FH - Studiengang für Informationstechnik und System-Management Salzburg

ITS

Software Architektur Dokumentation einer Verkehrssimulation

Version: 1

Datum: 13.07.2017

Autoren: Stephanie Kaschnitz, Christopher Wieland, Martin Wieser, Andreas Lippmann, Hannes Kleiner

Unterschrift des Autors / der Autorin:

Tabellenverzeichnis

Abbildungsverzeichnis

Inhaltsverzeichnis

1 Einführung

Dieser Abschnitt führt die Aufgabenstellung ein und skizziert die Ziele, welche die Verkehrssimulation verfolgt.

1.1 Aufgabenstellung

Die Aufgabenstellung beinhaltet eine mikroskopische Simulation von Fahrzeuge, sowohl PKW als auch LKW, und Ampelanlagen. Ein weiterer wichtiger Punkt stellt das zusammenhängende Verkehrs-/Straßennetz dar. Weiters ist es möglich ungeregelte und geregelte Kreuzungen darzustellen. Ein wichtiger Faktor ist die Parametrisierbarkeit der Verkehrsteilnehmer und der Lichtsteueranlage.

Die Lichtsteueranlage wird über einen eigenen Prozess geregelt. Die verfügt die Applikation über eine grafische Darstellung sowie über eine einfache Benutzerschnittstelle.

In weiterer Folge sollen nun die Simulationen der verschiedenen Gruppen untereinander kommunizieren können. Das heißt, dass Fahrzeuge bei einer Straße einer Simulation ausfahren und dann in der anderen Simulation über eine Straße wieder einfahren können. Dem Benutzer wird ermöglicht beliebige Hindernisse auf der Straße per Mausklick ein-/auszuschalten.

1.2 Qualitätsziele

Die folgende Tabelle beschreibt die zentralen Qualitätsziele.

Qualitätsmerkmal	**Erläuterung**
Parametrisierbarkeit	Sowohl die Lichtsteueranlage als auch das Verhalten der Verkehrsteilnehmer sollen veränderbar sein.
Leicht bedienbar	Die Simulation und die dazugehörigen Einstellungen sind leicht einzustellen.
Effizienz	Die Berechnungen, welche im Hintergrund stattfinden, sollten rasch geschehen, ohne eine flüssige Simulation zu verhindern.

Tabelle 1.1: Qualitätsmerkmale

1.3 Stakeholder

Die folgende Tabelle stellt die Stakeholder dar.

Wer?	**Interesse und Bezug**
Einfache Benutzer	Möchten die Auswirkungen bezüglich der Ampelanlagen, Hindernissen und des Verhaltens der einzelnen Verkehrsteilnehmer beobachten.
Stadt/Land	Möchten die Auswirkungen bezüglich der Ampelanlagen, Hindernissen und des Verhaltens der einzelnen Verkehrsteilnehmer beobachten, um bei einem geplanten Bau/Umbau Staurisiken zu minimieren.

Tabelle 1.2:Stakeholder

2 Randbedingungen

Bei einem Lösungsentwurf waren zu Beginn verschiedene Randbedingungen zu beachten. Dieser Abschnitt behandelt die Technischen und Organisatorischen Randbedingungen.

2.1 Technische Randbedingungen

Randbedingung	**Erklärung**
Implementierung in C#	Die gesamte Simulation basiert auf der Programmiersprache C#. Mit dieser ist es möglich eine einfache grafische Oberfläche zu gestalten und den dazugehörigen Code zu implementieren.
Moderate Hardwareausstattung	Der Betrieb der Simulation sollte auf einem marktüblichen Standard-Notebook ausführbar sein.

Tabelle 2.1: Technische Rahmenbedingungen

2.2 Organisatorische Rahmenbedingungen

Rahmenbedingung	**Erklärung**
Team	Hannes Kleiner, Andreas Lippmann, Stephanie Kaschnitz, Christopher Wieland, Martin Wieser
Zeitplan	Beginn der Entwicklung: 23.03.2017 Erster lauffähiger Prototyp: 11.05.2017 Fertigstellung bis 22.06.2017
Vorgehensmodell	1. Entwicklung einer Softwarearchitektur 2. Entwicklung des Softwaredesigns 3. Implementierung der Software Zur Dokumentation der Architektur kommt arc42 zum Einsatz.
Entwicklungswerkzeuge	Erstentwurf mit Stift und Papier. Architektur in Enterprise Architect. Erstellung der C# Quelltexte in Visual Studio 2016.
Versionsverwaltung	Git

Tabelle 2.2: Organisatorische Rahmenbedingungen

3 Kontext

Dieser Abschnitt beschreibt das Umfeld. Wie interagieren die Benutzer und mit welchen Fremdsystemen interagiert es?

3.1 Fachlicher Kontext

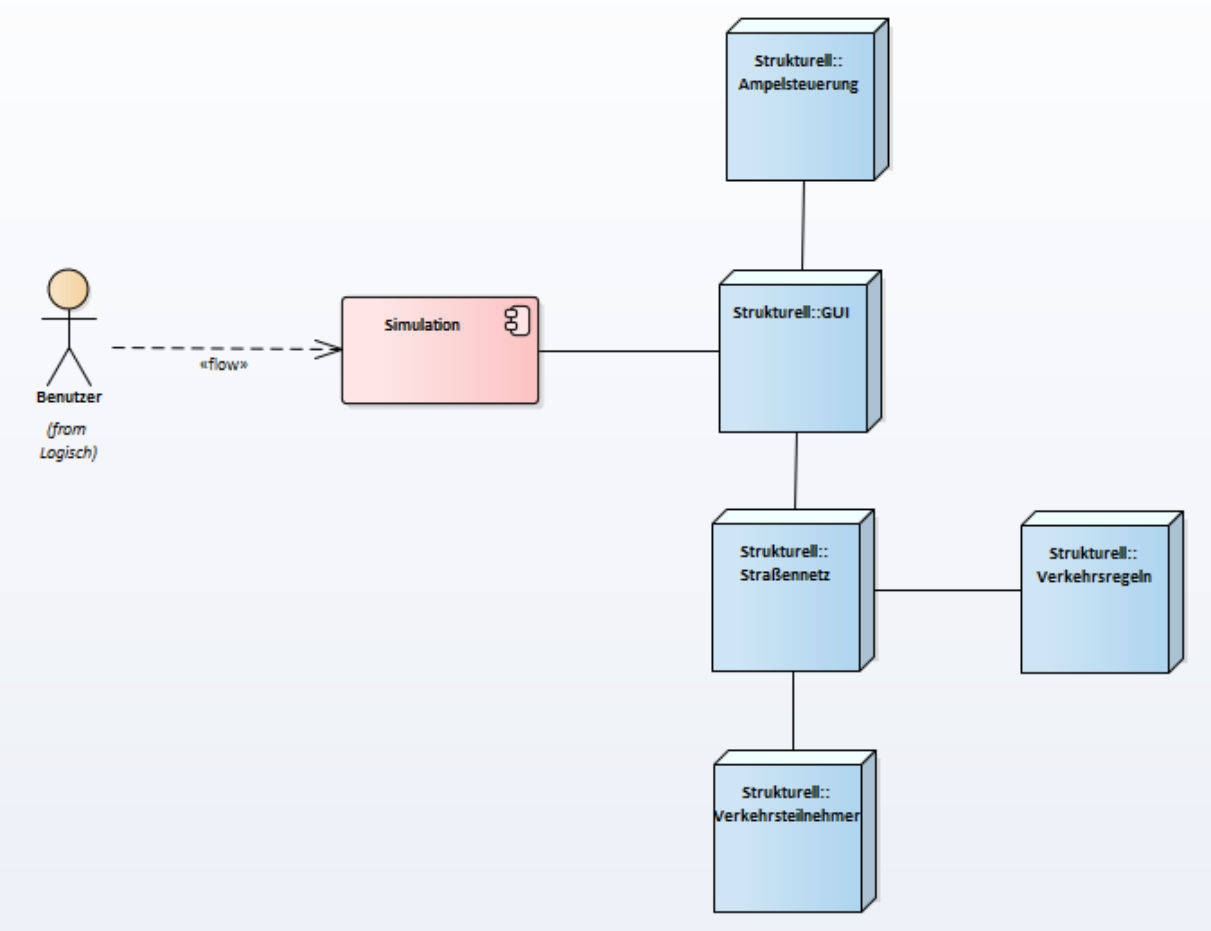

Abbildung 3.1: Kontext

Benutzer

Ein Benutzer kann die Simulation starten und verschiedene Werte einstellen, wie zum Beispiel das Verhalten der Verkehrsteilnehmer oder die Ampelsteuerung.

GUI

Das Grafical User Interface (= GUI) behandelt die graphische Darstellung aller Elemente der Verkehrssimulation. Dabei werden zum Beispiel Ampeln, sich bewegende Autos und Straßen dem Benutzer angezeigt.

Ampelsteuerung (Eigener Prozess)

Bei der Ampelsteuerung handelt es sich um einen eigenen Prozess, welcher die verschiedenen Phasen (grün, gelb, rot) einer Ampel regelt.

Verkehrsregeln

Die Verkehrsregeln beinhalten allgemeine Regeln, welche bei ungeregelten und geregelten Kreuzungen eingehalten werden müssen.

Straßennetz

Das Straßennetz behandelt die Anzahl der Kreuzungen und der Ampeln, sowie kümmert es sich um die Verbindung der Kreuzungen mit Straßen. Auf dem Straßennetz werden die fahrenden Autos abgebildet.

Verkehrsteilnehmer

Unter Verkehrsteilnehmer versteht man LKWs und PKWs. Darunter fällt auch deren Verhalten, welches vom Benutzer veränderbar ist.

4 Lösungsstrategie

Die folgende Tabelle stellt die Qualitätsziele von der Verkehrssimulation passenden Architekturansätzen gegenüber.

Qualitätsziel	**Ansatz in der Architektur**
Parametrisierbarkeit	Schnittstellen für die Veränderungen des Verhaltens der Verkehrsteilnehmer und der Ampeln. Geschieht über Json-Files.
Leicht bedienbar	Beim Start der Applikation öffnet sich zu der GUI eine Konsole, in der das momentane Verhalten der Verkehrsteilnehmer per Mausklick angezeigt und geändert werden kann.
Effizienz	Lesen/Schreiben von Json-Files benötigt wenig Zeit.

Tabelle 4.1: Lösungsstrategie

4.1 Aufbau der Verkehrssimulation

Bei der Verkehrssimulation handelt es sich um ein C#-WPF-Programm, wobei der grafische Teil in xaml geschrieben und der benötigte Code für die Simulation in der dazugehörigen Main-Routine realisiert ist.

Der Code zerfällt in folgende Teile:

- Straßennetz
- Verkehrsregeln
- Ampelsteuerung
- GUI
- Verkehrsteilnehmer

Die Zerlegung ermöglicht es, einzelne Sachen einfach auszutauschen und in das Programm einzugliedern. Alle Teile sind durch Schnittstellen abstrahiert.

5 Bausteinschicht

Dieser Abschnitt beschreibt die Zerlegung von der Simulation in Module, wie sie sich auch in der Struktur widerspiegelt. Dieses Kapitel beschreibt folgende Modelle: Computational Independent Model (=CIM) und dem Platform Independent Model (=PIM).

5.1 Computational Independent Model

Die nachfolgende Abbildung beschreibt die Gesamtübersicht, bestehend aus der Ampelsteuerung und der Simulation, und deren Interfaces. Hierbei handelt es sich um zwei unabhängige Systeme. Die Ampelsteuerung ist ein eigenständiges Programm, welches mittels Interfaces mit der Simulation kommuniziert.

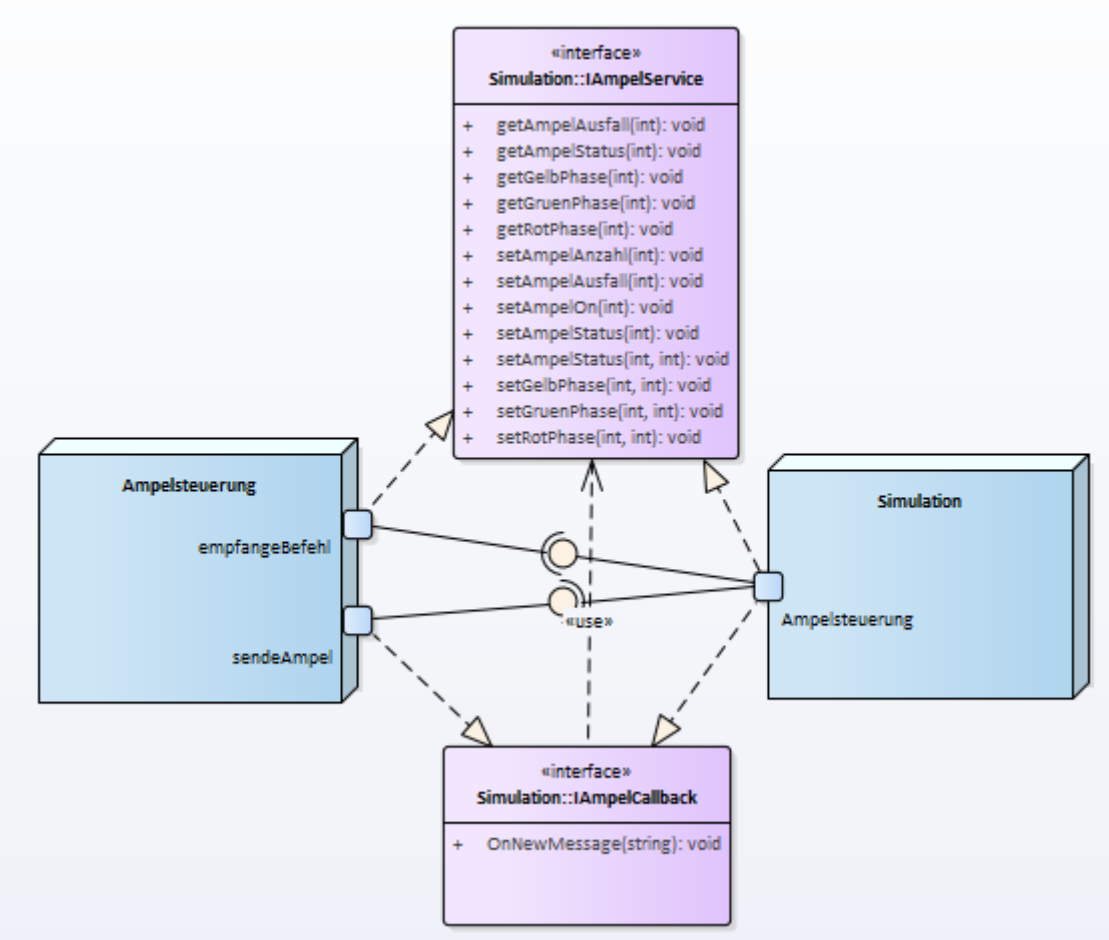

Abbildung 5.1: Gesamtübersicht (CIM)

5.2 Platform Independent Model

Dieses Kapitel zeigt die PIM der Simulation und der Ampelsteuerung.

5.2.1 PIM Simulation

Die nachfolgende Abbildung beschreibt das Platform Specific Model der Simulation. Hierbei werden die Interfaces und Komponenten ersichtlich.

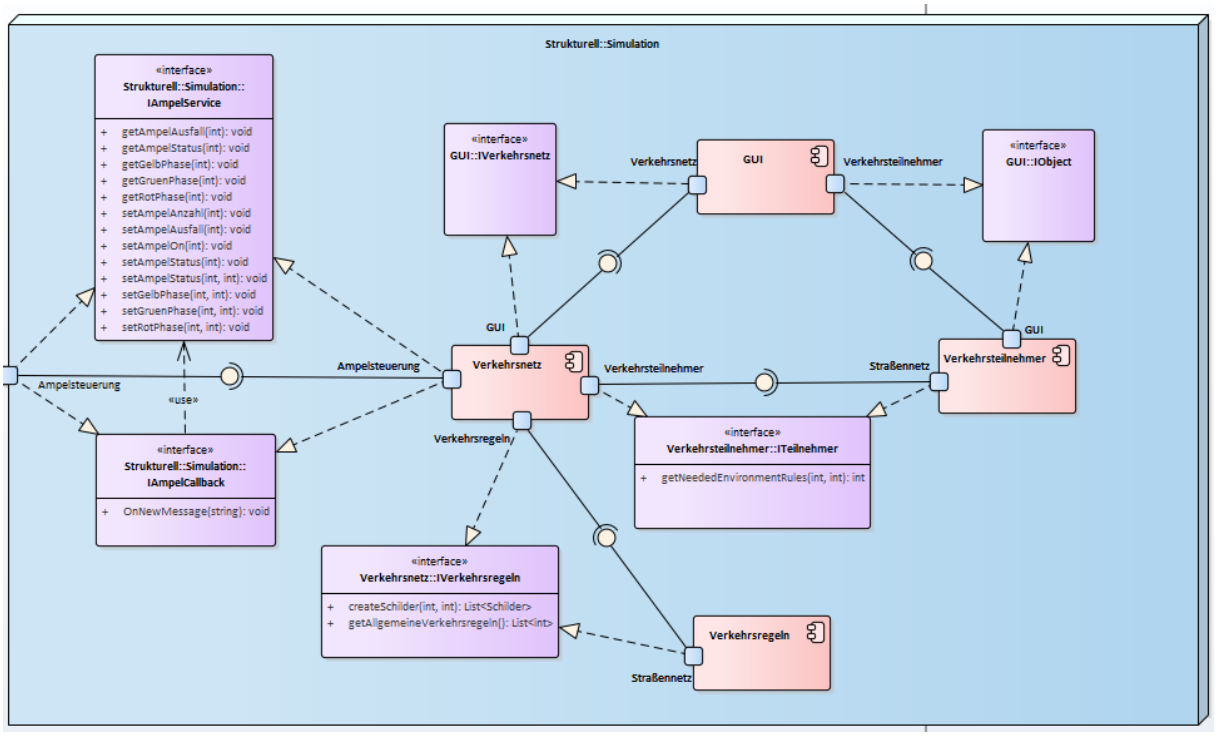

Abbildung 5.2: PIM Simulation

Die Simulation beinhaltet folgende Komponenten: Verkehrsnetz, Verkehrsregeln, GUI und Verkehrsteilnehmer. Jeder dieser Komponenten kommuniziert mit der darüberstehenden Komponente mittels Interfaces.

5.2.1.1 Zweck des Verkehrsnetzes

Das Verkehrsnetz behandelt die Anzahl der Kreuzungen und der Ampeln, sowie kümmert es sich um die Verbindung der Kreuzungen mit Straßen. Auf dem Verkehrsnetz werden die fahrenden Autos abgebildet.

<u>Schnittstellen:</u>

Das Verkehrsnetz bietet die Schnittstelle ITeilnehmer, welches von dem Verkehrsteilnehmer Modul verwendet wird, an.

Methode	**Kurzbeschreibung**
getNeededEnvironmentRules	Liefert den Straßentyp zurück. Die Parameter hierbei sind zwei Integer-Werte.
getNeededStreetRules	Liefert die Regeln zu den mitgelieferten Parametern, zwei Integer-Werte, welche die Position beschreiben, zurück.
getObstracles	Liefert eine Liste mit den Hindernissen zurück.

getEnvironmentEntries	Liefert die Einträge der Umgebung in einer Liste zurück.

Tabelle 5.1: ITeilnehmer Schnittstellenbeschreibung

5.2.1.2 Zweck der GUI

Das Grafical User Interface (= GUI) behandelt die graphische Darstellung aller Elemente der Verkehrssimulation. Dabei werden zum Beispiel Ampeln, sich bewegende Autos und Straßen dem Benutzer angezeigt.

Schnittstellen:

Die GUI bietet folgende Schnittstellen an: IObject und IVerkehrsnetz.

Methode	**Kurzbeschreibung**
addCarObject	Rückgabewert ist ein Boolean und die Parameter bestehen aus drei Integer-Werten. Die Funktion der Methode beinhaltet die Erstellung eines Autos in der GUI.
addLKWObject	Diese Methode lässt in der GUI einen LKW erstellen.
updateCarWithID	Diese Methode wird verwendet um die Position eines Autos in der GUI anhand seiner ID zu aktualisieren.

Tabelle 5.2: IObject Schnittstellenbeschreibung

Methode	**Kurzbeschreibung**
objshp	Setzt und Erhält ein Environment Element.
greenLight	Setzt und Erhält eine grüne Ampel
redLight	Setzt und Erhält eine rote Ampel
Objid	Setzt und Erhält die Objekt-ID
Xpos	Setzt und Erhält die X-Position auf der GUI
Ypos	Setzt und Erhält die Y-Position auf der GUI

Tabelle 5.3: IVerkehrsnetz Schnittstellenbeschreibung

5.2.1.3 Zweck der Verkehrsregeln

Die Verkehrsregeln beinhalten allgemeine Regeln, welche bei ungeregelten und geregelten Kreuzungen eingehalten werden müssen.

Schnittstellen:

Die Verkehrsregeln bieten dem Verkehrsnetz die Schnittstelle IVerkehrsregeln an.

Methode	**Kurzbeschreibung**
createSchilder	Liefert eine Liste an Schildern zurück, je nach mitgelieferter Anzahl und Typ der Schilder.
getAllgmeineVerkehrsregeln	Liefert eine Liste mit den Allgemeinen Verkehrsregeln zurück.

Tabelle 5.4: IVerkehrsregeln Schnittstellenbeschreibung

5.2.1.4 Zweck der Verkehrsteilnehmer

Unter Verkehrsteilnehmer versteht man LKWs und PKWs. Darunter fällt auch deren Verhalten, welches vom Benutzer veränderbar ist.

Schnittstellen:

Das Modul der Verkehrsteilnehmer bietet die Schnittstellen IGUI und ITrafficHandler an.

Methode	**Kurzbeschreibung**
updateCarAmount	Diese Methode aktualisiert die Anzahl der Autos
updateTruckRatio	Diese Methode aktualisiert die Wahrscheinlichkeit, mit welcher ein LKW erstellt wird.

Tabelle 5.5: IGUI Schnittstellenbeschreibung

Methode	**Kurzbeschreibung**
updateAll	Aktualisiert die Autos und LKWs
createNewVerkehrsteilnehmer	Diese Methode erstellt einen neuen Verkehrsteilnehmer
removeVerkehrsteilnehmer	Mit Hilfe dieser Methode werden Verkehrsteilnehmer gelöscht.

Tabelle 5.6: ITrafficHandler Schnittstellenbeschreibung

5.2.2 PIM Ampelsteuerung

Die nachfolgende Abbildung beschreibt das Platform Specific Model der Ampelsteuerung. Hierbei werden die Interfaces und Komponenten ersichtlich.

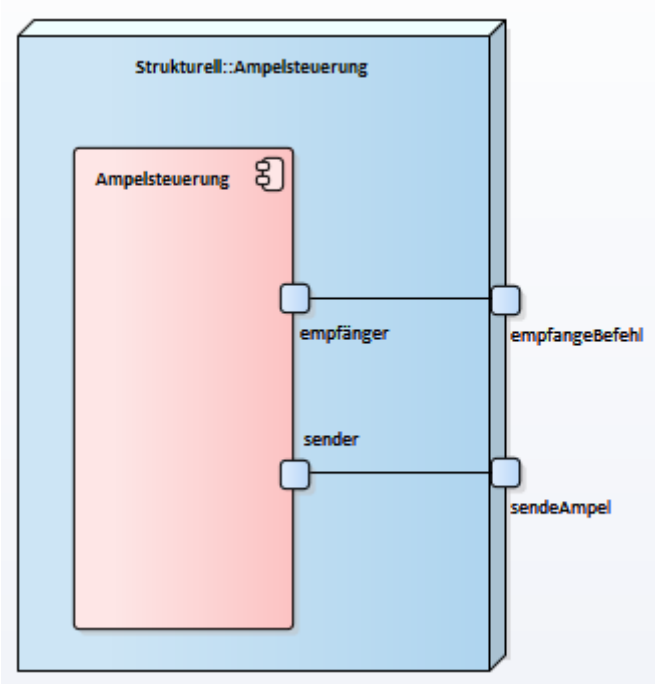

Tabelle 5.7: PIM Ampelsteuerung

Die Ampelsteuerung besteht aus dem Modul Ampelsteuerung. Der Zweck der Ampelsteuerung ist eine gewisse Anzahl an Ampeln zu generieren und diese je nach beliebigen Einstellungen zu schalten.

Schnittstellen:

Die Ampelsteuerung stellt insgesamt zwei Schnittstellen zur Verfügung: IAmpelCallback und IAmpelService.

Methoden	**Kurzbeschreibung**
OnNewMessage	Nimmt einen String Parameter entgegen

Tabelle 5.8: IAmpelCallback Schnittstellenbeschreibung

Methoden	**Kurzbeschreibung**
setAmpelAnzahl	Erstellt die mitgegebene Anzahl an Ampeln
getAmpelStatus	Liefert den Status der Ampel zurück
getAmpelAusfall	Gibt an, ob die Ampel funktioniert oder ausgeschaltet ist
setAmpelAusfall	Schaltet die Ampel, welche mittels der ID identifiziert wird, aus
setAmpelOn	Schaltet die Ampel, welche mittels der ID identifiziert wird, ein.
setAmpelStatus	Setzt den Status der Ampel
setRotPhase	Setzt die Rotphase einer einzelnen Ampel
setGelbPhase	Setzt die Gelbphase einer einzelnen Ampel
setGruenPhase	Setzt die Grünphase einer einzelnen Ampel
getRotPhase	Gibt die Sekunden der Rotphase zurück
getGelbPhase	Gibt die Sekunden der Gelbphase zurück
getGruenPhase	Gibt die Sekunden der Grünphase zurück
getAmpelAnzahl	Gibt die Anzahl der Ampeln zurück

Tabelle 5.9: IAmpelService Schnittstellenbeschreibung

6 Laufzeitschicht

Diese Schicht visualisiert im Gegensatz zur statischen Bausteinsicht dynamische Aspekte und zeigt, wie die einzelnen Teile zusammenspielen.

Sequenzdiagramm der Simulation

Wie in der nachfolgenden Abbildung ersichtlich wird, wird zuerst die GUI gestartet, wenn die Simulation ausgeführt wird. Im nächsten Schritt werden die Verkehrsnetzinformationen bei dem Verkehrsnetz abgefragt, dazu werden zuerst die Verkehrsregeln abgefragt und gesendet. Im Anschluss werden die Ampelinformationen abgefragt und erhalten. Im nächsten Schritt werden die Verkehrsteilnehmer die Straßeninformationen zugeschickt. Darauf werden die Verkehrsteilnehmer an die GUI gesendet. Außerdem erhält die GUI die Verkehrsnetzinformationen aus dem Verkehrsnetz.

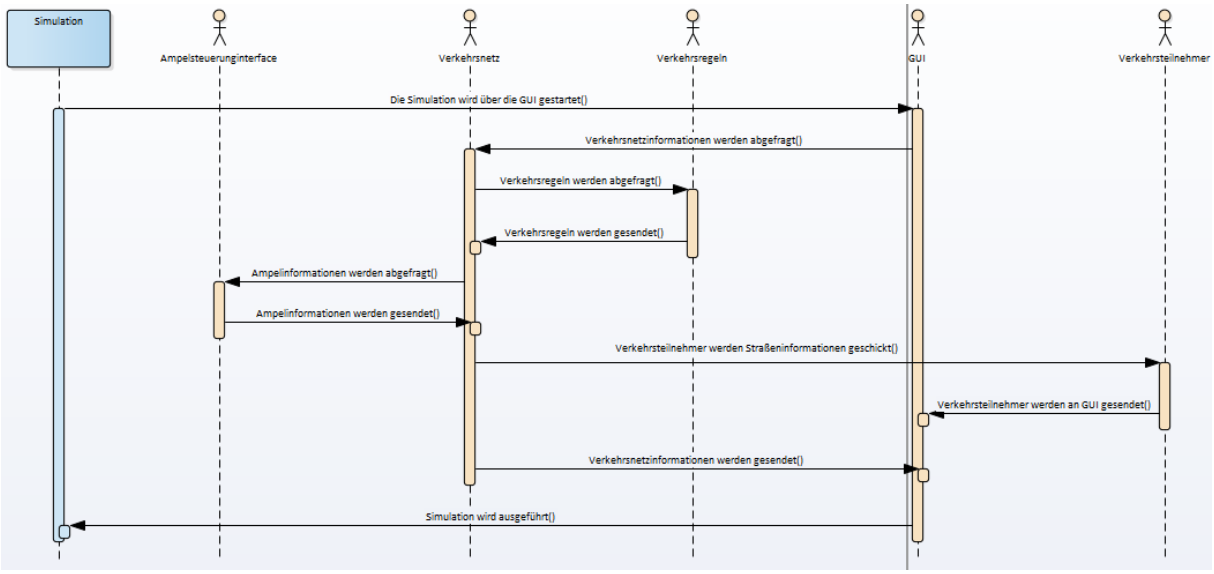

Tabelle 6.1: Sequenzdiagramm Simulation

Sequenzdiagramm der Ampelsteuerung

In der nachfolgenden Abbildung wird das Vorgehen der Ampelsteuerung ersichtlich. Das Ampelsteuerungsinterface fragt nach bestimmten Ampelstatus der Ampelsteuerung ab. Diese sendet den jeweiligen Ampelstatus zurück an das Interface.

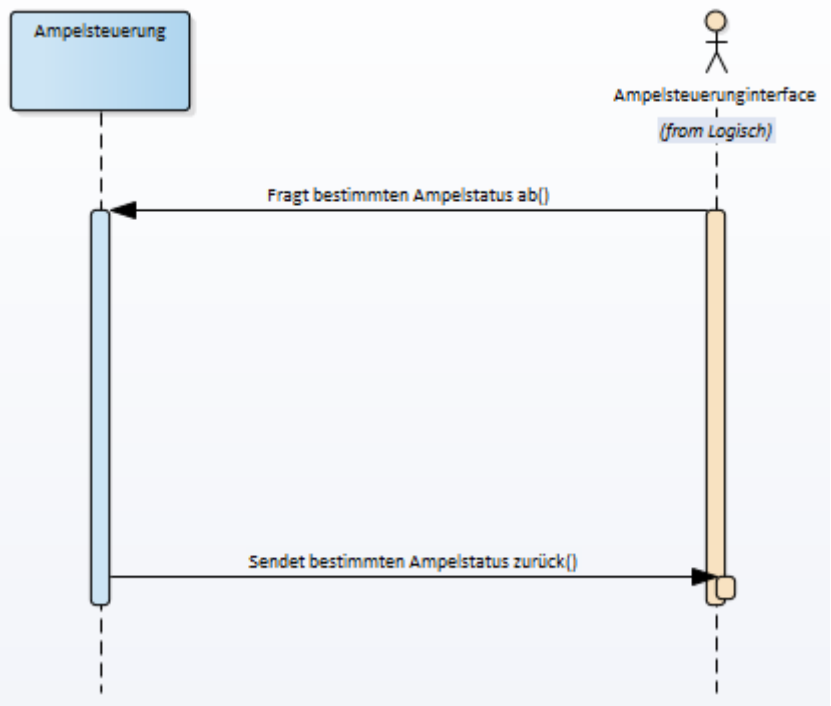

Tabelle 6.2: Sequenzdiagramm Ampelsteuerung

7 Konzepte

Dieser Abschnitt beschreibt die allgemeinen Strukturen und Aspekte, die systemweit gelten.

7.1 Benutzeroberfläche

Dieses Kapitel zeigt die Benutzeroberfläche der Simulation. Die Autos werden als kleine blaue Punkte und die LKWs als zwei lila Punkte dargestellt. Auf der rechten Seite befindet sich die Konfiguration, wobei die Geschwindigkeit, Anzahl der LKWs und die Anzahl an Fahrzeugen während der Laufzeit angepasst werden kann.

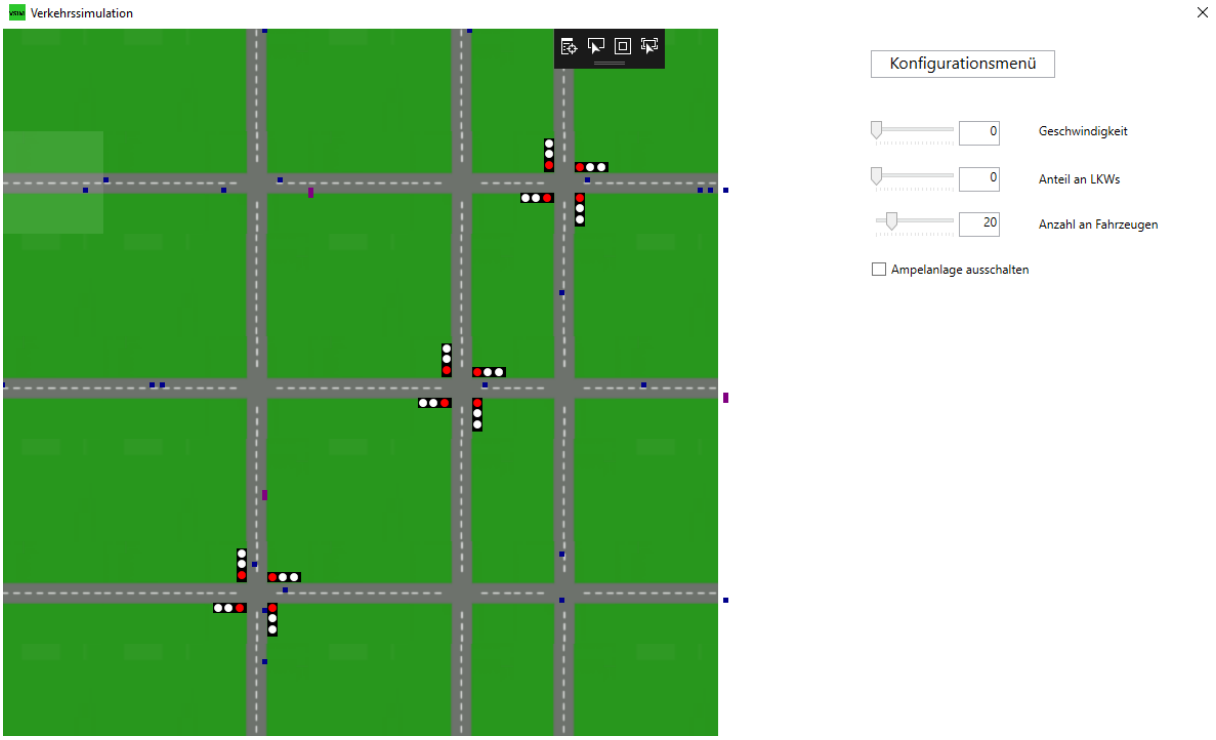

Tabelle 7.1: Benutzeroberfläche

8 Entscheidungen

Dieser Abschnitt befasst sich mit den Entscheidungen, welche im Team getroffen wurden.

8.1 Graphische Darstellung

Die Fragestellung lautete, wie die graphische Darstellung funktionieren sollte. Ob man auf ein Fremdsystem wie zum Beispiel Unity zurückgreifen sollte.

Nachdem die Aufgabenstellung beinhalt, dass die gesamte Simulation in C# und Visual Studio zu implementieren ist, fiel die Entscheidung auf eine WPF-Applikation. Die WPF-Applikation bietet zu der Main eine grafische Oberfläche, welche einfach mit XAML zu programmieren ist. Geeinigt haben wir uns auf ein Koordinatensystem und Canvas.

8.2 Kommunikation zwischen den Gruppen

Hierbei lautete die Fragestellung wie eine Kommunikation zwischen den einzelnen Simulationen stattfinden kann. Nachdem sich die Team-Mitglieder geeinigt haben, fiel das Ergebnis auf die Benutzung von RabbitMQ. Hierbei sendet jede Gruppe ihre Autos über ein zuvor allgemein definiertes Protokoll an eine andere Gruppe, diese bearbeitet die Requests und fügt die Autos an die dementsprechenden Stellen ein.

9 Risiken

Dieser Abschnitt befasst sich mit den Risiken, welche zu Beginn des Projektes identifiziert wurden.

9.1 Ausfall der Ampelsteuerung

Ein Risiko stellt der Ausfall der Ampelsteuerung dar. Da die Ampelsteuerung ein eigenständiges Programm ist, stellt es ein Risiko dar, wenn dieses Programm ausfällt.

Risikominderung:

Das Risiko kann beseitigt werden indem die Autos bei Ausfall der Ampeln die allgemeinen Verkehrsregeln beachten. Somit wird keine Rücksicht mehr auf die Ampeln gegeben und die Simulation läuft ohne weitere Probleme weiter.