Projektname:	Traffic Simulation
Projektnummer:	0002
Kurzbeschreibung:	Im Kapitel 1.1.1
Version:	1.1
Vorgelegt von:	Reimar Klammer, Daniel Komohorov, Gerold Katzinger, Tobias Mayer
Author:	Daniel Komohorov
Auftraggeber:	DI (FH) DI Roland Graf, MSc / DI Eduard Hirsch
Erstelldatum:	05.04.2017

DOKUMENTENHISTORIE

Autor	Datum	Version	Änderung
R. Klammer	05.04.17	1.0	Erstentwurf der Architektur
D. Komohorov	11.04.17	1.1	Erstellung der Dokumentenstruktur
•	•	•	•
•	•	•	•
•	•	•	•
•	•	•	•

I Inhaltsverzeichnis

1 Softwarearchitektur

1.1 Einführung und Ziele

1.1.1 Aufgabenstellung

Beim Projekt "Traffic Sumulation"handelt es sich um ein FH-Projekt im Zuge der Lehrveranstaltng Softwarearchitektur. Die wesentliche Aufgabenstellung ist Entwurf, Dokumentation und Implementierung einer verteilten komponentenbasierten Verkehrsimulation gemäß des Anforderungsdokuments (Kapitel 5.1).

Die Simulation sollte wie folgt unterstützen:

- Mikroskopische Simulation der Fahrzeuge (PKW und LKW) und Ampelanlagen
- Zusammenhängendes Verkehrs-/Straßennetz
- Geregelte und ungeregelte Kreuzungen
- Verhalten der Verkehrsteilnehmer sollte weitgehend parametrisierbar sein (variabel/-zufällig/individuell)
- Verhalten der Lichtsteueranlage parametrisierbar Regelung über eigenen Prozess
- Anzahl der über Einfahrtsstraßen in das System zufahrenden Fahrzeuge regelbar
- Grafische Darstellung und einfache Benutzerschnittstelle Konfiguration-GUI nicht unbedingt notwendig

1.1.2 Qualitätsziele

Tabelle 1: Qualitätsziele

Qualitätsmerkmal	**Beschreibung**
agiler Softwareentwicklungsprozess	- reine Entwurfsphase auf ein Mindestmaß zu reduzieren - im Entwicklungsprozess so früh wie möglich zu ausführbarer Software zu gelangen - in regelmäßigen, kurzen Abständen dem Kunden zur gemeinsamen Abstimmung die Ergebnisse für weitere Abstimmung vorzeigen
komponentenbasierte Entwicklung	mehrere logische Komponente und kein Monolit
Anpassbarkeit	In der Hinsicht auf mögliche Veränderungen oder Erweiterungen müssen die Komponenten zu einem gewissen Teil eine einfache Anpassung ohne großen Aufwand "vertragen"
Testbarkeit	Die einzelnen Komponenten können getrennt voneinander getestet werden.

1.1.3 Stakeholder

Tabelle 2: Stakeholder

Rolle	Kontakt	Erwartungshaltung
Roland Graf (Auftraggeber)	roland.graf@fh-salzburg.ac.at	- Fertigstellung des Projekts nach Anforderungsdokument - Professionelle Architektur und Designdokumente
Eduard Hirsch(Auftraggeber)	eduard.hirsch@fh-salzburg.ac.at	- Fertigstellung des Projekts nach Anforderungsdokument - Professionelle Architektur und Designdokumente
Teammitglieder	gkatzinger.itsb-m2016@fh-salzburg.ac.at tmayer.itsb-m2016@fh-salzburg.ac.at iklammer.itsb-m2016@fh-salzburg.ac.at dkomohorov.itsb-m2016@fh-salzburg.ac.at	- Beschreibung Softwarearchitektur - Leichte Umsetzbarkeit - Gut testbare Komponenten

1.2 Randbedingungen

1.2.1 Technische Rahmenbedingungen

- C# / .NET

- Visual Studio 2015 und höher

Alternativ kann zur Entwicklung auch C#/Mono verwendet werden. Die Abgabe am Ende des Semesters hat aber ausnahmslos als Visual Studio Solution zu erfolgen!

1.2.2 Organisatorische Rahmenbedingungen

Verteilte Zusammenarbeit

Das Team schafft entsprechende Rahmenbedingungen, dass einzelne Teammitglieder ihre Arbeitspakete von einander unabhängig fertigstellen können.

Vorgehensmodell

Die Entwicklung erfolgt iterativ und agil. Offene Punkte werden in Paketen nach Scrum-Prinzip abgearbeitet. Die Architekturdokumentation, das Designdokument und die Umsetzung müssen bis Mitte Juli 2017 abgeschlossen werden. Bei relevanten Änderungen werde das entsprechende Dokument unverzüglich angepasst.

Sourcecodeverwaltung

Die Verwaltung erfolgt über Microsoft Tool "Team Foundation"MSDN.

Dokumentation
Alle relevanten Dokumente werden in Deutsch verfasst. Die Dokumentation im Sourcecode erfolgt in Englisch.
Konventionen

- 1
- 2
- 3

1.3 Kontextabgrenzung

1.3.1 Technischer und fachlicher Kontext

1.4 Lösungsstrategie

1.5 Bausteinsicht

1.6 Laufzeitsicht

1.7 Verteilungssicht

1.8 Konzepte

1.9 Entwurfsentscheidungen

1.10 Szenarien

1.11 Risiken und technische Schulden

1.12 Glossar

2 Softwaredesign

2.1 Vorwort

!!!!!!hier kannst du so viel schreiben wie du willst :-)))!!!!!!!!

2.2 Logging

!!!!!!hier kannst du so viel schreiben wie du willst :-)))!!!!!!!!

2.2.1 Contracts

!!!!!!hier kannst du so viel schreiben wie du willst :-)))!!!!!!!!

2.2.2 Log Server

!!!!!!hier kannst du so viel schreiben wie du willst :-)))!!!!!!!!

2.2.3 Log Viewer

!!!!!!hier kannst du so viel schreiben wie du willst :-)))!!!!!!!!

2.2.4 Tests

!!!!!!hier kannst du so viel schreiben wie du willst :-)))!!!!!!!!

2.3 Simulation

!!!!!!hier kannst du so viel schreiben wie du willst :-)))!!!!!!!!

2.3.1 Contracts

!!!!!!hier kannst du so viel schreiben wie du willst :-)))!!!!!!!!

2.3.2 Engine

!!!!!!hier kannst du so viel schreiben wie du willst :-)))!!!!!!!!

2.3.3 Environment

!!!!!!hier kannst du so viel schreiben wie du willst :-)))!!!!!!!!

2.3.4 Settings

!!!!!!hier kannst du so viel schreiben wie du willst :-)))!!!!!!!!

2.3.5 Simulation Objects

!!!!!!hier kannst du so viel schreiben wie du willst :-)))!!!!!!!!

2.3.6 Tests

2.4 Traffic Light Control

!!!!!!hier kannst du so viel schreiben wie du willst :-)))!!!!!!!!

2.4.1 Contracts

!!!!!!hier kannst du so viel schreiben wie du willst :-)))!!!!!!!!

2.4.2 Engine

!!!!!!hier kannst du so viel schreiben wie du willst :-)))!!!!!!!!

2.4.3 Tests

!!!!!!hier kannst du so viel schreiben wie du willst :-)))!!!!!!!!

2.4.4 UI

!!!!!!hier kannst du so viel schreiben wie du willst :-)))!!!!!!!!

2.5 User Interface UI

!!!!!!hier kannst du so viel schreiben wie du willst :-)))!!!!!!!!

2.5.1 Application

!!!!!!hier kannst du so viel schreiben wie du willst :-)))!!!!!!!!

2.5.2 Contracts

!!!!!!hier kannst du so viel schreiben wie du willst :-)))!!!!!!!!

2.5.3 Tests

!!!!!!hier kannst du so viel schreiben wie du willst :-)))!!!!!!!!

2.5.4 View Model

!!!!!!hier kannst du so viel schreiben wie du willst :-)))!!!!!!!!

2.6 WCF Demo

!!!!!!hier kannst du so viel schreiben wie du willst :-)))!!!!!!!!

3 Softwareimplementation

4 TODO

5 Anhang

5.1 Anforderungsdokument

Softwarearchitektur

Thema / Zweck der Aufgaben – Labor

Termin	Aufgabe	Fertigstellung
1	Entwurf der **Softwarearchitektur**	1.Termin
2	Entwurf des **Softwaredesigns**	1. + 2. Termin
3	**Implementierung der Software** nach den Architektur- und Designentscheidungen => Rückkopplungsprozess!	2. + 3. Termin
4	Prototypische Demonstration der Simulation	4.Termin
5	Erweiterung des Projekts nach den Vorgaben von Koll. Eduard Hirsch	
6		
Jul 17	• Architektur & Design & Implementierung • Visual Studio Solution	Abgabetermin von Hr. Hirsch

Gruppenteilteilung

- jeweils **5er Gruppen,**
- wenn Anzahl von Studenten es notwendig machen, dann 4 oder 6er Gruppen erlaubt

Aufgabenstellung

Entwerfen, dokumentieren und implementieren einer

Verteilten komponentenbasierten Verkehrssimulation

Die Simulation sollte wie folgt unterstützen:

- Mikroskopische Simulation der Fahrzeuge (PKW und LKW) und Ampelanlagen
- Zusammenhängendes Verkehrs-/Straßennetz
- Geregelte und ungeregelte Kreuzungen
- Verhalten der Verkehrsteilnehmer sollte weitgehend parametrisierbar sein (variabel/zufällig/individuell)

- Verhalten der Lichtsteueranlage parametrisierbar
  Regelung über eigenen Prozess

- Anzahl der über Einfahrtsstraßen in das System zufahrenden Fahrzeuge regelbar

- Grafische Darstellung und einfache Benutzerschnittstelle
  Konfiguration-GUI nicht unbedingt notwendig

Beispiele: https://www.google.at/search?q=SUMO+Simulation ; http://veins.car2x.org/ ;
http://www.traffic-simulation.de/

Technische Vorgaben

- Visual Studio 2015
- C# / .NET

Alternativ kann zur Entwicklung auch C#/Mono verwendet werden. Die Abgabe am Ende des Semesters hat aber ausnahmslos als Visual Studio Solution zu erfolgen!

Vorbereitung / Hilfestellung / Quellen

- Lesen:

  - Aktivitäten: http://www.arc42.de/aufgaben/processdetails.html

  - Beispiele: http://dokchess.de/dokchess/arc42/index.html
    http://arc42.org/page13/index.html

  - FAQ: http://arc42.org/page7/faq.html

- Hören:

  Aus *SoftwareArchitekTOUR – Podcast für den professionellen Softwarearchitekten*

  Episode 12 - **Systematischer Softwarearchitekturentwurf**
  Download (mp3-Datei): http://heise.de/-353501

  Episode 15 - **Architekturdokumentation**
  Download (mp3-Datei): http://heise.de/-852598

  Weitere Podcasts zum Thema Softwarearchitektur finden Sie unter
  http://www.heise.de/developer/podcast/

Jetzt geht's dann los! - Bedenken Sie aber…

Agile Softwareentwicklung bedeutet, den Entwicklungsprozess flexibler und schlanker zu machen, als dies bei den klassischen Vorgehensmodellen der Fall ist. Eine agile Vorgehensweise hat das Ziel, den Softwareentwicklungsprozess durch Abbau der Bürokratie und durch die stärkere Berücksichtigung der menschlichen Aspekte effizienter zu gestalten. Den meisten agilen Prozessen liegt zu Grunde, dass sie versuchen, die reine Entwurfsphase auf ein Mindestmaß zu reduzieren und im Entwicklungsprozess so früh wie möglich zu ausführbarer Software zu gelangen, die dann in regelmäßigen, kurzen Abständen dem Kunden zur gemeinsamen Abstimmung vorgelegt werden kann. Damit erfolgt die Entwicklung in einem stetigen Rückkopplungsprozess, bei dem sich die Software in jedem Iterationsschritt mehr und mehr den (Kunden-)Anforderungen annähert.

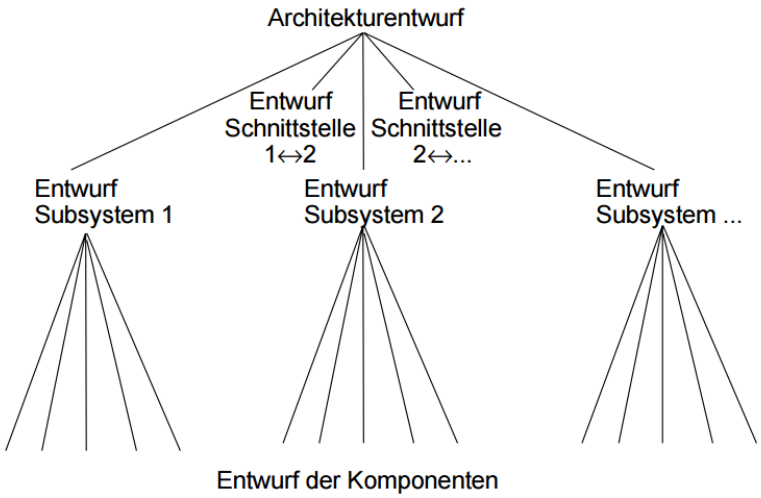
1. Entwurf einer Architektur

Das zu erstellende Dokument soll einen Überblick geben, wie die Software aufgebaut sein soll und wie die einzelnen Softwareteile zusammenwirken, ohne jedoch auf das Design (Klassen, Methoden) und Implementierungsdetails einzugehen (vgl. Architektur vs. Design). Hier sollten nur die grundlegendsten Entscheidungen getroffen werden, ohne jedoch zu sehr ins Detail zu gehen.

Zwei ausgewählte Teams haben im nächsten Labortermin Ihre Architektur in einer kurzen Präsentation zur Diskussion zu stellen. Bereiten Sie dafür eine kurze Präsentation Ihrer Architektur vor (PDF des Architekturentwurfs reicht auch).

2. …von der Architektur zum Design

Designen Sie nun die entworfene Software auf Basis Ihres Architekturentwurfs.



Das zu erstellende Dokument soll einen detaillierten Überblick geben, wie die zu erstellende Software konkret aufgebaut bzw. später implementiert werden soll. Das Design soll die zuvor geplanten Architekturvorgaben umsetzen.

Beschreiben Sie das Design bzw. die geplante Software bis zu jener Detailtiefe (Komponenten-, Interface-, Klassen- und Protokollebene) die eine verteilte Entwicklung (=mehrere Entwickler an verteilten Standworten) möglich macht.

Nutzen Sie Design Pattern, wenn möglich.

Sollten Sie während des Designentwurfs Anpassungen am Architekturentwurf vornehmen müssen (sehr wahrscheinlich), dann passen Sie den Entwurf an. Vergessen Sie nicht, dass alle Änderungen in vorhandenen Dokumenten immer nachvollziehbar dokumentiert werden müssen (Historie, Versionsnummer, usw.)

3. … und vom Entwurf zur Implementierung

Verteilen Sie nun die Aufgaben in Ihrem Team und Implementieren Sie prototypisch Ihre Entwürfe.

4. … und wieder zurück zum Anfang …

Bedenken Sie, dass bei einer agilen Entwicklung durch die frühe Implementierung eine ständige Prüfung des Entwurfs durch die Realisierung erfolgt. Diese Rückkopplung führt zu einer stetigen Anpassung der Anforderungen, der Entwürfe und einer schrittweisen adaptiven Implementierung.

Erst mit dem Ende der Implementierung sind auch die Dokumente fertig. Zuvor handelt es sich nur um momentan gültige Zwischenstände und Entwürfe. Das heißt aber nicht, dass man einfach drauflos programmieren sollte. Nein, je besser die Planung, desto weniger Änderungen sind nachträglich notwendig. Dies soll aber auch nicht heißen, wir planen alles bis ins letzte Detail und dann legen wir erst los.

Die optimale Lösung liegt irgendwo in der Mitte. Die Devise lautet - wie so oft im Leben - also:

So früh/wenig wie möglich und so spät/viel wie notwendig!

Abgaben (eine Teamabgabe am Ende des Semesters)

- *Alle* **Entwicklungsdokumente** (Software Architektur Doku, Software Design Doku, Software Implementations Doku) jeweils als PDF!!

- *Visual Studio Solution* mit allen Projekten (über F5 kompilierbar & ausführbar!!!) mit
  - allen eigenen **Quelltexten**
  - etwaigen **Hilfsbibliotheken**
  - allen notwendigen **Konfigurationdateien** für einen Demolauf
  - weitere Details von Hr. Hirsch (Einheit 5-6)

- **1 ScreenCast** (=> Video), **3** bis **5 Min. Länge**
  mit kurzer Einführung in die lauffähige Software
  => Zeitvorgaben einhalten!! Komprimiertes Format, maximal 40 MByte)

- per Email an roland.graf@fh-salzburg.ac.at und
  Betreff: **SWA_LB_TrafficSIM_[Nachname1].[Nachname2]….[NachnameX]**

- **Abgabetermin:** am Ende des Semesters. Der genaue Termin wird noch bekanntgegeben.