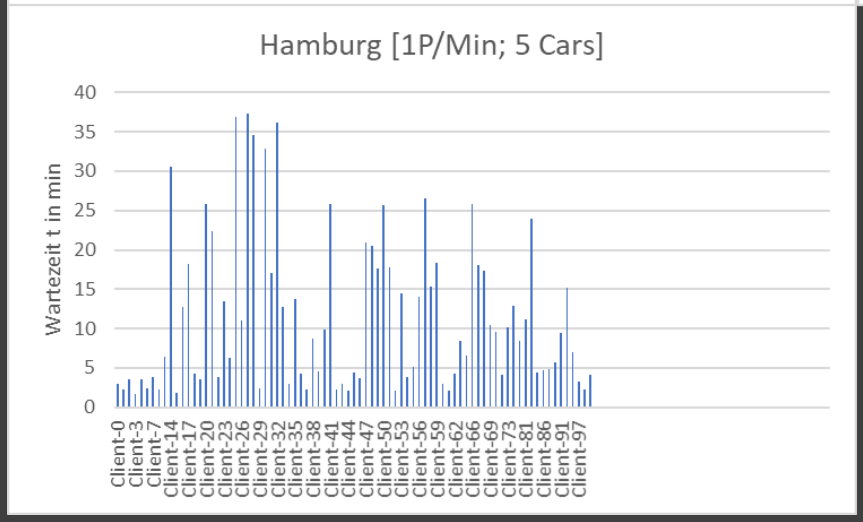
Testen
-Lübeck (kurze Entf.), 5 Autos



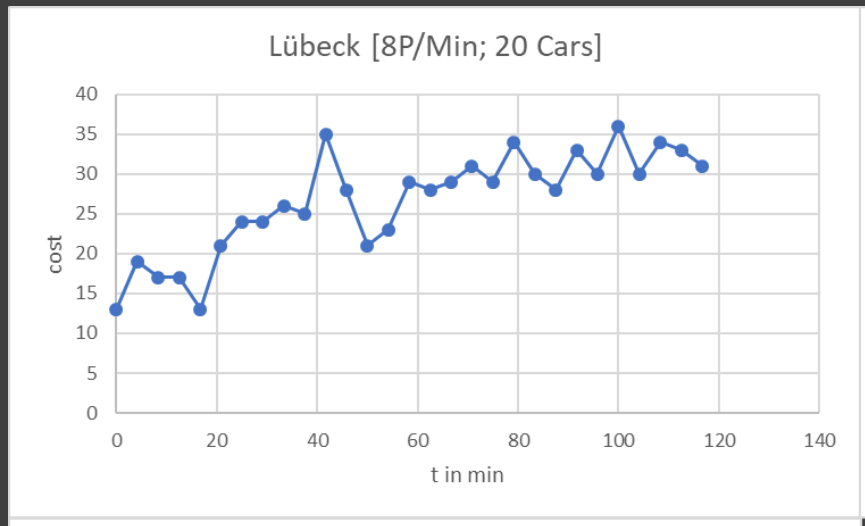
Testen
-Hamburg (große Entf.), 5 Autos

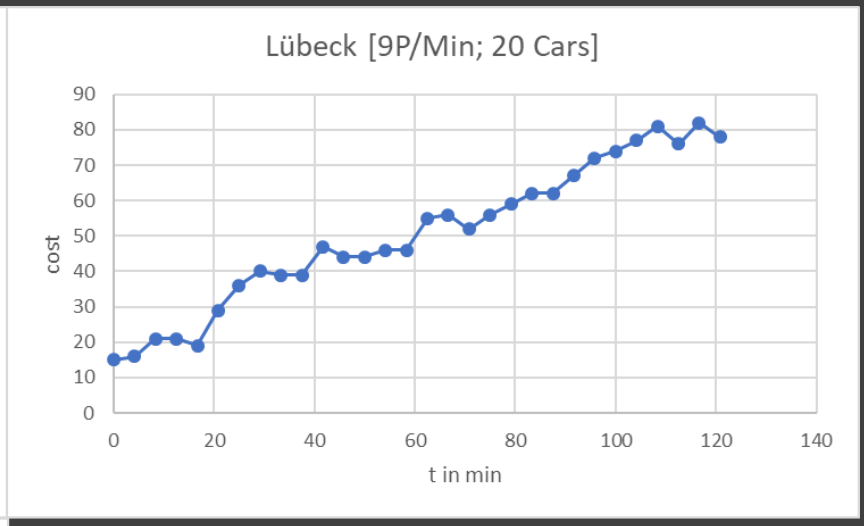


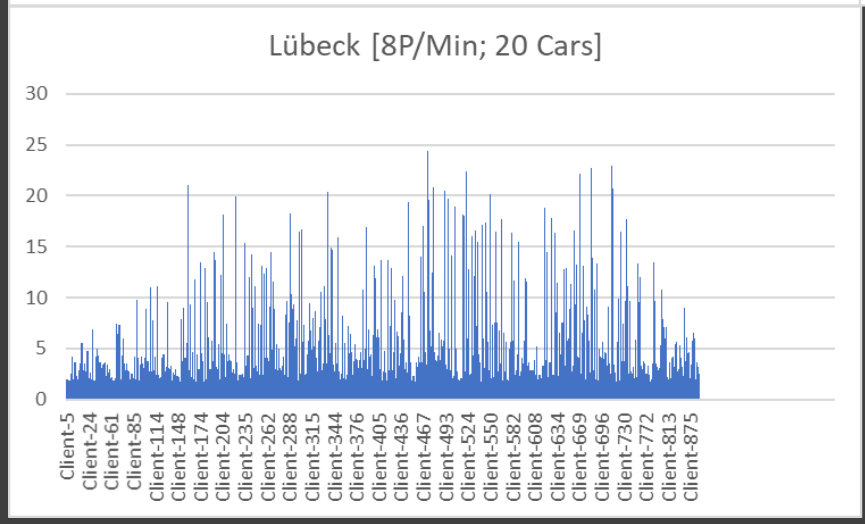




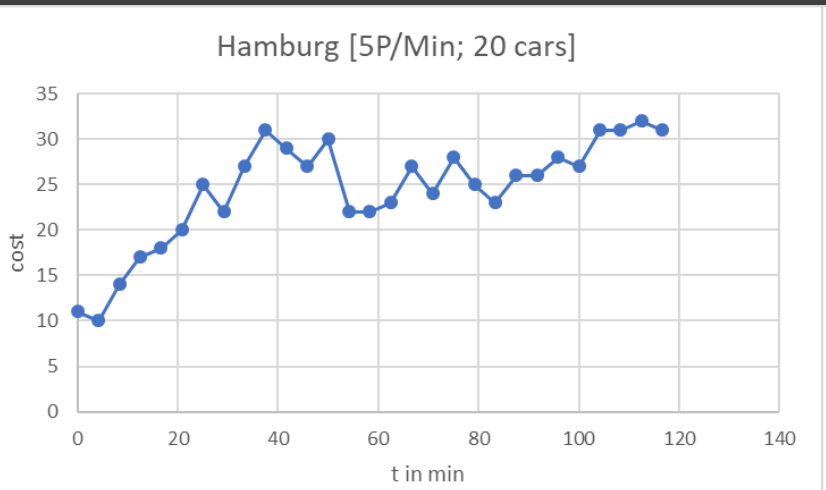
Testen
-Lübeck (kurze Entf.), 20 Autos

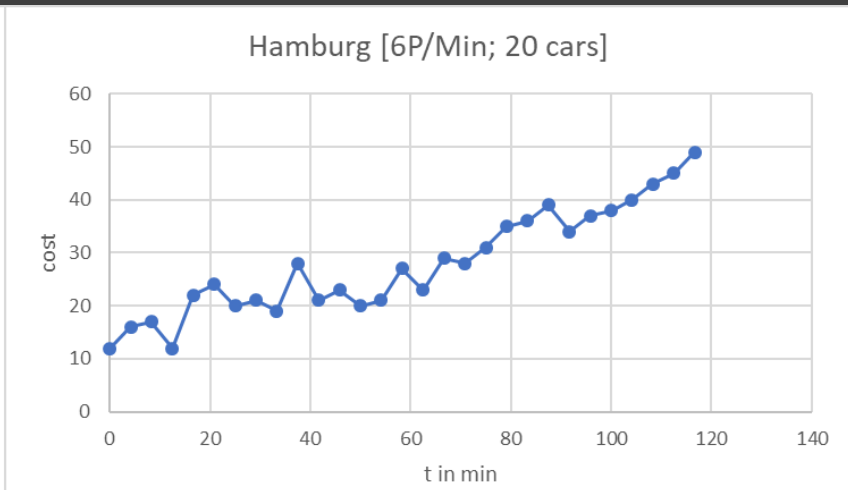


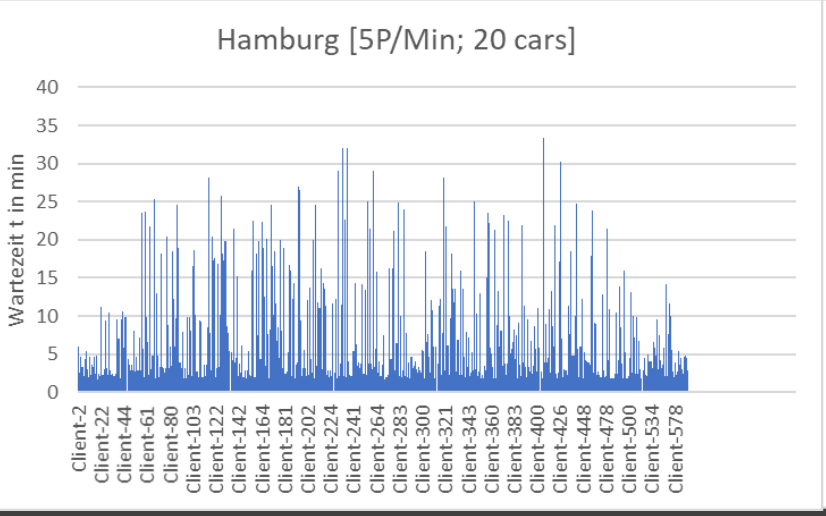


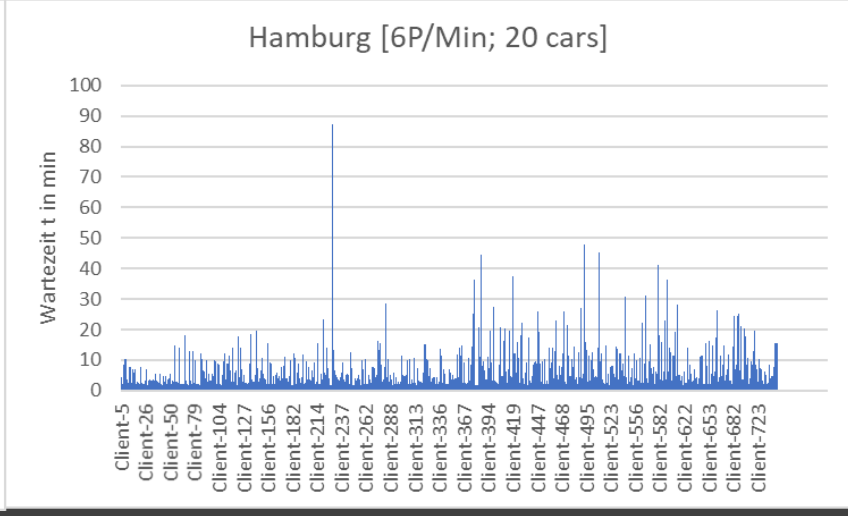


Testen
-Hamburg (große Entf.), 20 Autos









Testen
-Quantitativ Fazit

- Generell: Wachstum von cost unterbinden
  - AutonomousVehicles im Standby bereit halten
  - Routenwechsel
  - Schwellwert von cost auch in Abhängigkeit von Routen-Länge
- Großzügiges bereitstellen von AV um zu verhindern, dass Nutzer eine Runde wartet
- Noch viele weitere Faktoren möglich zum Einbeziehen in die Analyse und Adaption
  - Nutzer-Anmelde-Rate oder Routenlänge
  - Prognosen anhand von Events

Fazit
-Gesamt

- Idee: Beförderungs-Dienstleistung als Cyberphysisches System
  - Idee eignet sich als Cyberphysisches System
- Problem: Kann man schneller Personen befördern als öffentl. Verkehrsmittel und das günstiger als Taxis?
  - Umsetzung nach CPS-Schema erlaubt autonome Anpassung an Bedarf und damit potentiell geringere Wartezeiten als ÖPNV
  - Weglassen von Fahrzeugführern reicht bereits aus, um günstiger als Taxis fahren zu können

Evaluation

- Weitere Optimierungsmöglichkeiten:
  - Mehr Daten, mehr Analyse, mehr Faktoren für Adaptivität
  - Zu- und wegschalten von AutonomousVehicles
  - Reverse-Routen für kürzere Distanzen
    - z.B. Strecke A-B-C-D, Nutzer möchte von D nach C
  - Routen-los, d.h. Abholen und Abliefern an bel. Orten
    - Komplexität steigt stark an
- Was konnte man von Grund auf besser machen?
  - Effizientere Modellierung in Uppaal, da bereits bei mittlerer Anzahl Zeit zum verifizieren $\rightarrow \infty$
  - Mehr Zeit investieren, um o.g. Möglichkeiten mit zu implementieren

Evaluation

- Weitere Optimierungsmöglichkeiten:
  - Mehr Daten, mehr Analyse, mehr Faktoren für Adaptivität
  - Zu- und wegschalten von AutonomousVehicles
  - Routen-los, d.h. Abholen und Abliefern an bel. Orten
    - Komplexität steigt stark an
- Was konnte man von Grund auf besser machen?
  - Effizientere Modellierung in Uppaal, da bereits bei mittlerer Anzahl Zeit zum verifizieren $\rightarrow \infty$
  - Mehr Zeit investieren, um o.g. Möglichkeiten mit zu implementieren
  - Bessere Mechanismen zum Debuggen einrichten

Frage & Antwort