VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

SIMULÁTOR DOPRAVNÍCH INFRASTRUKTUR A SITUACÍ

SIMULATOR OF TRAFFIC INFRASTRUCTURES AND SITUATIONS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. PETR ŠVAŇA**
AUTHOR

VEDOUCÍ PRÁCE **doc. RNDr. JITKA KRESLÍKOVÁ, CSc.**
SUPERVISOR

BRNO 2018

Zadání diplomové práce/21216/2017/xsvana00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2017/2018

Zadání diplomové práce

Řešitel: **Švaňa Petr, Bc.**
Obor: Informační systémy
Téma: **Simulátor dopravních infrastruktur a situací**
Simulator of Traffic Infrastructures and Situations
Kategorie: Informační systémy

Pokyny:

1. Seznamte se s konfigurací dopravních řadičů řady sX společnosti Siemens, s.r.o.
2. Po konzultaci s konzultantem zadávající firmy specifikujte požadavky na systém, který bude tvořit infrastrukturu (napojení) křižovatek na základě poskytnutých konfigurací jednotlivých řadičů. Systém navrhněte.
3. Vytvořte simulátor různých dopravních prostředků a situací (automobily, vozy MHD, slabý/silný provoz, a další) tak, aby spojoval křižovatky se záměrem co nejvíce optimalizovat určité dopravní toky.
4. Navrhněte a implementujte webové rozhraní prezentující vygenerovanou infrastrukturu a dopravní situace.
5. Použitelnost vytvořeného systému otestujte v simulovaném prostředí a demonstrujte na vhodně zvoleném vzorku dat vybraném po dohodě s vedoucí.
6. Zhodnoťte dosažené výsledky a diskutujte další možnosti rozšíření vytvořené aplikace.

Literatura:

- TP 81. *NAVRHOVÁNÍ SVĚTELNÝCH SIGNALIZAČNÍCH ZAŘÍZENÍ PRO ŘÍZENÍ PROVOZU NA POZEMNÍCH KOMUNIKACÍCH.* 3. vydání. Praha, 2015.
- SMĚLÝ, M. *Dopravní inženýrství: Řízené úrovňové křižovatky, část 2.* Brno: VUT v Brně, Fakulta stavební, 2007.
- CRAGG, S. *How microsimulation modelling can address the real-world problems of navigation in congested networks.* TEC. 2007, 48, pg. 123-126.

Při obhajobě semestrální části projektu je požadováno:

- Splnění bodů zadání 1 až 3.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Kreslíková Jitka, doc. RNDr., CSc.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 23. května 2018

L.S. VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ Fakulta informačních technologií Ústav informačních systémů 612 66 Brno, Božetěchova 2

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Cílem této diplomové práce je vytvořit simulační systém s využitím konfigurací dopravních řadičů sX od společnosti Siemens. Systém se skládá dvou aplikací. První aplikace využívá existující platformu SUMO jako zdroj simulačních dat a slouží jako server. Doplňuje ji webová aplikace sloužící jako editor simulačních situací a zároveň vizualizuje data ze simulační aplikace. Úvod do problematiky dopravních řadičů a dynamického řízení dopravy je rozebrán v první kapitole. Následuje popis návrhu a implementace simulační aplikace s důrazem na komunikační rozhraní. V dalších částech text rozebírá návrh a implementaci webové aplikace. V poslední části je probráno ucelené testování celého systému.

Abstract

The aim of this master's thesis is to develop a simulation system using the Siemens sX traffic controllers configurations. The system is composed of two separate applications. The first one uses the existing platform SUMO as a source of simulation data and is also used as a server. The second is a web-based application for creating and editing simulation situations and also for visualisation of simulation data from the server. The introduction to the topic of traffic engineering and the dynamic traffic control is discussed first. The description of the design and implementation of the simulation application directly follows. In the next part the design and implementation of the web-based application is discussed. The last part of the thesis describes the testing of the whole system.

Klíčová slova

simulace, SUMO, TraCI, dopravní kontrolér, konfigurace kontroléru, světelné křižovatky, SVG, webová animace, HTTP

Keywords

simulaton, SUMO, TraCI, traffic controller, controller configuration, traffic lights, SVG, web animation, HTTP

Citace

ŠVAŇA, Petr. *Simulátor dopravních infrastruktur a situací.* Brno, 2018. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. RNDr. Jitka Kreslíková, CSc.

Simulátor dopravních infrastruktur a situací

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením paní doc. RNDr. Jitky Kreslíkové, CSc. a mého konzultanta, pana Bc. Martina Slámy. Další informace mi ve formě jednorázových konzultací poskytli pánové Ing. Lukáš Duda (řízení dopravy na křižovatkách a konfigurace kontroléru), Ing. Zbyšek Gajda, Ph.D. (detektory a virtuální sX) a Ing. Zdeněk Kuna (mikrosimulace). Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Petr Švaňa
22. května 2018

Poděkování

Zde bych rád poděkoval své vedoucí, paní doc. RNDr. Jitky Kreslíkové, a svému konzultantovi, panu Bc. Martinovi Slámovi, za cenné rady a odbornou pomoc při řešení této diplomové práce. Také bych chtěl poděkovat svým kolegům z ITS oddělení společnosti Siemens s.r.o. za jejich ochotu a odborné rady.

Obsah

Kapitola 1

Úvod

Světelné křižovatky mají své místo v dopravě pevně dané. Již po několik desetiletí mají za úkol zabezpečovat provoz na frekventovaných dopravních uzlech. S nástupem moderních dopravních řadičů máme stále více možností, jak dopravu na křižovatkách zefektivnit. Základem je dynamické řízení dopravy, které nám dává k dispozici nástroje pro odstranění dlouhých kolon, zmírnění hlukové zátěže pro okolní obyvatelstvo a zvýšení kvality vzduchu v oblastech se zvýšenou hustotou dopravy.

V mé diplomové práci se věnuji zkoumání právě dynamického řízení dopravy a možnostem dnešních dopravních řadičů. Na základě nasbíraných znalostí a definovaných požadavků bude navrhnuta a implementována simulační aplikaci, která bude jako vstup využívat právě konfigurace těchto řadičů (konkrétněji řadiče řady sX od společnosti Siemens), a která bude sloužit jako základ pro konečný produkt. Tím bude webová aplikace s uživatelským rozhraním, která bude vizualizovat výslednou simulaci.

Cestu k současnému stavu práce popisuji v následujících kapitolách. Druhá kapitola lehce odbočí z informačních technologií do dopravního inženýrství, ačkoliv jako mnoho oborů dnes, i tyto jsou již úzce propojeny. V následujících dvou kapitolách poté popisuji svůj návrh simulační aplikace za využití simulátoru SUMO a její implementaci.

Kapitoly č. 5 a 6 popisují návrh, respektive implementaci webové aplikace, která slouží pro editaci simulované situace a vizualizaci samotné simulace. V následující kapitole se věnuji podrobnému testování různých aspektů obou implementovaných aplikací. V poslední kapitole před závěrem se potom věnuji zhodnocení dosažených výsledků a zamyslím se nad různými vylepšeními implementovaného systému.

Kapitola 2

Konfigurace dopravních řadičů

Tato kapitola obsahuje stručný popis řadiče, co obsahuje a k čemu slouží. Především se potom věnuje jeho konfiguraci, tedy tomu, co vše lze v konfiguraci nastavit a z ní vyčíst. Přitom se zaměřuje na signální plány, signální skupiny a detektory, což jsou prvky konfigurace, které jsou pro tuto práci nejdůležitější.

2.1 Dopravní řadič

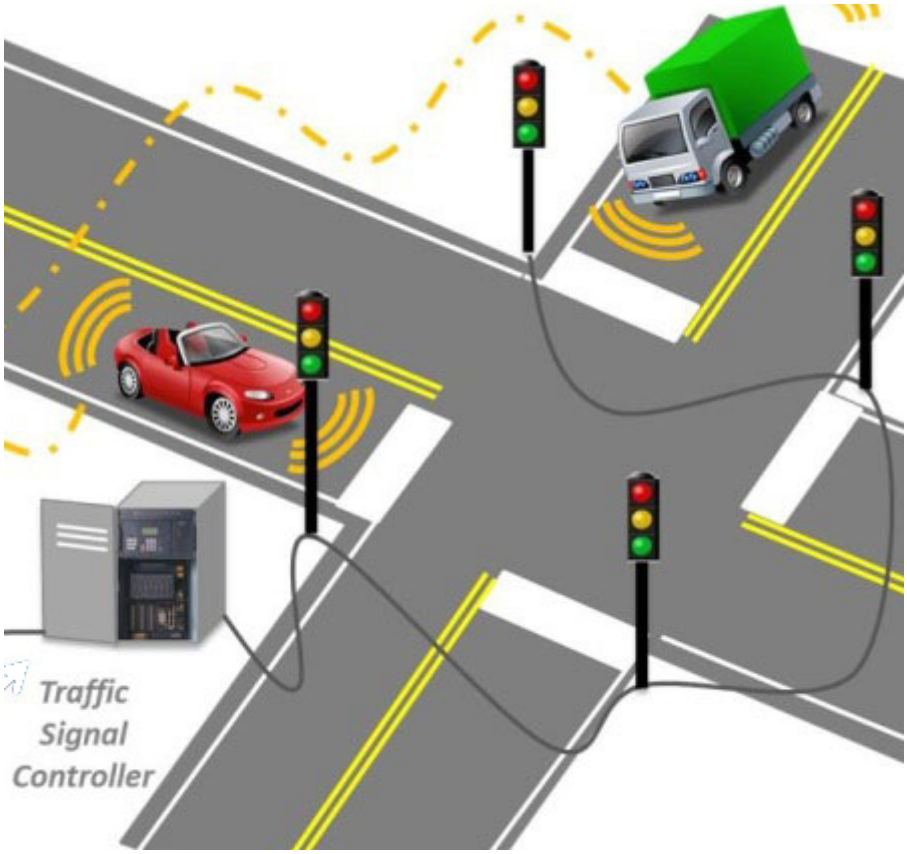
Řadič je hardwarové zařízení, které bývá fyzicky umístěno v blízkosti křižovatky, a které řídí provoz na křižovatce. Je propojen se semafory, kterým určuje, které signály mají zobrazit, a také s detektory (pokud jsou v křižovatce přítomny), které pokročilejší dopravní řadiče využívají jako zdroj dat pro dynamické řízení dopravy. Pokud je oblast, do které křižovatka spadá, ovládána centrálně, je řadič také připojen na centrální systém, odkud může být v případě potřeby na dálku ovládán.

2.1.1 Konfigurace

Standardní chování řadiče je definováno jeho konfigurací. Konfigurace obsahuje kromě jiného následující důležité parametry pro řízení křižovatky:

- signální skupiny,
- signální plány,
- detektory,
- aktivační a deaktivační rutiny,
- tabulka mezičasů a
- informace o hromadné dopravě.

Co z konfigurace nevyčteme, to je potom kompletní topologie křižovatky s ohledem na počet a směrování dopravních pruhů v křižovatce. Je to z toho důvodu, že řadič by tuto informaci pro řízení nijak nevyužil. Musím si tedy v této diplomové práci vystačit pouze s umístěním signálních skupin do jednotlivých vstupů do křižovatky (2.3).



Obrázek 2.1: Zjednodušené schéma role dopravního řadiče na křižovatce (Převzato z [2])

2.2 Klíčové vlastnosti konfigurace

Hlavní vlastnosti, které by každá konfigurace měla splňovat jsou dvě: bezpečnost a průjezdnost. Žádná konfigurace nedokáže fyzicky zabránit dopravní nehodě, při jejím návrhu se však musí co nejvíce dbát na maximální bezpečnost všech účastníků provozu na křižovatce. K tomu slouží mechanismy, jako je například výpočet mezičasů kolizních směrů (2.6.2).

Moderní řadiče se potom vyznačují používáním dynamického řízení provozu, kdy dokáží automaticky rozhodnout například o přepnutí fází signálního plánu, nebo o prodloužení volna v aktuální fázi tak, aby byl provoz co nejplynulejší. Nejlépe však řadič dokáže řídit dopravu tehdy, když je zařazen do většího celku řízení dopravy, např. pro celé město nebo jeho část.

2.3 Signální skupiny

Signální skupina [6] (Signal Group, SG) je soubor návěstidel (Signal Head), která udávají pro jeden dopravní tok stejný signál. Příkladem dopravního toku může být jeden vjezd do křižovatky nebo vstup na jeden přechod pro chodce. Typicky návěstidel bývá více (například u onoho přechodu pro chodce), běžně se však také vyskytují signální skupiny s jediným návěstidlem (například návěstidla pro tramvaje nebo doplňková zelená šipka pro odbočení

vpravo). Protože dopravní řadič typicky obsluhuje jednu nebo více křižovatek, jeho konfigurace obsahuje popis všech signálních skupin, které ovlivňují dění na křižovatce.

Pro účely této práce tedy potřebujeme vědět, které různé signální skupiny jsou obsaženy v konfiguraci. Dále nás u klasických tříbarevných signálních skupin pro vozidla zajímají údaje o tom, jak dlouho trvají přechody mezi okrajovými stavy signální skupiny. Jedná se o tyto údaje:

- Doba trvání přepnutí zeleného signálu "VOLNO" na červený signál "STŮJ", která typicky odpovídá době trvání přechodného signálu "POZOR" s rozsvíceným žlutým světlem.
- Doba trvání přepnutí červeného signálu "STŮJ" na zelený signál "VOLNO", která typicky odpovídá době trvání přechodného signálu "POZOR" s rozsvíceným žlutým a červeným světlem, který značí povinnost připravit se k jízdě.

U řadiče typu sX, jehož konfigurace ve své práci využívám, lze také z definice vyčíst úhel dané signální skupiny vzhledem k centru křižovatky. Tímto lze tedy určit dopravní toky, které daná signální skupina řídí. Jedná se tedy o klíčový parametr. Dalším důležitým parametrem čitelným z konfigurace je typ signální skupiny (např. klasická, pro pěší, odbočení vlevo, vyklizení křižovatky, pro tramvaje) a pokud se jedná o signální skupinu pro vozidla, tak také jejich případný směr (jsou-li na křižovatce použity semafory se signalizací směru).

2.4 Signální plány a jejich využití

Signální plán je program řízení SSZ[1], určující pořadí a délku signálů "VOLNO"jednotlivých signálních skupin.

Z hlediska konfigurace sX řadiče existují 2 typy signálních plánů:

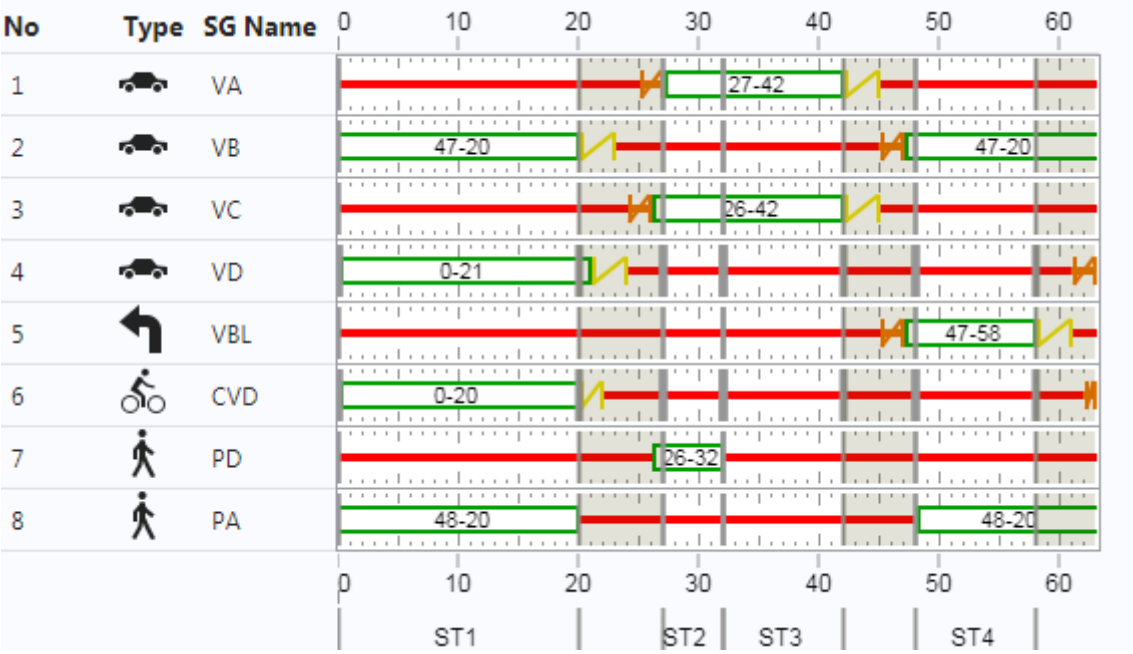
- Signální plány orientované na signální skupiny (pevné)[2].
- Signální plány orientované na etapy (dynamické)[3].

Hlavní rozdíl mezi těmito typy je ten, že signální plány orientované na signální skupiny jsou naplánovány pevně, není tedy možné jimi dynamicky řídit dopravu, například změnou délky volna. Jejich výhodou je jejich jednoduchost, a u některých dopravních situací (nebo u některých křižovatek) mohou postačovat.

Dynamické řízení křižovatky umožňuje až signální plán orientovaný na etapy. Ten je rozdělen na fáze a na fázové přechody. Fáze mohou mít proměnnou délku a jejich pořadí se může měnit. Fázový přechod definuje změnu, která se provede mezi dvěma fázemi, a jeho trvání je neměnné. Pro ilustraci na obrázku 2.2 jsou fázové přechody podbarveny šedou barvou. Vlevo na obrázku jsou vypsány signální skupiny na křižovatce, vpravo je na časové ose vyznačena posloupnost jejich fází (bílé pozadí) a přechodů (šedé pozadí). Protože se pořadí fází může měnit, je nutné mít všechny možné fázové přechody. Dynamickým řízením křižovatky se myslí především tyto možnosti:

- Proměnná délka volna.
- Změna pořadí fází.

[1] SSZ = Světelné signalizační zařízení
[2] Signal group oriented signal plans
[3] Stage oriented signal plans



Obrázek 2.2: Ukázka signálního plánu orientovaného na etapy [snímek obrazovky]

- Vkládání fáze na výzvu.
- Okamžité doplnění nekolizního volna do probíhající fáze.

Tvorbou signálních plánů se v této práci detailněji zabývat nebudu, neboť výsledná aplikace pouze využívá již existující konfigurace.

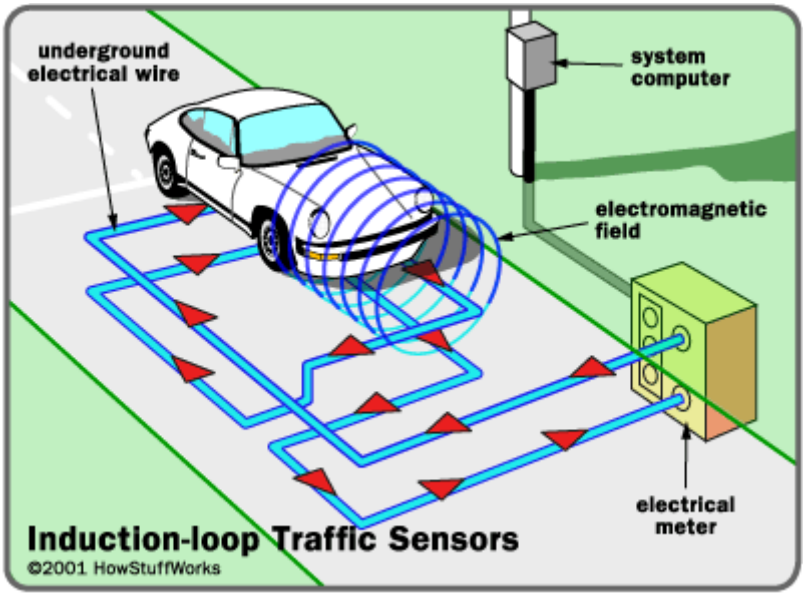
2.5 Detektory

Detektory na křižovatce slouží především k detekci průjezdu jednoho nebo více vozidel. Rozlišovací schopnost závisí na druhu detektoru a jeho technologii. Dále se rozlišují podle toho, k čemu konkrétně slouží. Správné určení mezičasů má tedy zásadní vliv na bezpečnost na křižovatce.

Rozlišujeme následující druhy detektorů:

- Indukční smyčky - Tyto detektory se dělí ještě na jednoduché a dvojité. Jednoduché mohou pouze detekovat projíždějící auto, dvojité dokážou navíc detekovat i jeho směr a rychlost. Jeho fungování je popsáno na obrázku 2.3.
- Video detektory - Jsou instalovány na stojanech semaforů, dokážou detekovat více vozidel, problémem mohou někdy být povětrnostní podmínky.
- Magnetické detektory.
- Infračervené detektory.
- Radarové detektory.
- Rádiové detektory.

- Tlačítka na přechodu pro chodce, kterým chodec signalizuje záměr přejít přechod pro chodce.



Obrázek 2.3: Jak funguje indukční smyčka (Převzato z [1])

2.5.1 Dynamické řízení dopravy

(dopsat další způsoby dy. řízení dopravy) Traffic Actuation[4] je jedním z nejčastějších prostředků dynamického řízení dopravy na světelných křižovatkách. Řadič na základě dat z detektorů vyhodnocuje dopravu z jednotlivých směrů. V praxi to funguje tak, že když je detekováno projíždějící vozidlo a zároveň na semaforu svítí signál "VOLNO", je právě běžící fáze prodloužena o časový úsek definovaný v konfiguraci. Fáze má zároveň definovány své minimální a maximální doby trvání, prodloužení tedy může fungovat maximálně do doby nejdelšího definovaného trvání fáze, poté fáze končí.

2.6 Další údaje čitelné z konfigurace a jejich využití

2.6.1 Aktivační a deaktivační rutiny

Aktivační a deaktivační rutiny jsou ve své podstatě speciální signální plány. Ty určují, jak se křižovatka zachová, pokud má dojít k deaktivaci[5] aktivního řízení provozu, nebo naopak k jeho aktivaci.

2.6.2 Tabulka mezičasů

Mezičas je časový interval od konce signálu "VOLNO" signální skupiny po začátek signálu "VOLNO" kolizní signální skupiny. A za tuto dobu musí poslední projíždějící vozidlo opustit

[4] příp. Vehicle Actuation

[5] Neaktivní křižovatkou myslíme křižovatku, která se nepodílí na řízení provozu (je zcela vypnutá, nebo její signální skupiny blikají žlutou barvou)

křižovatku. Kolizní vztah mezi signálními skupinami znamená, že směry jejich dopravních toků se navzájem kříží. Používá se pro tvorbu signálních plánů[6].

2.6.3 Veřejná hromadná doprava

Hromadné dopravy se také týká dynamické řízení dopravy, neboť dnešní moderní systémy zvládnou prioritizaci hromadné dopravy na světelných křižovatkách. Z konfigurace by nás k tomuto tématu zajímala především spárovaná signální skupina, která má prioritizaci provádět a také detektor, kterým musí vozidlo hromadné dopravy projet pro jeho registraci na křižovatce a jeho směr.

2.7 Dopravní řadič sX

Na závěr této kapitoly věnuji pár vět dopravnímu řadiči sX [4]. Jedná se o novou řadu radičů od společnosti Siemens. Má za cíl zvýšení bezpečnosti a také uživatelské přívětivosti - jednou z hlavních novinek bylo představení webového rozhraní, které technikům dovoluje přistupovat k datům z řadiče například pomocí webového prohlížeče. Také toto rozhraní dovoluje řadič nastavovat, případně nahrávat konfigurace nebo měnit signální plány s tím, že není pro tyto činnosti potřeba vypnutí řízení křižovatky. Řadiče sX byly poprvé veřejnosti představeny na dopravním veletrhu Intertraffic v roce 2014.

[6] angl. Intergreen times Matrix, ItMatrix

Kapitola 3

Návrh simulátoru

V této kapitole se věnuji návrhu programu, který z dané konfigurace (konfigurací) a z dalších zadaných parametrů vytvoří simulovaný provoz. Program bude základem celé webové aplikace, jejíž návrh bude následovat v dalších kapitolách.

3.1 Požadavky na simulátor

Po dohodě s mým konzultantem jsme určili funkční požadavky na výsledný simulátor:

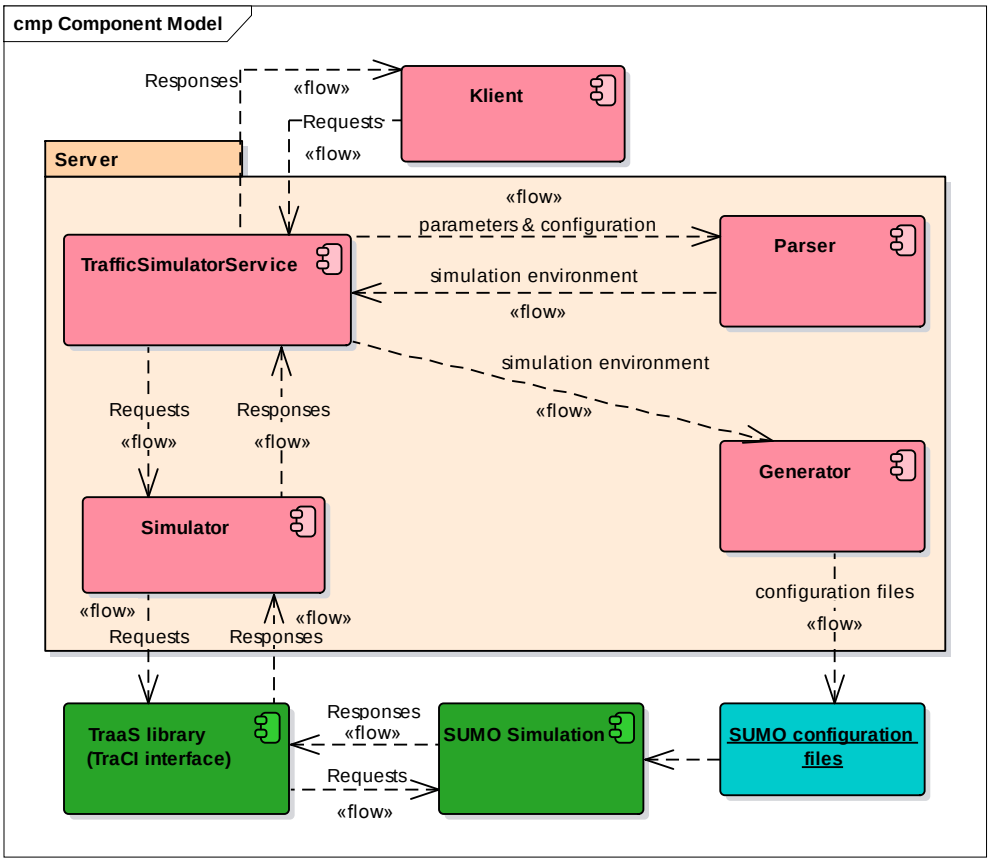
- Podpora křižovatek, které mají 2 až 4 ramena.
- Detektory budou zohledněny.
- Pokud je křižovatek více, budou spojeny automaticky nebo manuálně.
- Pro samotnou simulaci použijte nástroj SUMO.
- Parametry simulace lze z webového rozhraní měnit. Změna parametrů může mít za následek restart simulace.
- Uživatel může konfigurovat simulaci následovně:
  - Přidáním nové křižovatky do simulace a definováním její pozice vzhledem k existující infrastruktuře.
  - Přidáním jízdního pruhu do křižovatky.
  - Otočením libovolné křižovatky dle poskytnutých možností.
  - Vybráním předdefinovaných typů dopravních prostředků a nastavením jejich hustoty.
  - Systém bude obsahovat základní prostředky pro optimalizaci provozu (např. Traffic Actuation).
  - Dynamicky ovlivňovat simulaci aktivací předdefinovaných situací (např. průjezd záchranného vozidla simulovaným prostředím).
  - Systém si parametry určí automaticky, pokud nejsou zadány uživatelem.

Určili jsme také volitelné požadavky:

- Uživatel může určit hustotu pro jednotlivé směry.

- Simulace bude podporovat chodce - implementace přechodů pro chodce.
- Uživatel explicitně vybere, která ramena z přidávané křižovatky chce připojit ke kterým ramenům existující infrastruktury.
- Podpora aktivačních a deaktivačních rutin pro možnost libovolnou křižovatku vypnout.

3.2 Návrh architektury



Obrázek 3.1: Návrh architektury [vlastní]

Navrhovaný program bude mít jako vstup zadané konfigurace sX řadičů (ve formátu XML). Tyto konfigurace zpracuje do své vnitřní reprezentace, ze které jednak vygeneruje konfigurační soubory pro program SUMO, a kterou bude využívat pro další ovládání simulace pomocí rozhraní TraCI.

Při návrhu jsem se snažil o jistou modularitu tak, aby nebylo příliš problematické vytvořenou aplikaci v budoucnu rozšířit. Jedním z reálných požadavků pro rozšíření může být například nevyužití programu SUMO jako generátoru dat pro vizualizaci, ale využití reálných dopravních dat z řadičů.

Jak je patrné z obrázku 3.1, rozdělil jsem si tedy program na 4 logické celky (moduly):

- Parser, který dostane na vstup parametry a vstupní konfigurace řadičů ve formátu XML. Tyto informace použije pro vytvoření vnitřní reprezentace konfigurace simulačního systému.
- Generator z vnitřní reprezentace konfigurace vygeneruje konfigurační soubory pro platformu SUMO.
- Simulator ovládá simulaci pomocí rozhraní TraCI (viz 3.5.3).
- TrafficSimulatorService slouží jako hlavní spustitelný modul, jenž bude přijímat HTTP požadavky od klientské aplikace a reagovat na ně.

Jelikož jsem si pro implementaci simulátoru vybral jazyk Java, budu využívat knihovnu TraaS, která v sobě kombinuje rozhraní TraCI a možnost využít jej jako webovou službu. Přes ní bude možné v další části diplomové práce zasílat informace o pohybu vozidel do webové aplikace, která bude simulaci zobrazovat a také simulaci ovládat z webové aplikace.

Nakonec SUMO simulace se bude spouštět s vygenerovanými soubory na příkaz z klientské aplikace.

3.2.1 SpringBoot

Pro implementaci spustitelného modulu `TrafficSimulatorService` jsem si zvolil framework SpringBoot[1], především kvůli jeho jednoduché konfiguraci a integrovanému HTTP serveru.

3.3 Návrh komunikačního rozhraní

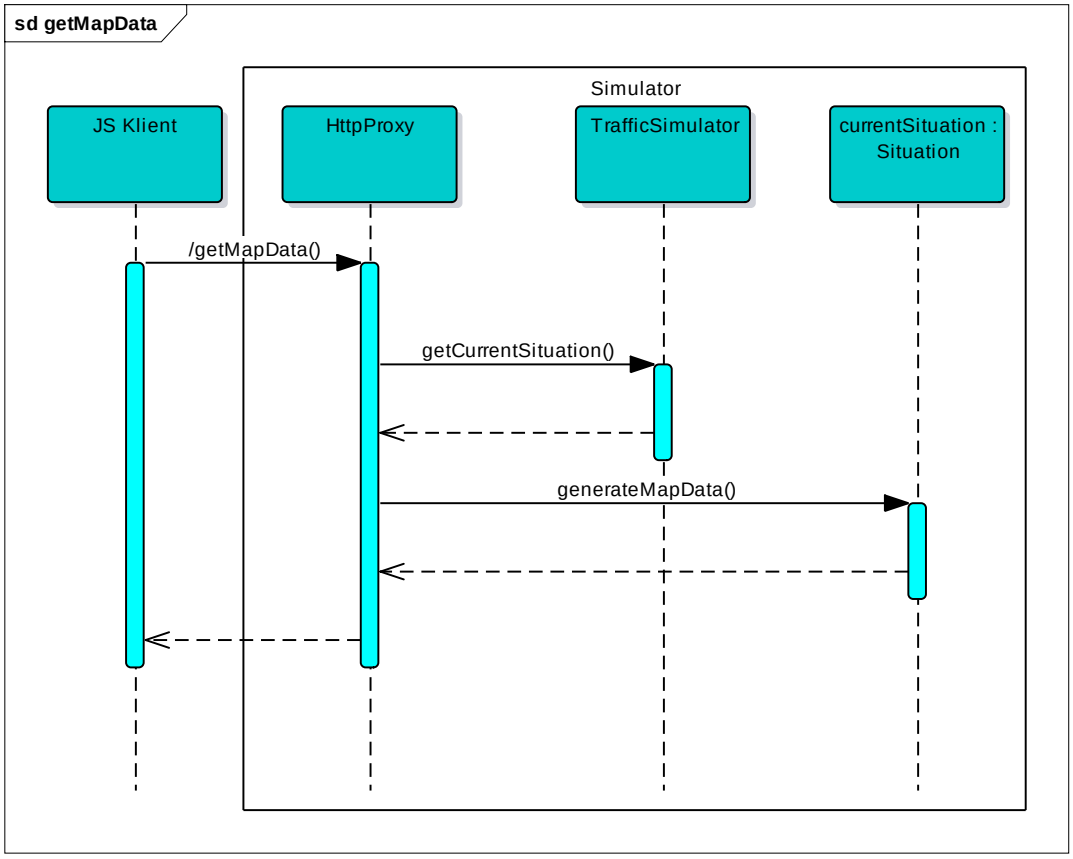
Simulátor by také měl poskytovat rozhraní pro komunikaci s klientem. V této diplomové práci budu pro jednoduchost počítat pouze s jednou běžící instancí klienta. Možnost připojení více klientů k serveru není cílem této diplomové práce a může být vhodným rozšířením v budoucnu. Komunikaci bude iniciovat klientská aplikace, která bude na základě svého stavu zasílat požadavky, a modul `TrafficSimulatorService` v roli serveru bude na tyto požadavky odpovídat. Příkladem takového požadavku může být dotaz na statickou část simulace, resp. podklad vizualizace, který je ukázán na obrázku 3.2. Z něj je patrné, že o příchozí požadavky se bude starat třída `HttpProxy`, která bude zodpovědná za správné přečtení požadavku, za sestavení příslušné odpovědi a za korektní odeslání odpovědi.

Komunikace bude využívat standardní HTTP protokol ve verzi 1.1 a jeho metody `GET` pro obecné získávání data a `POST` pro zasílání náročnějších dat klientem (např. konfigurace křižovatek) [9]. Klientská aplikace se připojí na IP adresu a port ke spuštěnému serveru. Jednotlivé požadavky se budou rozlišovat dle jejich zadaných URN[2].

3.4 Sumo - simulátor dopravy

SUMO[3] [11] je volně dostupná, open-source simulační platforma, vyvíjená od roku 2001. Je specializovaná na simulaci dopravy, a to až na úroveň jednotlivých elementů (např. vozidlo, chodec, apod.), jejichž počet není nijak limitován. Umožňuje také vnější interakci s běžící simulací pomocí doplňkového rozhraní TraCI. Velmi široké možnosti nastavení umožňují jeho

[1] https://projects.spring.io/spring-boot/
[2] Uniform Resource Names
[3] Simulation of Urban MObility



Obrázek 3.2: Sekvenční diagram znázorňující získávání statických simulačních dat [vlastní]

využití v této diplomové práci. Na konci této sekce se však budu také věnovat alternativám k SUMO.

Simulační program SUMO jako svůj vstup používá sadu XML souborů které určují jednak dopravní infrastrukturu, tzv. síť (network), dále potom doplňkové parametry, jako jsou cesty v této síti a detektory [3].

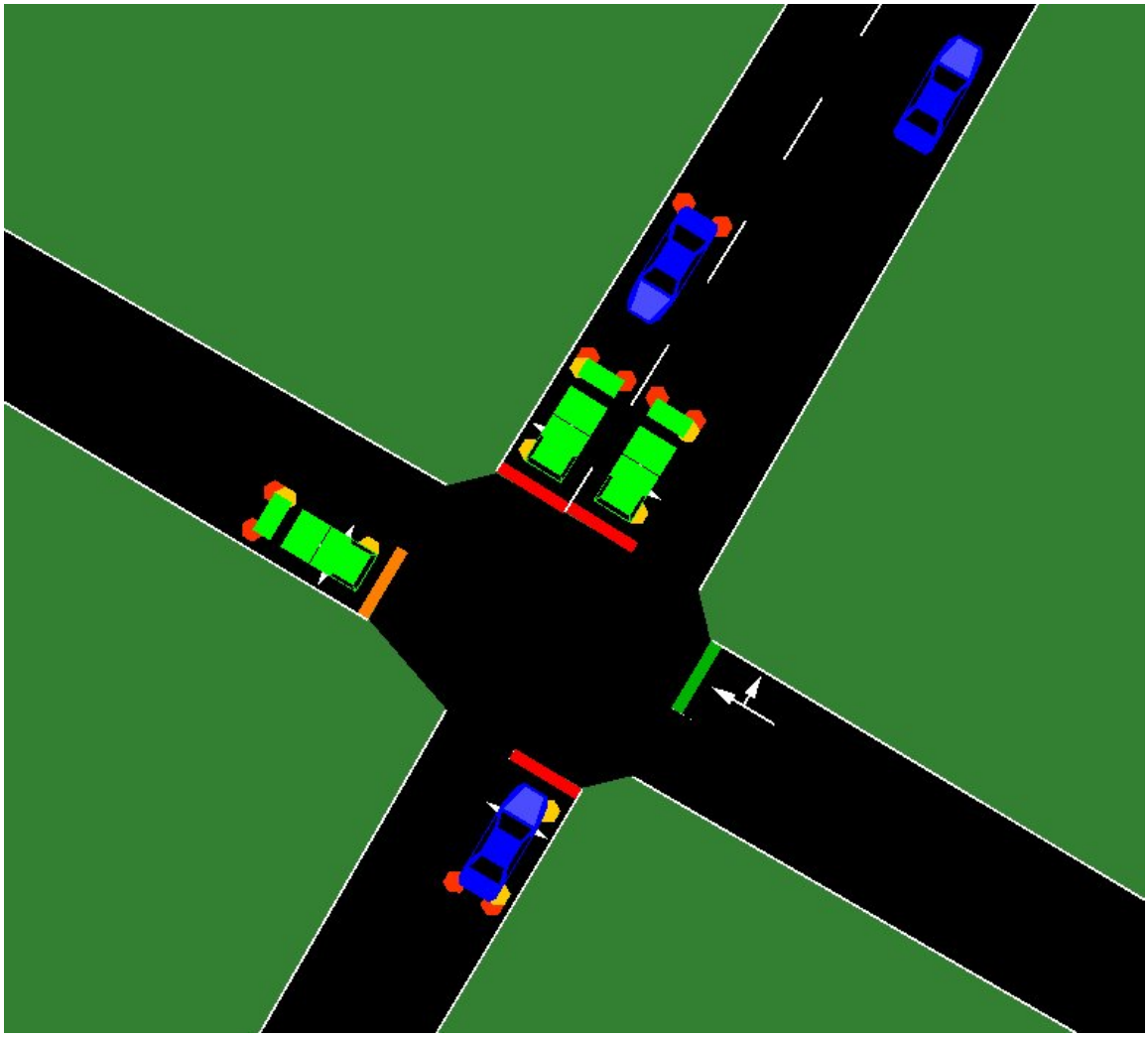
3.4.1 Konfigurační soubory pro tvorbu dopravní infrastruktury

Dopravní síť se vytváří nástrojem `netconvert`(3.5.1), který kombinuje 3 vstupní XML soubory:

- Soubor obsahující definici uzlů - Nodes file.
- Soubor obsahující definici hran - Edges file.
- Soubor obsahující definici spojení - Connections file.

Uzel reprezentuje křižovatku nebo začátek/konec příjezdové cesty ke křižovatce. Každý uzel je definován svým interním identifikátorem, pozicí v souřadnicovém systému a typem. Typ[4] určuje, jak se má daný uzel v simulaci chovat. V této práci budu využívat typ

[4]Další typy jsou popsány v dokumentaci projektu SUMO.



Obrázek 3.3: Ukázka z vizualizační aplikace platformy SUMO [snímek obrazovky]

`traffic_light` pro uzly reprezentující křižovatky se světelnou kontrolou. U příjezdových cest tolik na typu nezáleží, neboť se nejedná o křižovatku, takže zde poslouží jakýkoliv typ který neočekává řízení pomocí SSZ (např. `priority`).

Hrany v SUMO reprezentují silnice, tedy spojnice mezi dvěma uzly v jednom směru. Jsou definovány svým identifikátorem a počátečním a koncovým uzlem. Lze jim zadat také další parametry, například počet jízdních pruhů, maximální rychlost, nebo které druhy vozidel zde mají zákaz vjezdu.

Nakonec spojení definuje, jak se jednotlivé příjezdové cesty na křižovatce propojí s odchozími. Pro tento problém jsou dvě metody. U té jednodušší se definuje pouze příjezdová a odjezdová hrana, u té složitější poté konkrétní jízdní pruhy daných hran. V této práci bude využita druhá možnost.

3.4.2 Další konfigurační soubory

Doplňující konfigurační soubory se spolu se souborem s dopravní infrastrukturou předkládají programu SUMO při spuštění simulace. V této práci se budou využívat následující XML soubory:

- Soubor s definicí dopravních toků a tras (Routes, Flow).
- Soubor s definicí světelných signalizací (Traffic lights signalisation).
- Soubor s definicí detektorů.

Vytvoření dopravních toků (flow) je jednodušší, než vytvoření trasy (route), neboť pro správnou definici toku nám stačí znát pouze počáteční a koncovou hranu, a na programu SUMO necháme konkrétní trasu. Pro explicitní definici trasy ale potřebujeme vyjmenovat postupně všechny hrany, kterými trasa od počátku až na konec prochází, což může být u složitější infrastruktury poměrně zajímavý problém. Zjistil jsem však, že podobný úkol může vyřešit nástroj `duarouter`(3.5.2), který je součástí SUMO platformy.

U světelné signalizace vyvstává další zajímavý problém, kde je výrazný rozdíl mezi tím, jak jsou signální plány definovány v konfiguraci řadiče sX a jak by měly být definovány pro SUMO. V konfiguraci jsou dynamické signální plány definovány tak, že jednotlivé signální skupiny mají určený časy, kdy se přepnou na jiný signál. SUMO naopak očekává lineárnější přístup, který je popsaný na výpisu níže.

```
<additional>
   <tlLogic id="0" programID="my_program" offset="0" type="static">
      <phase duration="31" state="GGggrrrrGGggrrrr"/>
      <phase duration="5" state="yyggrrrryyggrrrr"/>
      <phase duration="6" state="rrGGrrrrrrGGrrrr"/>
      <phase duration="5" state="rryyrrrrrryyrrrr"/>
      <phase duration="31" state="rrrrGGggrrrrGGgg"/>
      <phase duration="5" state="rrrryyggrrrryygg"/>
      <phase duration="6" state="rrrrrrGGrrrrrrGG"/>
      <phase duration="5" state="rrrrrryyrrrrrryy"/>
   </tlLogic>
</additional>
```

Výpis 3.1: Ukázka vygenerovaného souboru s definicí světelných signalizací pro SUMO.

Ukázka popisuje signální plán pro jednu křižovatku, kde je signální plán rozdělen do fází. Fáze zde trvá tak dlouho, dokud nenastane změna v řetězci `state`, který značí stavy všech signálních skupin na křižovatce pro danou fázi[5].

3.4.3 Model dopadu na životní prostředí

Díky projektu iTetris[6] byl projekt SUMO rozšířen o modelování dopadu na životní protředí [12]. Nyní tedy je již možné pomocí SUMO získat informace o vypouštěných škodlivých látkách jednotlivých vozidel, jejich spotřebě paliva nebo jejich zvukových emisích. Tyto

[5] Vysvětlivky: `g`, `r`, `y` značí "VOLNO", "STŮJ", "POZOR". Současný signál červené a žluté před startem signálu "VOLNO"je označen písmenem `u`. Velké písmeno `G` značí přednost vozidel v daném směru.

[6] an Integrated Wireless and Traffic Platform for Real-Time Traffic Management Solutions

údaje jsou založeny na tabulce HBEFA[7], což je databáze emisí jednotlivých typů vozidel. Tyto údaje by bylo možné využít v této diplmové práci jako rozšíření výstupu simulace.

Druhý pro mě příznívý dopad tohoto projektu na projekt SUMO bylo zrušení minimální hranice délky simulačního kroku. To bylo původně nastaveno na 1s [7]. Délce simulačního kroku se budu také věnovat dále.

3.4.4 Alternativy k SUMO

SUMO není jediným mikro-simulačním nástrojem k dispozici. Mezi jeho nejbližší konkurenci se řadí nástroje TRANSIMS[8] a VISSIM[9].

TRANSIMS je stejně jako SUMO volně dostupný open-source nástroj. Jeho odlišností od SUMO je jeho zaměření - TRANSSIMS je rychlejší [13], ale méně detailní než SUMO - nepodporuje například prostorově kontinuální simulaci, pouze diskrétní. TRANNSIM je více zaměřený na modelování a simulaci větších regionů s vyšším počtem vozidel.

VISSIM je naopak velmi detailní komerční nástroj od společnosti PTV. Kromě 2D nabízí i 3D vizualizaci a jeho modely pro chování vozidel (a řidičů) jsou nejpřesnější z této trojice.

3.4.5 Alternativy k mikrosimulaci

Při vymýšlení návrhu jsem zlehka prozkoumal zcela jiný přístup k problému, který v sobě místo využití mikrosimulačního systému (jako je SUMO) zahrnuje přímo využití simulovaných řadičů sX ve formě virtuálních stojů. Tento přístup jsem ale nakonec zavrhl jako celkově příliš náročný pro diplomovou práci - dosáhnutí některých požadavků na systém by zde bylo velmi pracné. Kromě toho jsem musel také zvážit hardwarové požadavky - pro jednu křižovatku by bylo potřeba mít jeden virtuální stroj s nemalými požadavky na zdroje. Nelze však vyloučit, že bude zájem o bližší prozkoumání této cesty po ukončení vlastní diplomové práce.

3.5 Sumo - nástroje

Platforma SUMO obsahuje kromě vlastního simulačního programu také další nástroje, které pomáhají například s tvorbou dopravní infrastruktury, nebo pro vizualizaci simulace. Ty, které jsou použité v této práci, se pokusím představit na následujících řádcích.

3.5.1 Netconvert

Nástroj `netconvert`[10] je jeden z několika nástrojů SUMO pro konverzi simulačních sítí do formátu pro SUMO. Lze jej využít na konverzi sítí z jiných systémů (např. VISSIM nebo OpenStreetMap). Já jej budu využívat pro vytvoření simulační sítě ze 3 souborů (uzly, hrany, spojení - 3.4.1). Tyto soubory lze relativně jednoduše vygenerovat automaticky.

[7] http://www.hbefa.net/
[8] TRansportation ANalysis and SIMulation System, https://sourceforge.net/projects/transimsstudio/
[9] http://vision-traffic.ptvgroup.com/en-us/products/ptv-vissim/
[10] netconvert: http://sumo.dlr.de/wiki/NETCONVERT

3.5.2 Duarouter

Další nástroj `duarouter`[11] budu využívat pro řešení jednoho z problémů, který při řešení práce nastal - problém s vyhledáváním cest v simulační síti (3.9). Tedy pro konverzi souboru s definicí dopravních toků na soubor s definic cest. Tento výsledný soubor však bude potřeba ještě dodatečně opravit. Dalším využitím tohoto programu může být například oprava neúplných definicí cest. `Duarouter` používá pro vyhledání cest již existující směrovací algoritmy, jako třeba Dijkstra (ve výchozím nastavení), nebo A*.

3.5.3 TraCI

TraCI[12] je rozhraní pro SUMO, které slouží pro vzdálený přístup k běžící simulaci. Pro účely této práce mě zajímá především následující on-line funkcionalita, kterou TraCI nabízí[13]:

- Získávání aktuálních informací z detektorů.
- Získávání aktuálních informací ze světelných křižovatek.
- Získávání aktuálních informací o poloze/rychlosti libovolných vozidel.
- Řízení světelných křižovatek.
- Přidávání/modifikace vozidel.

3.5.4 TraaS

TraaS[14] [5] je rozšíření k simulační sadě SUMO. Prakticky se jedná o Java knihovnu, která v sobě má zakomponovány volání TraCI rozhraní. Zároveň se však také dokáže chovat jako webová služba.

3.6 Parametry simulace

Simulační program SUMO umožňuje široké možnosti nastavení. Jedním z těch nejdůležitějších nastavení je délka simulačního kroku.

3.6.1 Ideální délka simulačního kroku

Délka simulačního kroku v našem případě odpovídá reálnému času. Pokud zvolíme délku simulačního kroku `1`, jeden simulační krok bude odpovídat pohybu, který vozidla provedou za 1 vteřinu reálného času. Hodnota simulačního kroku není omezena, lze tedy docílit i velmi jemné simulace, kdy jeden simulační krok odpovídá 0,001 s. Pro výběr správné hodnoty potřebuji zohlednit následující faktory:

- Přesnost simulace s ohledem na vizualizaci.
- Objem přenesených dat.
- Časová náročnost generování jednoho simulačního kroku.

[11] duarouter: http://sumo.dlr.de/wiki/NETCONVERT
[12] Traffic Control Interface:
[13] Kompletní výčet je uveden zde: http://sumo.dlr.de/wiki/TraCi#TraCI_Commands
[14] TraCI as a Service

První faktor je jednoznačně pro co nejmenší délku simulačního kroku. Zbývající však více zohledňují praktickou stránku věci a také fakt, že příliš malá délka simulačního kroku může ve výsledku přinést problémy s přenosem dat na klientskou aplikaci a s jejich vykreslením a animací. Bude tedy potřeba najít přijatelný kompromis mezi výkonem a přesností.

3.6.2 Modely chování vozidel

Simulační projekt SUMO dává uživateli na výběr ze dvou druhů chování vozidel:

- Model následování vozidla (Car-following model)
- Model pro přejíždění mezi dopravními pruhy (Lane-changing model)

Modely pro následování vozidel

Modely pro následování vozidel slouží k napodobení chování řidiče, který následuje vozidlo jedoucí před ním [10]. Tyto modely jsou široce používány v simulátorech dopravy. Model typicky podrobně implementuje chování řidiče a jeho reakce na pohyb vozidla, jenž se nachází před ním. Cílem je vyhnout se kolizím. Tyto modely se většinou soustředí pouze na vozidlo, které je ve stejném pruhu. V SUMO je na výběr několik takovýchto modelů, z nichž mezi ty nejpoužívanější patří:

- Model Krauß (implicitní model pro SUMO)
- Inteligent Driver Model

Další dostupné modely jsou většinou odvozené právě od těchto modelů. Tyto modely předpokládají, že druhé vozidlo bude dodržovat bezpečnou vzdálenost vůči prvnímu vozidlu. Podle toho přizpůsobí svoji rychlost tak, aby byl schopen případně zastavit včas a zamezit tak kolizi. Tyto modely se dají dále konfigurovat speciálními parametry, které umožní nastavit například bezpečnou vzdálenost. Zajímavostí u některých modelů je možnost nastavit míru řidičovy dokonalosti.

Pro tuto diplomovou práci využiji základní model Krauß, neboť Inteligent Driver Model má v současné verzi SUMO problém se změnou dopravních pruhů. Pokud se problém časem opraví, možná by bylo vhodné tento model vyzkoušet. Podle citované studie [10] vykazuje menší chybovost než standardní model Krauß.

Modely pro přejíždění mezi dopravními pruhy

Tato třída modelů popisuje chování řidičů při přejíždění do jiného pruhu v souběžné dopravě ve více dopravních pruzích v jednom směru. Motivací pro řešení takového problému v mikrosimulacích dopravy může být vícero. Například není žádoucí vznik kolon z toho důvodu, že se vozidla nezařadila do správného odbočovacího pruhu včas a tím blokují zbytek. Cílem je přitom co nejvíce se přiblížit reálnému chování řidičů v provozu. Úlohou těchto modelů je tedy především minimalizovat situace, ke kterým v reálném provozu dochází jen zřídka.

Zde využiji standardní SUMO model, který je označen jako LC2013.

3.7 Implicitní určení počtu pruhů na křižovatce

Jak jsem již zjistil při zkoumání konfigurace (2.3), mohu při jejím zpracování zjistit o topologii jen následující: kolik existuje vstupů do křižovatky a ke kterému vstupu se má daná

signální skupina přiřadit. Nemohu však s přesností určit počet pruhů v křižovatce, ačkoliv je tento údaj pro dynamiku dopravy důležitý v reálném světě (a tím pádem i v simulaci).

Jeden přístup k tomuto problému je umožnění uživateli, aby si mohl přidat pruhy sám. Tím tak umožním zpřesnit situaci tak, aby dopravní pruhy plně odpovídaly simulované křižovatce (pokud má reálnou předlohu). Nakonec je to také jeden z požadavků na implementaci simulátoru (3.1).

Z konfigurace mohu alespoň určit minimální počet pruhů v jednom vstupu. Minimální počet pruhů pro jednu signální skupinu v jednom vstupu je 1, nicméně signálních skupin z jednoho směru může být více (např. pro cestu rovně, pro odbočení doprava a pro odbočení doleva = minimálně 3 pruhy).

Teoreticky by mi mohla pomoci také definice detektorů, protože ty jsou spárovány se svou signální skupinou. Zde však bohužel neplatí rovnice detektor = pruh, neboť počet detektorů v jednom pruhu (i stejného typu) není nikde omezen, a tak je tento přístup v praxi nepoužitelný.

3.8 Spojování křižovatek

Protože systém by měl podporovat více křižovatek najednou, další otázkou bylo, jakým způsobem přistupovat ke spojování křižovatek (manuálním i automatickém), a jak vlastně křižovatky do simulace umisťovat.

Jak již bylo naznačeno výše, z pohledu místa v simulaci je křižovatka určena jedním uzlem definujícím střed křižovatky a několika dalšími uzly, které reprezentují počátky příjezdových tras ke křižovatce. Mezi těmito uzly vedou cesty (hrany) které slouží jako příjezdové trasy. Tyto příjezdové trasy mají dostatečně velkou konstantní délku.

Rozhodl jsem se pro zjednodušení celé situace tuto definici křižovatky zapouzdřit tak, že jedna křižovatka bude jeden bod na nadřazeném souřadném systému, kde bude první křižovatka nést souřadnice [0,0].

Manuální přidávání křižovatek by tedy ve výsledné aplikaci mohlo fungovat tak, že si uživatel vybere pozici relativní k první (nebo další) křižovatce. Jejich spojení by poté bylo podmíněno jejich přirozeným navazováním (s případnou malou odchylkou). Uživatel by měl mít v tomto výhodu v tom, že si bude moci navíc definovat případné natočení křižovatky. Takto k sobě blízké vstupy/výstupy z křižovatek se automaticky propojí přidáním spojovací cesty (hrany).

Pokud uživatel pozici manuálně nevybere, bude vybrána automaticky podle toho, kde má jedna z křižovatek v simulaci volný vstup.

3.9 Vyhledání cest v simulační síti

Simulace vozidla v SUMO funguje v závislosti na způsobu jeho definice. Definováno může být buď explicitně jako vozidlo, které lze jednoznačně identifikovat a má určenou trasu z hran (`edges`), které má navštívit (a které na sebe musí navazovat). Druhý způsob je takový, že není definováno přímo vozidlo se svou trasou, ale dopravní tok (`flow`), který se o generování vozidel stará sám na základě zadaných parametrů (pravděpodobnost vyslání za jeden simulační krok). Tento způsob má tu nevýhodu, že zde nelze adresovat konkrétní vozidlo. Na druhou stranu však není nutné definovat celou posloupnost hran, stačí pouze vstupní a výstupní bod, simulace sama vybere cestu s nejmenším možným počtem hran.

Je tedy potřeba vymyslet způsob, jak oba přístupy zkombinovat do takového přístupu, kde bude možné adresovat konkrétní vozidlo, ale kde nebude potřeba hledat cesty pro všechny kombinace vstupních a výstupních bodů. To by byl problém, který by mohl být náročný po implementační a výkonové stránce - zvláště pro větší počet křižovatek.

Nakonec jsem pro řešení tohoto problému rozhodl využít nástroj `duarouter` (3.5.2), který provádí konverzi `flow` elementů v konfiguraci na `route` elementy. Bude však potřeba upravit jeho výstup, neboť tento nástroj do výstupního souboru také specifikuje konkrétní vozidla pro simulaci. Ty však já nepotřebuji, neboť simulace bude teoreticky nekonečná (nelze ji tedy prostým výčtem stejně obsáhnout). Vozidla se budou do simulace vkládat za běhu pomocí rozhraní TraCI na základě definované hustoty vozidel. Tyto údaje z výstupního souboru tedy odstraním.

Kapitola 4

Implementace simulátoru

V této kapitole se budu věnovat popisu struktury výsledné simulační aplikace a také popisu základní logiky programu.

Konkrétní činnost programu se odvíjí od přijatého požadavku z klientské aplikace. Tyto požadavky lze rozdělit do dvou oblastí.

- Přípravná fáze
- Simulační fáze

Do přípravné fáze patří operace, které manipulují s elementy jako jsou křižovatky nebo dopravní pruhy na křižovatce, které se v průběhu simulace nemění. Do simulační fáze patří požadavky na provedení simulačních kroků a případné další operace prováděné za běhu simulace (jako třeba přepnutí signálního programu). Jediná operace vykonávaná při startu simulační aplikace je vytvoření nové prázdné instance třídy `Situation`, která reprezentuje simulační situaci.

4.1 Struktura projektu

Simulátor byl implementován v programovacím jazyce Java verze 8 jako SpringBoot aplikace (3.2.1) ve verzi 2.0.0. Implementace byla provedena dle původního návrhu (3.2). Stejně jako v návrhu poté také existuje 5 modulů `TrafficSimulatorService`, `Simulator`, `Parser`, `Generator` a `Shared`. Pro sestavení, získání externích závislostí a běh projektu využívám nástroj Gradle[1] ve verzi 4.5.1.

4.1.1 Modul `TrafficSimulatorService`

Tento modul propojuje všechny ostatní moduly a stará se tak o řízení všech operací. Je to spustitelný modul a pro jeho implementaci jsem využil SpringBoot framework(3.2.1), kvůli snadné konfiguraci především HTTP serveru. Tento modul tedy také obsahuje HTTP rozhraní.

4.1.2 Modul `Parser`

Modul `Parser` má pouze jeden úkol, a to zpracování XML souboru s konfigurací křižovatky. Ze zadané konfigurace jsou vyextrahovány nutné informace (odkaz na informace) a uloženy do vnitřní reprezentace.

[1] Gradle: https://gradle.org/

4.1.3 Modul `Generator`

Tento modul má také jeden úkol: z vnitřní reprezentace vygenerovat validní konfigurační soubory pro nástroj SUMO. Využívá k tomu další nástroje ze SUMO distribuce, a to `duarouter` (3.5.2) pro vyhledání cest v dopravní situaci a `netconvert` (3.5.1) pro generování dopravní sítě. Stejně jako v modulu `Parser` se i zde používá knihovna `org.w3c.dom` pro usnadnění práce s XML.

4.1.4 Modul `Simulator`

`Simulator`, jak název napovídá, zajišťuje všechny operace přímo související se simulací. Což znamená její spuštění a zastavení, získávání informací za běhu a zajišťuje požadavky na změnu parametrů simulace taktéž za běhu.

4.1.5 Modul `Shared`

Tento modul obsahuje třídy společné několika modulům. Jedná se mj. o třídy zodpovědné za jednotlivé prvky konfigurace (např. `Intersection, Lane, SignalGroup`), výjimky, enumerátory a konstanty.

4.2 Přípravná fáze

Přípravná fáze sestává ze zpracování vstupů a konfigurací a z generování konfiguračních souborů pro SUMO. Tato fáze se nemusí v průběhu programu vykonat jenom jednou. Vykoná se při prvotním spuštění a poté vždy, pokud se prostřednictvím webového klienta provede jakákoliv změna v zobrazených křižovatkách (např. přidání pruhů, jiné natočení křižovatky). Pokud dojde k takovému požadavku, simulační fáze se zastaví, provede se přípravná fáze a poté se opět spustí simulační fáze na základě aktualizovaných údajů.

4.2.1 Zpracování vstupů a konfigurací

Tato část projektu má na starosti vytvoření počáteční vnitřní reprezentace simulace. Vstupem jí jsou konfigurace řadiče sX ve formátu XML a případné další parametry (poloha přidávané křižovatky, její úhel).

Zpracování XML souborů se provádí pomocí knihovny `org.w3c.dom`, což je základní knihovna jazyka Java. Z XML konfigurace používám především informace o signálních skupinách, signálních plánech a detektorech.

4.2.2 Generátor SUMO konfigurace

Generátor má za úkol z vnitřní reprezentace vytvořit konfigurační soubory pro SUMO. Generované soubory a jejich obsah jsou blíže popsány výše (3.4.1). Generátor využívá ke své činnosti pomocné nástroje platformy SUMO: `netconvert` (3.5.1) pro generování konečného souboru, který obsahuje definici simulační sítě (uzly, hrany a spojení), a nástroj `duarouter` (3.5.2) pro vyhledání cest dopravní situaci. Ten má za úkol na základě definice dopravních toků (`flow`) vytvořit definici tras (`route`). Jím vytvořený soubor pro definici tras však ještě není konečný a po použití programu `duarouter` se ještě upravuje, aby měl mnou požadovaný formát (popsáno v 3.5.2 a v 3.9). Při úpravě tohoto souboru si uložím vygenerované trasy také do vnitřní reprezentace simulace, neboť je budu potřebovat pro generování vozidel.

4.3 Řízení simulace

Fáze řízení simulace nastává po přípravné fázi, kdy máme vygenerované konfigurační soubory a počáteční stav simulace je načtený ve vnitřní reprezentaci.

Řízení simulace spočívá ve spuštění programu SUMO (ať už s vizualizací nebo bez) s vytvořenými konfiguračními soubory a poté následuje spuštění simulace. Simulace spuštěná skrze rozhraní TraCI implicitně nemá definovaný konec, o ten se stará programátor. Běžně se jako ukončovací podmínka může použít například nulový počet vozidel v simulaci, zde simulaci ukončuje uživatel zadáním speciálního příkazu (popsáno dále). Je to tak proto, že v řízení simulace konec nedefinuji, vozidla se generují stále. Dalším vnějším podnětem k ukončení simulace může být taková změna parametrů z webové aplikace, která vyžaduje vygenerování nových konfiguračních souborů pro SUMO.

Simulace potřebuje mít také přístup k vnitřní reprezentaci, a to konkrétně k těmto informacím:

- Signální programy a fáze
- Detektory
- Trasy (vygenerované programem `duarouter`)

Tyto údaje používám ke generování vozidel a k dynamickému řízení dopravy.

4.3.1 Změny v běžící simulaci

Při inicializaci simulace se vždy spustí separátní vlákno, které slouží jako obsluha běžící simulace (instance třídy `SimulationListener`). Příchozí požadavky na změnu v běžící simulaci jsou ukládány do fronty v obluhující třídě, která je každý simulační krok kontrolována a případné akce jsou vykonány. Jsou implementovány tyto akce pro změnu simulace za běhu:

- Změna hustoty provozu
- Změna signálního programu

4.3.2 Implementace dynamického řízení

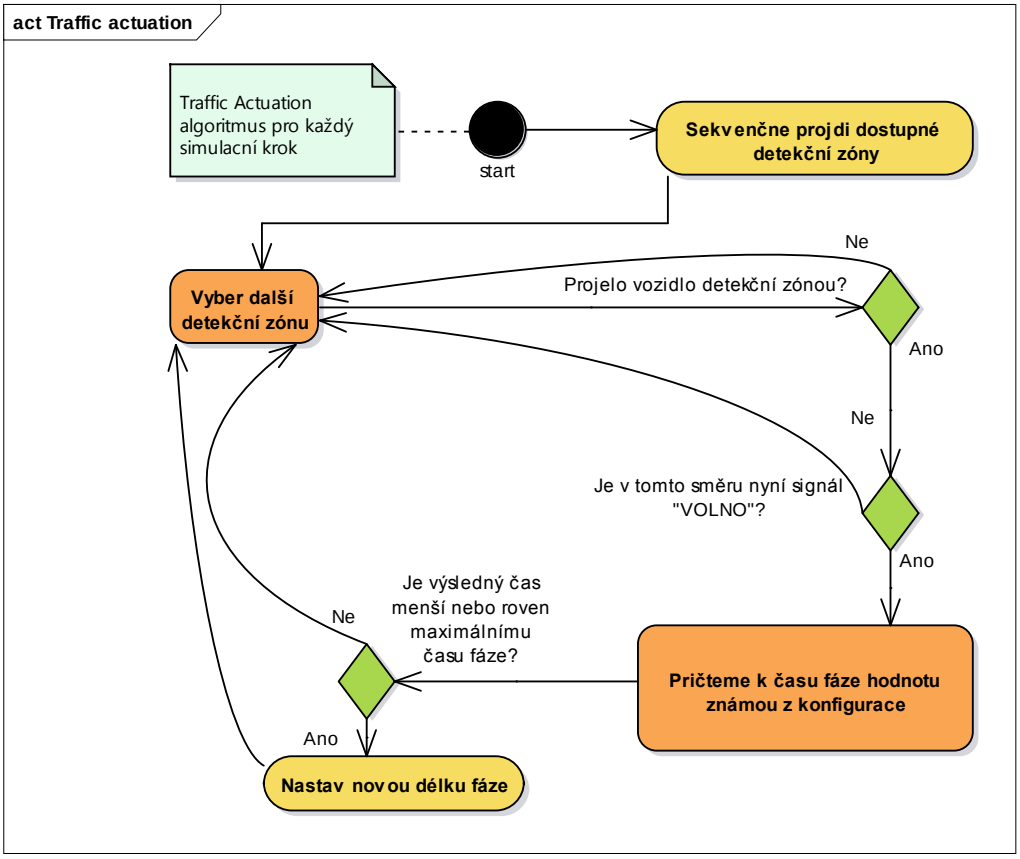
Dopravní toky v simulaci jsou aktivně optimalizovány mechanismem Traffic Actuation, který umožňuje prodloužení aktivní fáze ve směru s hustou dopravou. Aplikace v tomto případě simuluje dopravní řadič tak, jako bylo jeho fungování popsáno v kapitole 2 (2.5.1). Diagram reprezentující tento algoritmus je popsán na obrázku (4.1).

4.3.3 Generování vozidel

Generování vozidel probíhá od startu simulace. Četnost generování vozidel závisí na definované hustotě provozu. V současném stavu se vozidla generují náhodně na vstupních bodech do simulace a je jim náhodně přiřazena jedna z vygenerovaných tras, které začínají v daném vstupu.

Generují se různé typy vozidel. Uživatel si může vybrat z vozidel, která jsou uvedena v dokumentaci SUMO[2]. Poměr typů generovaných vozidel je poté určen uživatelem. Typy

[2]Kompletní výčet typů je uveden v dokumentaci projektu SUMO



Obrázek 4.1: Algoritmus pro Traffic Actuation použitý v simulátoru [vlastní]

vozidel se mohou lišit svou velikostí, rychlostí, zrychlením / brzděním, nebo například znečištěním ovzduší emisemi.

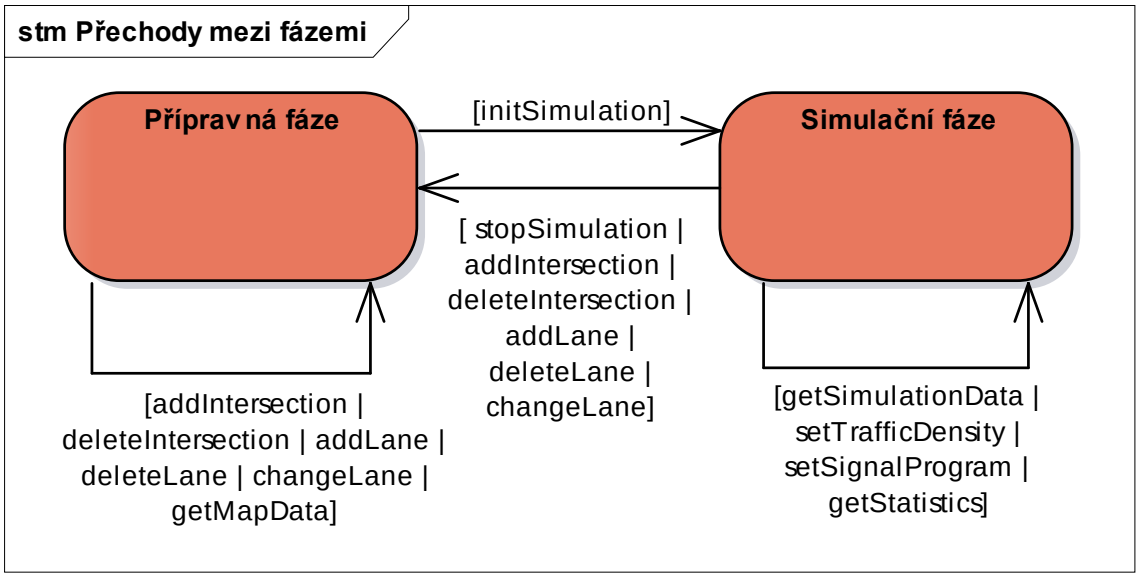
4.4 Komunikační rozhraní

V této sekci podrobněji popíšu implementované HTTP rozhraní. Až na jeden případ jsou všechny dostupné metody založeny na HTTP metodě GET. Všechny odpovědi na požadavky klienta jsou instance třídy `JsonMessage`, která obsahuje číselný kód odpovědi, stavový text a případně také objekt s daty.

Na obrázku 4.2 je ilustrován vliv jednotlivých metod rozhraní na vnitřní stav simulační aplikace.

4.4.1 Přípravná fáze

Přípravná fáze se vtahuje k procesu budování statického podkladu vizualizace, tedy dopravní situace, na které bude následně provedena simulace. Jedná se tedy především o manipulaci s křižovatkami a dopravními pruhy.



Obrázek 4.2: Metody rozhraní a jejich vliv na interní stav simulační aplikace [vlastní]

getMap

Tato metoda poskytuje klientovi všechny podstatné údaje, které jsou potřebné k vykreslení statického podkladu vizualizace. Po přijetí požadavku se z vnitřní reprezentace získají informace o všech křižovatkách a dopravních pruzích. Dále se ze simulace pro tyto prvky získají geometrická data pro vykreslení přesných tvarů křižovatek a dopravních pruhů. Tím pádem odpadnou složité výpočty geometrie těchto prvků na straně klienta, které nejsou triviální. Metoda `getMap` nevyžaduje žádné parametry.

addIntersection

Metoda pro přidání křižovatky je jediným zástupcem HTTP metody POST v tomto rozhraní. Jejím parametrem je instance třídy `JsonRecievedConf`, která obsahuje následující pole:

- Booleovská hodnota značící, jestli se jedná o první křižovatku v situaci nebo nikoliv. Podle její hodnoty se odvíjí další důležité parametry.
- Pokud se jedná o první křižovatku, zajímá nás následující:
  - Typy vozidel, které se budou v následné simulaci generovat.
- Pokud se jedná o další křižovatku, zajímá nás následující:
  - Identifikátor vybrané křižovatky, v tomto kontextu se jedná o referenční křižovatky, ke které se přidává sousední křižovatka.
  - Relativní pozice nově přidávané křižovatky k vybrané křižovatce.
- Úhel natočení křižovatky oproti konfiguraci, který je zadaný uživatelem.

- Obsah XML souboru s konfigurací.

Nejprve se při přidávání křižovatky zjistí, zda se jedná o první křižovatku v situaci, nebo nikoliv. Pokud ano, poloha nové křižovatky bude ve zjednodušeném souřadnicovém systému na pozici $[0, 0]$. V opačném případě se jeho poloha vypočítá podle polohy vybrané křižovatky a dle k ní zadané relativní pozice. Pozice již nesmí být obsazená jinou křižovatkou. V případě přidání na existující místo se v odpovědi klientské aplikaci vrátí patřičná chybová hláška.

Konfiguraci nové křižovatky potom zpracuje modul `Parser`. Pokud byl uživatelem zadán úhel natočení, přičte ke každému ramenu tento úhel. Zpracovaná křižovatka se poté přidá do situace. Na základě dříve specifikovaných pravidel se poté automaticky propojí nebo nepropojí se sousedními křižovatkami. Jakmile je jasná celková topologie situace, modul `Generator` vygeneruje SUMO konfigurační soubory pro aktualizovanou situaci. Pokud už konfigurační soubory existovaly, jsou přepsány.

deleteIntersection

Parametrem této metody je identifikátor křižovatky, kterou chce klientská aplikace vymazat ze situace. Pokud křižovatka existuje, vymaže se z vnitřní reprezentace a následně modul `Generator` vygeneruje nové konfigurační soubory pro SUMO.

addLane

Tato metoda slouží pro přidávání nového dopravního pruhu. Důležitými parametry metody jsou identifikátory křižovatky a jejího ramene a uživatelem definované směry nového pruhu. Uživatel nemůže specifikovat pořadí přidávaného pruhu. To je vypočítáno automaticky na základě dříve specifikovaných pravidel, kdy se všechny pruhy seřadí podle jejich směrů. Pokud má nově přidaný pruh specifikované takové změny, které nejdou v rámci bezpečného provozu dohromady s ostatními, pruh se nepřidá a klientské aplikaci se vrátí chyba (takovým případem jsou třeba dva pruhy, ze kterých lze odbočit do všech směrů).

Po přidání pruhu se opět přegenerují konfigurační soubory pro simulaci.

changeLane

Zde se jedná pouze o změnu odbočovacích směrů z konkrétního dopravního pruhu. Parametry jsou identifikátor dopravního pruhu a nově specifikované směry. Poté opět proběhne seřazení pruhů dle jejich směrů, takže se může stát, že tento pruh změní svou pozici. Nehledě na to se opět přegenerují konfigurační soubory, neboť se změnily směry a tím pádem i možné cesty pro vozidla v simulaci.

deleteLane

Parametrem této metody je identifikátor dopravního pruhu. Protože není možné nemít alespoň jeden dopravní pruh v ramenu křižovatky, před jeho smazáním se provede kontrola na tuto skutečnost. Po provedení metody se přegenerují konfigurační soubory.

clearSituation

Tato metoda smaže všechny křižovatky ze simulace.

4.4.2 Simulační fáze

Následující metody se vztahují k simulační části. Hlavní rozdíl oproti přípravné fázi je ten, že se již nevyužívají moduly `Parser` a `Generator`, ale začíná se využívat modul `Simulator`.

initSituation

Tato metoda provádí inicializaci simulace. Ověří se, že je simulační situace schopná simulace kontrolou počtu křižovatek a počtu validních cest v simulaci. V případě, že je jeden z těchto parametrů nulový, situace není připravena na simulaci a klientské aplikaci se vrátí patřičná chybová hláška. Pokud je vše v pořádku, inicializuje se spojení s TraCI rozhraním a spustí se proces programu SUMO.

getSimulationData

Metoda `getSimulationData` vyžaduje číselný parametr určující počet simulačních kroků, které má simulátor vykonat. V každém simulačním kroku se provede následující posloupnost činností:

- Zkontroluje se, zda proces třídy `SimulationListener` nehlásí nějakou akci, kterou je potřeba v rámci simulačního kroku udělat. Příkladem takové akce je změna signálního plánu.
- Provede se dynamické řízení dopravy pomocí Traffic Actuation (viz 2.5.1).
- Na základě aktuálního nastavení hustoty provozu se provede generování nových vozidel.
- Uložení dat o simulaci pro pozdější analýzu.

V každém simulačním kroku se navíc ukládají data, která se po vykonání klientem definované sekvence simulačních kroků odešlou klientské aplikaci. Tato data se sestávají z následujících částí:

- Nové souřadnice a úhel každého vozidla v rámci simulačního kroku. Kromě těchto základních parametrů pro vizualizaci jsou zde také obsažena on-line data o rychlosti, ujeté vzdálenosti a aktuálnímu čekacímu času.
- Statická data o nově přidaných vozidlech (barva, velikost, typ).
- Informace o aktuálním stavu signálních skupin. Tyto informace jsou roztříděny tak, že ke každému dopravnímu pruhu je přiřazen aktuální signál příslušné signální skupiny.
- On-line informace o simulaci (simulační krok)
- On-line informace o každé křižovatce (signální program, fáze)

setTrafficDensity

Metoda `setTrafficDensity` mění na základě číselného parametru četnost generování vozidel. Mění se tedy počet vozidel vygenerovaných za jeden simulační krok a případně také, jak často se vozidlo generuje (pro nižší hodnoty hustoty). Tato metoda využívá paralelního vlákna instance třídy `SimulationListener`, která hlavnímu vláknu reprezentovanému instancí třídy `SiulationRunner` předává informace o akci, kterou je potřeba provést.

setSignalProgram

Tato metoda umožňuje on-line změnu signálního plánu vybrané křižovatky. Parametrem této metody je právě identifikátor vybrané křižovatky a řetězec identifikující nový signální plán. Tato metoda také využívá paralelní vlákno `SimulationListener`.

stopSimulation

Metoda slouží pro restartování simulace do počátečního stavu před inicializací.

getStatistics

Získání nashromážděných simulačních dat za celý běh simulace.

4.5 Konfigurace aplikace

Aplikace poskytuje možnosti nastavení pomocí souboru `application.properties`, který slouží jako externího konfigurace. Hlavní důvod pro parametrizování aplikace právě pomocí konfiguračního souboru a nikoliv pomocí argumentů zadávaných při spuštění v příkazové řádce byla povaha některých parametrů.

Parametry, které může uživatel nastavit, jsou:

- Cesta ke složce pro ukládání vygenerovaných konfiguračních souborů pro SUMO.
- Délka simulačního kroku - tato hodnota bude již přednastavena na přijatelnou hodnotu s ohledem na faktory definované dříve (3.6.1), avšak bude možné ji měnit dle uvážení uživatele.
- Maximální počet najednou přítomných vozidel v simulaci.
- Port pro HTTP komunikaci.

4.5.1 Ošetření kolizí

Zjistil jsem, že za určitých podmínek vznikají kolize. Kolize u tohoto modelu znamená, že se vozidla dostala k sobě na vzdálenost menší než je jejich modelem definovaný minimální rozestup. Kolize tedy nemusí nutně znamenat srážku vozidel. Problémem je, že SUMO reaguje na kolize teleportací vozidel, které kolizi zaviní. Teleportace znamená, že se vozidlo přesune na první další neobsazené místo na jeho trase. Což není chování, se kterým bychom se setkávali v reálném provozu. Vždy při spuštění simulace je tedy procesu SUMO předán speciální parametr `collision.action` nastavený na hodnotu `warn`. Kolize jsou zaznamenány, avšak k teleportaci již nedochází.

4.6 Generování dat pro výstup simulace

V modulu `Simulator` se po dobu běhu simulace shromažďují data o jejím průběhu. Tato data jsou především o vozidlech a snaží se sbírat taková data, aby byla vhodná pro jejich analýzu celé situace a usnadnila tak vývoj např. signálních plánů. V současné době se jedná o data o spotřebě paliva, době čekání na semaforech, době strávené v simulaci a data

o vypuštěných emisích (CO, CO_2, NO_x, PM_x a HC)[3]. Nasbíraná data se dají rozdělit do tří celků.

Prvním celkem jsou informace o všech vozidlech v rámci jednoho simulačního kroku. Tím druhým poté souhrnné informace nasbírané za dobu vozidla v simulaci. Posledním jsou absolutní stavy všech měřených parametrů všech vozidel za celou dobu trvání simulace.

[3] CO: oxid uhelnatý, CO_2: oxid uhličitý, NO_x: oxidy dusíku, PM_x: pevné částice (particulate matter), HC: uhlovodík.

Kapitola 5

Návrh klientské aplikace

V této fázi je tedy již simulátor s komunikačním rozhraním navrhnut a implementován. Nyní se zaměřím na popis návrhu aplikace pro vizualizaci, která bude zároveň i klientem simulátoru. Aplikace bude v této práci koncipována jako webová stránka. Snažil jsem se však simulátor navrhnout tak, aby nebyl nijak svázaný s žádnou konkrétní verzí klientské aplikace.

5.1 Komunikace se simulátorem

Jak je již známo z předchozí kapitoly, komunikační rozhraní simulátoru přijímá HTTP požadavky ve formátu JSON. Ve stejném formátu také na požadavky odpovídá. Komunikaci vždy iniciuje klient.

5.1.1 Základní komunikace

Komunikace v přípravné fázi bude vždy probíhat následujícím způsobem:

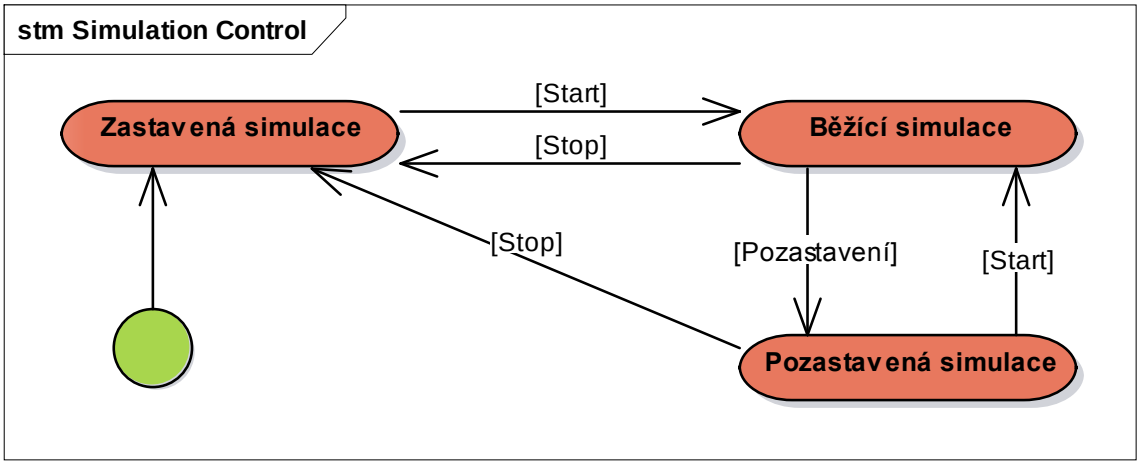
1. Po načtení stránky je zaslán požadavek na získání dat pro podklad simulace.
2. Je provedena změna uživatelem nad podkladem (dopravní situací).
3. Požadavek na změnu je zaslán simulační aplikaci.
4. Pokud simulační aplikace zašle odpověď, že je vše v pořádku, zašle se požadavek na získání nových dat pro podklad simulace.

5.1.2 Řízení simulace

Princip změny stavu simulace z pohledu klientské aplikace je znázorněn na obrázku 5.1. Mohou nastat 3 stavy simulace:

- běžící simulace,
- pozastavená simulace a
- zastavená simulace

Běžící simulace je inicializovaná pomocí příkazu `initSimulation` a získává data pomocí `getSimulationData`. Zároveň se simulační data zpracovávají a animují. Požadavek na



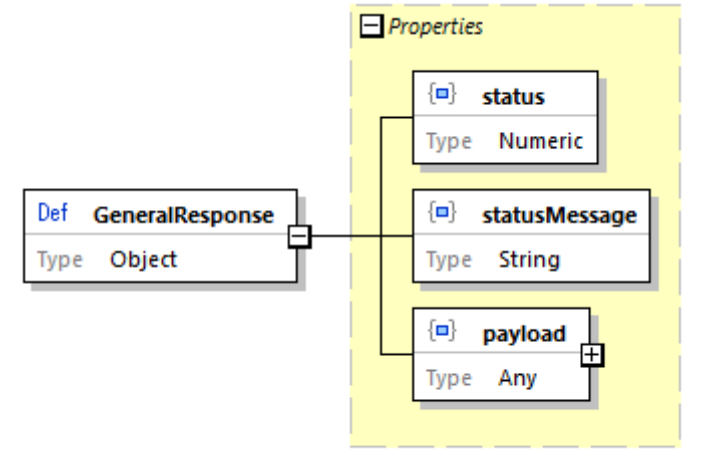
Obrázek 5.1: Změny simulačních stavů v klientské aplikaci [vlastní]

simulační data se zasílá vždy, jakmile počet simulačních kroků ke zpracování klesne pod určitou úroveň.

Pozastavená simulace je inicializovaná, ale nezískává aktivně data. Animace je taktéž pozastavena, ale prvky (vozidla a semafory) zůstávají v posledním známém stavu. Po přechodu do stavu běžící simulace nedojde ke ztrátě dat, animace bude pokračovat tam kde skončila.

Zastavená simulace není inicializovaná. Nezískávají se data a neprobíhá animace. V takovém stavu se simulace nachází vždy po změně podkladu, nebo po explicitním zastavení příkazem `stopSimulation`.

5.2 Data přijímaná klientem



Obrázek 5.2: Základní JSON schéma [vlastní]

Jak ilustruje obrázek (5.2), každá odpověď ze simulátoru obsahuje vždy pevnou strukturu. Obsahem odpovědi je vždy jednoduchý číselný kód `status` reprezentující stav požadavku:

- 0: Požadavek zpracován korektně.
- 1: Požadavek nezpracován kvůli chybě na vstupu nebo kvůli interním omezením.
- 2: Chyba v simulátoru.

Pole `statusMessage` obsahuje bližší popis nastalé situace. Pokud je pole `status` rovno 0, pole je prázdné.

Posledním polem v této hlavičce je pole reprezentující samotnou strukturovanou odpověď, pokud je to z kontextu požadavku potřeba. To je pouze ve dvou případech, které jsou popsány v dále (viz 5.2.1 a 5.2.2).

5.2.1 Data pro podklad simulace

Při popisu této a následující podkapitoly se budu opírat o grafickou reprezentaci JSON schémat, která jsou uvedena v příloze A.1.

Data pro vykreslení podkladu jsou členěna do pěti částí (A.1). První částí jsou tzv. `metadata` (A.2), kde se uvádí důležité obecné údaje o simulaci, např. použitá délka simulačního kroku, počet křižovatek, nebo velikost podkladu.

Pole `intersectionList` (A.4) obsahuje seznam křižovatek aktuálně obsažených v simulaci. Každá křižovatka je pak kromě svého identifikátoru dále definovaná následujícími parametry:

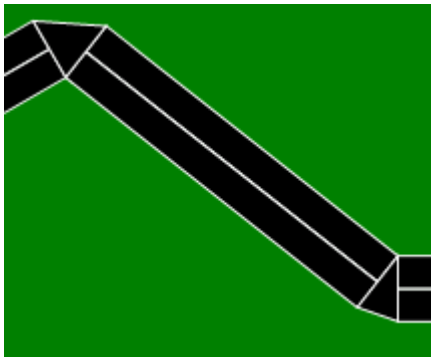
- `shape` - Popisuje tvar centra křižovatky sekvencí souřadnic (A.8).
- `gridPosition` - Pozice křižovatky na mapě.
- `legList` - Seznam ramen křižovatky (A.3).
- `signalProgramList` - Seznam unikátních jmen signálních programů definovaných pro křižovatku.
- `angle` - Úhel natočení křižovatky oproti originální konfiguraci (zadaný uživatelem při vkládání).

Každé rameno v seznamu ramen křižovatky pak dále obsahuje další informace, jako úhel ramene a seznam dopravních pruhů. Dopravní pruh (A.5) poté také obsahuje informace, zejména o svém tvaru. Pokud se jedná o vstupní pruh, je zde také přítomna informace o směrech, kterými lze po opuštění pruhu pokračovat (reprezentují šipky v dopravních pruzích v reálných křižovatkách).

Část `connectionLegs` (A.6) obsahuje seznam ramen spojujících dvě křižovatky. Obsahuje především opět informaci o svém tvaru (`shape`) a také konkrétně která ramena spojuje.

V další části `connectionPolygons` (A.7) jsou pouze uvedeny souřadnice polygonů, které v mapě vyplňují místa mezi rameny křižovatek a propojovacími rameny. Pro ilustraci přikládám obrázek (5.3).

Poslední sekce `alreadyExistingVehicles` (A.13) obsahuje statická data o vozidlech, která jsou v momentu zaslání požadavku přítomná v simulačním prostředí. Pokud zahajujeme novou simulaci, je tato část odpovědi prázdná. Slouží tedy především pro ty případy, kdy znovu pouštíme již rozběhnutou simulaci (např. po pozastavení simulace, nebo po obnovení stránky).



Obrázek 5.3: Ilustrační ukázka vyplňovacích polygonů. [snímek obrazovky]

5.2.2 Simulační data

Jak je již blíže popsáno v jiné podkapitole (5.1.2), data s údaji o jednotlivých simulačních krocích se posílají ve shlucích. Simulační data tedy nejsou nic jiného než pole struktur reprezentujících jednotlivé simulační kroky (A.9). Jako identifikátor kroku slouží simulační čas, který byl aktuální v době provedení tohoto kroku. Dále se dají informace opět rozčlenit do logických celků.

Prvním takovým celkem v rámci simulačního kroku jsou on-line data o vozidlech v simulaci (A.10). Zde se nachází množství parametrů důležitých pro vizualizaci:

- `coords` - Souřadnice, na kterých se nacházel referenční bod vozidla po ukončení simulačního kroku. Referenční bod se konkrétně nachází na přední hraně vozidla uprostřed.
- `angle` - Úhel vozidla, pomocí kterého můžeme dopočítat krajní body celého vozidla (spolu se známou délkou a šířkou vozidla).
- `signaling` - Tento parametr je ve formě bitové masky. Jednotlivé bity určují, které světlo vozidla je právě aktivní (např. brzdy, směrovky).

Jsou zde obsaženy také další parametry, které však již nejsou důležité pro vizualizaci ale slouží spíše pro informaci. Jedná se o rychlost, ujetou vzdálenost a aktuální dobu čekání na semaforu. SUMO simulátor a potažmo rozhraní TraCI však poskytuje těchto parametrů mnohem více (např. aktuální/celkové emise vozidla), je tedy snadno možné tento výčet rozšířit.

Další částí jsou informace o současných signálech na semaforech v simulaci (A.11). Každému dopravnímu pruhu je zde přiřazen jeden znak reprezentující barvu.

Následují on-line informace o křižovatce (A.12). Tato data nejsou důležitá pro vizualizaci a stejně jako např. rychlost u vozidel slouží pro informaci.

Aby se ušetřila režie zpracování na klientovi, jsou zvlášť vyvedena statická data vozidel (A.13), která jsou do simulace v konkrétním simulačním kroku přidána. Jedná se tedy o informace, které se za životní cyklus vozidla zpravidla nemění (šířka, výška, barva).

5.3 Vizualizace

Zde se dostáváme k problematice zobrazování vizualizovaných dat. Vizualizovaná data můžeme logicky rozdělit na data, které se v průběhu simulace nemění (statická) a na ta která se mění (dynamická).

5.3.1 Statická vizualizace

V této části se jedná o vykreslení prvků, jež se nijak v průběhu simulace nemění, a není tedy potřeba je nijak animovat. Mezi tyto prvky budou tedy patřit křižovatky, jejich ramena a potažmo dopravní pruhy a spojovací pruhy. Můžeme zde také zařadit statickou část semaforů beze změny barvy.

5.3.2 Animovaná vizualizace

Zde se budeme zabývat prvky, které se animují:

- Vozidla - pohyb, signalizace
- Semafory - signalizace

Animace signalizace semaforů spočívá ve vypínání a zapínání příslušných světel na semaforu v závislosti na aktuální informaci.

Animovaná vizualizace vozidel již bude složitější, neboť bude potřeba kontinuálně přidávat, odebírat a pohybovat s desítkami až stovkami elementů. Složitost zde spočívá právě v počtu vozidel. Jejich animace samotná se skládá pouze z pohybu elementu na nové souřadnice a případné vizualizaci jejich signalizace, což v případě jednoho vozidla samo o sobě problém není. Bude tedy potřeba vymyslet takový postup a zvolit takové nástroje, aby se animace těchto vozidel nestala úzkým hrdlem aplikace.

5.4 Nástroj pro vizualizaci

Klientská aplikace ve své vizualizaci může potřebovat zobrazit desítky až stovky pohybujících se objektů. Potřebujeme tedy použít vhodnou technologii pro velké množství jednoduchých animací.

Vybíral jsem si mezi dvěma technologiemi, a to CSS a SVG. CSS animace bodují svou relativní jednoduchostí. Avšak jakmile je potřeba řetězit za sebou více animací, stává se tato technologie velmi komplikovanou. Po zvážení různých zdrojů jsem se rozhodl využít technologii SVG ve spolupráci s jazykem JavaScript.

5.4.1 Vhodné knihovny pro práci s SVG

Pro využití SVG v jazyce JavaScript jsem chtěl využít jednoduchou knihovnu, která by podporovala rychlé zpracování velkého množství objektů. Nalezl jsem dvě knihovny, které tyto vlastnosti splňují:

- Svg.js[1]
- Snap.js[2]

[1] Svg.js: http://svgjs.com/
[2] Snap.js: http://snapsvg.io/

5.5 Návrh uživatelského rozhraní

Většinu prostoru by měla zabírat vizualizace simulace. Ovládání aplikace by se nacházelo v panelu při některém z okrajů vizualizačního plátna. Případné zadávání dat se bude realizovat pomocí modálních dialogových oken.

5.5.1 Ovládání aplikace

Aplikace bude muset poskytnout dostatečně funkční ovládání pro tyto úkony:

- Přidání/Smazání křižovatky.
- Přidání/Změna/Smazání dopravního pruhu.
- Specifikace typu a hustoty dopravy.
- Ovládání simulace: start, pozastavení, zastavení.
- On-line změna signálního programu dané křižovatky.
- Návrat do počátečního stavu.

Přidání křižovatky do simulace je nezbytným úkonem při tvorbě požadované situace pro simulaci. Na druhou stranu, práce s dopravními pruhy přímo nezbytná není. Avšak jak již bylo vysvětleno při návrhu simulátoru (viz 3.7), dopravní pruhy nejsou v konfiguraci přímo specifikovány. Tento úkon slouží tedy především k dotvoření křižovatky k reálnějšímu obrazu.

Co se týče ovládání simulace, tak sémantika jednotlivých akcí je následující:

- Start: Počátek získávání dat ze serveru a počátek vizualizace.
- Pozastavení: Animace se zastaví, stejně jako získávání dat. Vozidla zůstanou vykreslená a po provedení úkonu start se bude pokračovat od tohoto bodu.
- Zastavení: Získávaní dat se zastaví, vozidla ze simulace zmizí. Po provedení úkonu Start se začne vše od začátku.

Zároveň jakákoliv změna simulované situace ve smyslu přidání či odebrání křižovatky nebo manipulace s dopravními pruhy, se projeví stejně jako zastavení simulace.

Návrat do počátečního stavu kombinuje zastavení simulace a smazání všech křižovatek ze simulace.

5.5.2 Interaktivní mapový podklad

V klientské aplikaci bude možné vybrat konkrétní křižovatku kliknutím na její střed. Takto vybraná křižovatka se zvýrazní. V postranním panelu se poté zobrazí volba pro její smazání. Při běžící simulaci také v postranním panelu uvidí on-line informace o této křižovatce.

Obdobná funkcionalita bude fungovat pro dopravní pruhy. Po kliknutí na některý ze vstupních dopravních pruhů se zobrazí volby pro manipulaci s dopravními pruhy popsané výše.

Nakonec se bude stejným způsobem vybírat konkrétní vozidlo, což zobrazí jeho on-line informace.

5.5.3 Zadávání dat

Zadávání složitějších dat (tj. více než jednu hodnotu) bude realizováno jednoduchými formuláři v modálních oknech. Bude tedy potřeba vytvořit obsah těchto formulářů pro:

- Přidání křižovatky: nahrání souboru s konfigurací, zadání úhlu pro natočení křižovatky oproti konfiguraci a specifikace polohy křižovatky.
- Přidání nebo změna dopravního pruhu: zde se specifikují pouze směry. Poloha dopravního pruhu na křižovatce je určena automaticky na základě směrů všech pruhů (odkaz).
- Specifikace typů vozidel v simulaci: výběr z předdefinovaných typů.

Specifikaci hustoty dopravy v celé simulaci lze realizovat nejintuitivněji pravděpodobně posuvníkem. Ovládací prvek pro změnu signálního programu potom může být umístěn přímo u on-line informací týkajících se konkrétní křižovatky.

Kapitola 6

Implementace webového klienta

Webový klient je implementován v jazycích HTML, CSS a JavaScript, a to opět z důvodu jejich jednoduchosti. Pro spuštění aplikace a její správnou funkčnost není potřeba mít funkční internetové připojení za předpokladu, že je na příslušné pracovní stanici spuštěn náš simulační server. Při implementaci webového klienta jsem vycházel z návrhu použitého v předchozí kapitole.

6.1 Použité nástroje a knihovny

Rozhodl jsem se využít novějších funkcionalit standardu ECMA, který od verze 6[1] podporuje definici tříd. Využití tříd mi umožní vytvořeni čitelnějšího a lépe strukturovaného kódu. Jako nevýhoda se může jevit skutečnost, že tento standard nemá podporu ve starších verzích prohlížečů.

Pro implementaci animací jsem využil technologii SVG a pro snadnější práci s ní jsem použil knihovnu Svg.js pro jazyk JavaScript. Důvod, proč jsem nakonec nepoužil nástroj Snap.js byl ten, že Svg.js má kompletní dokumentaci a je rychlejší.

Pro vykreslení grafů ve výstupu simulace jsem využil knihovnu Chart.js pro jazyk JavaScript. Důvodem je opět vysoká míra použitelnosti při přijatelně jednoduchém používání. Při volbě této knihovny jsem specifické požadavky na výkon neměl.

Nakonec pro usnadnění práce s HTML strukturou a pro zasílání HTTP požadavků využívám známou knihovnu jQuery[2].

6.2 Adresářová struktura

Klientská aplikace byla vyvíjena odděleně od vývoje simulační aplikace. V kořenovém adresáři se nachází soubor `index.html` a následující složky:

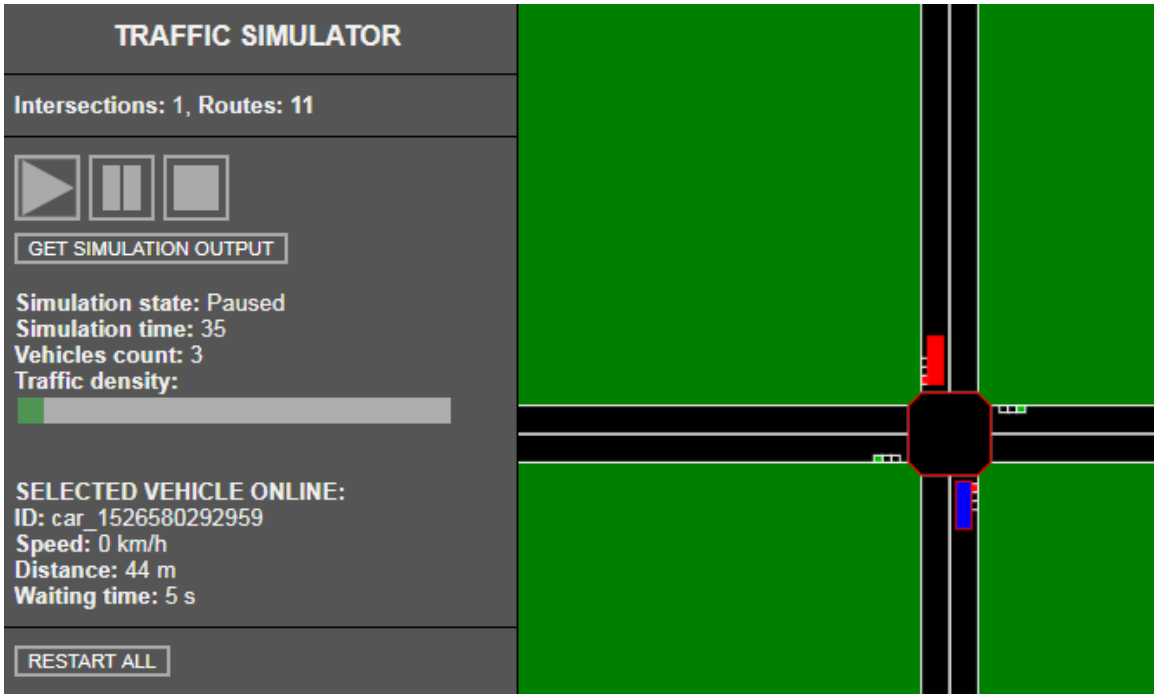
- `css` obsahující soubor s kaskádovými styly pro definici vzhledu jednotlivých prvků,
- `img` obsahující ikony,
- `js` se soubory v jazyce JavaScript a
- `lib` s externími JavaScript knihovnami.

[1] ECMAScript 6 (ES6), nebo tak0 ECMAScript 2015 (ES2015)
[2] jQuery: https://jquery.com/

Zdrojové soubory v jazyce JavaScript se dále dělí na soubory obsahující definici tříd (`classes`) a na další zdrojové soubory.

6.3 Uživatelské rozhraní

Jak je vidno na obrázku 6.1, uživatelské rozhraní můžeme opticky rozdělit na dvě části. Vlevo je vidět postranní panel, který má pevnou velikost a obsahuje ovládací prvky s informacemi o simulaci. Ve zbytku okna prohlížeče se nachází vizualizace simulační situace.



Obrázek 6.1: Ilustrační ukázka uživatelského rozhraní [vlastní, snímek obrazovky]

Ovládací prvky se zobrazují nebo skrývají podle aktuální situace. Další obrázky uživatelského rozhraní jsou obsaženy v příloze B.

6.4 Operace v přípravné části

Operacemi v přípravné fázi rozumíme přípravu dopravní situace pro simulaci.

6.4.1 Překreslování statické části vizualizace

Překreslení statické části vizualizace probíhá vždy, pokud se provede změna nad statickou částí simulace pomocí webového klienta a na požadavek je odpovězeno kladnou odpovědí. Mezi takové akce patří vkládání nebo mazání křižovatky a vkládání, změna nebo smazání dopravního pruhu. Překreslení statické části neprobíhá při běžící simulaci a vizualizaci. Pokud uživatel provede změnu situace v průběhu simulace, před odesláním požadavku na změnu situace na server se pošle tamtéž nejprve požadavek na zastavení a restartování simulace. Ta tedy poté vždy začíná od začátku, od simulačního času 0.

6.4.2 Operace nad křižovatkami

Nad křižovatkami lze provádět operaci vybrání, operaci vkládání a operaci mazání.

Selekci křižovatky lze provést pouze v situaci, kdy již nějaká křižovatka v situaci existuje. Provádí se kliknutím myši na střed křižovatky. Vybraná křižovatka má svůj střed barevně odlišen. Výběr křižovatky slouží k tomu, aby bylo možné zobrazit on-line informace o ní při běhu simulace a vizualizace v postranním panelu. Ještě důležitějším důvodem pro výběr je ten, že pouze vybranou křižovatku lze smazat, a pouze k vybrané křižovatce lze přidat sousední křižovatku. Při výběru křižovatky se neposílá na server žádný požadavek.

Vložení křižovatky se dělí podle situace, kdy ji přidáváme. Pokud chceme přidat první křižovatku, v modálním okně nahrajeme XML soubor s konfigurací křižovatky, zadáme počet stupňů natočení (pokud je to vyžadováno) a specifikujeme druhy dopravy, které chceme v celé simulaci vidět. Druhy dopravy se specifikují pouze u přidávání první křižovatky, u dalších se tento údaj již nezadává. Pokud již nějaká křižovatka v situaci existuje, je možné přidat další křižovatku jejím vybráním. Následné křižovatky se v této aplikaci vkládají vždy pouze v sousedství již existující křižovatky, a to ve čtyřech základních směrech: nad vybranou křižovatku, pod vybranou křižovatku, a vpravo a vlevo od ní. Po stisknutí potvrzovacího tlačítka se provede základní kontrola formuláře na klientské straně, poté se data odešlou a podobná kontrola se provede také na straně serveru. Tato kontrola spočívá v testování přítomnosti povinných hodnot. Testuje také navíc, zda je místo pro novou křižovatku volné. Server po zpracování požadavku vrátí buď kladnou nebo zápornou odpověď. Po přijetí kladné odpovědi se situace v prohlížeči překreslí, aby reflektovala změny v situaci. Po přijetí odpovědi s chybou se tato chyba zobrazí uživateli.

Smazat lze pouze křižovatku která je vybrána. Na rozdíl od vkládání při mazání není brán ohled na okolní křižovatky. Ty zůstanou tak jak byly před vymazáním. Pokud byla křižovatka propojena s jinými křižovatkami, toto spojení zanikne není provedena žádná akce pro jejich obnovu.

6.4.3 Operace nad pruhy na křižovatkách

Stejně jako nad křižovatkami existují stejné operace nad dopravními pruhy: selekce, vkládání a mazání. Navíc je zde operace pro změnu směrů dopravních pruhů.

Selekce dopravního pruhu funguje na stejném principu jako selekce křižovatek. Pokud se vybere vstupní dopravní pruh, v postranním panelu se zobrazí informace o jeho směrech.

U vstupního dopravního pruhu potom můžeme upravit jeho směry. Případně lze přidat nový vstupní dopravní pruh a určit u něj směry. Vkládání dopravních pruhů funguje podobně jako vkládání křižovatek. Nový dopravní pruh se vloží do stejného ramene křižovatky, ke kterému náleží vybraný dopravní pruh. Také se mu přiřadí stejná signální skupina. Vložení nového pruhu nebo i změna směrů stávajícího pruhu může mít za následek změnu pořadí vstupních dopravních pruhů. Výstupní dopravní pruhy přidávat nebo měnit nelze.

Vybraný vstupní dopravní pruh lze také smazat. Platí však podmínka, že alespoň jeden vstupní dopravní pruh musí v daném rameni zůstat.

Manipulace s dopravními pruhy nemá žádný vliv na propojení křižovatek. Má však vliv na počet možných tras pro vozidla.

6.5 Vizualizace simulace

V této sekci popíšu, jakým způsobem funguje vizualizace a animace dynamické části simulace.

Simulační data získáváme od simulační aplikace ve formě sekvence simulačních kroků (viz 5.2.2). Simulační kroky se ukládají do fronty. Vizualizace poté probíhá následovně:

1. Vybereme první prvek z fronty.
2. Podle informací o signálních skupinách nastavíme světelné signály.
3. Pokud jsou v simulačním kroku obsažena nová vozidla, zobrazí se na zadaných souřadnicích.
4. Pro již existující vozidla se vytvoří animace o délce trvání stejné jako je délka simulačního kroku (viz 6.6.1). Počátečním bodem nechť jsou původní souřadnice a koncovým nové souřadnice.
5. Po uplynutí doby odpovídající délce simulačního kroku se proces opakuje.

6.5.1 Vizualizace vozidel

Ze simulační aplikace získáváme pro vozidlo jediné souřadnice, které reprezentují bod uprostřed přední hrany vozidla ve směru jízdy. Pomocí získaného úhlu, délky a šířky jsou dopočítány okrajové body vozidla, které je tak vizualizováno jako obdélník. U animovaného vozidla je také patrná jejich signalizace, a to ve formě brzdových a směrových indikátorů. Vozidlo je přidáno do vizualizace při jeho prvním výskytu v rámci simulačního kroku. Odstraněno je, pokud v dalším simulačním kroku již není pro toto vozidlo záznam.

6.5.2 Vizualizace semaforů

Semafory nejsou na rozdíl od vozidel do simulace přidávány dynamicky. Vytvoří se automaticky pro každý vstupní dopravní pruh. V průběhu procházení simulačními kroky se pouze mění signály semaforů dle přijatých dat.

6.6 Řízení simulace z webového klienta

Řídit simulaci lze jednak změnou jejího stavu, dále lze potom měnit a získávat informace o specifických prvcích on-line.

6.6.1 Volba délky simulačního kroku

Jak jsem již popsal dříve (3.6.1), volba simulačního kroku závisí na několika faktorech. Po testování v průběhu implementace jsem usoudil, že základní délka simulačního kroku (1) nestačí pro dostatečně přesnou vizualizaci. Nakonec jsem po zhodnocení všech faktorů vybral hodnotu délky simulačního kroku odpovídající $0.5s$.

6.6.2 Spuštění simulace

Simulace se spustí kliknutím na příslušné tlačítko "Start simulation". Po kliknutí se zašle požadavek na serverovou aplikaci pro inicializaci simulace. Po vrácení kladné odpovědi se začnou periodicky zasílat požadavky na provedení simulačních kroků a získání simulačních dat. Počet provedení simulačních kroků je nastaven na 5. Pokud délka fronty simulačních kroků ke zpracování klesne pod určitou úroveň (nastaveno na hodnotu 3), je zažádáno o další dávku simulačních dat. Současně s tímto probíhá na popředí vizualizace simulace.

6.6.3 Pozastavení simulace

Po kliknutí na tlačítko "Pause simulation"u běžící simulace se po ukončení animace nejbližšího simulačního kroku vizualizace zastaví. Rovněž se přestanou zasílat požadavky na další simulační data. Vozidla po ukončení animace zůstanou na svých pozicích v dopravní situaci, rovněž stavy semaforů zůstanou stejné. Po uvedení aplikace do stavu běžící simulace je animace plynule spuštěna od posledního animovaného simulačního kroku.

6.6.4 Zastavení simulace

Po kliknutí na tlačítko "Stop simulation"u běžící nebo pozastavené simulace se ze situace odeberou všechna vozidla a semafory se nastaví do vypnutého stavu. Po opětovném stisknutí tlačítka "Start simulation"simulace započne od začátku.

6.6.5 On-line informace v postranním panelu

V postranním panelu se zobrazují tyto typy informací o právě běžící nebo pozastavené simulaci:

- stav simulace (stav, simulační krok, počet vozidel),
- stav vybrané křižovatky (fáze, signální program),
- a stav vybraného vozidla (rychlost, ujetá vzdálenost, čekací doba na semaforech).

Vozidlo lze vybrat stejně jako lze vybrat křižovatku nebo dopravní pruh - kliknutím na vozidlo. Vybrané vozidlo je od ostatních odlišeno svou barvou, která je sjednocena s ostatními vybranými prvky.

6.6.6 Změna hustoty generované dopravy

Při běžící simulaci je v postranním panelu pomocí posuvníku možnost nastavit hustotu dopravy. Nastavuje se budoucí hustota, čili rychlost generování nových vozidel. Se současnými vozidly v simulaci se změnou hustoty neprovede nic. Každá změna posuvníku značí odeslání požadavku na změnu hustoty na simulační aplikaci. Projevení změny může trvat až několik sekund kvůli použitému mechanismu získávání simulačních dat (popsáno v 6.6.2).

6.6.7 Změna signálního programu vybrané křižovatky

Pokud je vybraná nějaká křižovatka v běžící simulaci, je možné ji změnit signální program. Změna se provede vybráním příslušného signálního programu ze seznamu. Po kliknutí na novou položku se odešle požadavek na změnu signálního programu dané křižovatky na

simulační aplikaci. Stejně jako u změny hustoty může trvat několik sekund, než se změna projeví.

Kapitola 7

Testování systému

Testování systému započalo již při prvních prototypech simulační aplikace. Testováním jsem si vždy ověřoval správnost vnitřního fungování systému. Jak se však aplikace rozrůstala, testování většiny funkcionality při vývoji se stávalo náročnějším.

Bylo tedy potřeba otestovat celou aplikaci uceleně, a to jak po stránce funkční, tak i po stránce výkonové. Tento proces bude popsán na následujících stránkách. Testování probíhalo v simulovaném prostředí za použití jednoduchých konfigurací dopravních řadičů sX vytvořených pro tento účel.

Testování systému dělím do tří kategorií. První z nich, jednotkové neboli Unit testy, se píší a provádí průběžně v rámci implementace. Další kategorii, totiž funkční testování celého systému provádím až po ukončení implementace. Nakonec také otestuji jednotlivé části systému z hlediska výkonu pro nalezení kritických míst v systému.

7.1 Jednotkové testy

Účelem jednotkových (Unit) testů je testování menších částí systému, tzv. jednotek [14]. Jednotkou může být např. modul, třída nebo její metoda. Unit testy pro danou jednotku se vytvářejí při jejím vývoji. Cílem je najít a odstranit co nejvíce logických chyb již při vývoji, a tím usnadnit práci při testování systému.

Při vývoji této diplomové práce bylo jednotkových testů využito při implementaci důležitých tříd především v modulu `Shared`.

7.2 Testování funkčnosti systému

Testování funkčnosti celého systému se zde dělí na dvě části. První částí je testování funkčnosti systému bez ohledu na jeho vnitřní fungování (tzv. black-box testování). Druhou částí je testování některých funkčních prvků, které se black-box testováním testují obtížně. Tyto dva přístupy budou využívat společnou testovací sadu konfigurací.

7.2.1 Testovací sada konfigurací

Pro testování funkčnosti systému jsem vytvořil sadu křižovatek určenou pro testování. Tyto křižovatky se od sebe navzájem liší počtem ramen, úhly mezi rameny, typem signálních programů a také tím, jestli obsahují detektory. Parametry testovacích konfigurací byly vybrány po dohodě s vedoucí a jsou popsány v tabulce 7.1.

Tabulka 7.1: Testovací sada pro funkční testování systému

Číslo	Počet ramen	Úhly ramen	Typ signálních plánů	Detektory
1	2	0, 45	Dynamický	Ano
2	2	0, 90	Pevný	-
3	2	0, 135	Dynamický	Ne
4	2	0, 180	Pevný	-
5	3	0, 45, 90	Dynamický	Ano
6	3	0, 45, 135	Pevný	-
7	3	0, 45, 180	Dynamický	Ne
8	3	0, 90, 180	Pevný	-
9	3	0, 90, 225	Dynamický	Ano
10	4	0, 45, 90, 135	Pevný	-
11	4	0, 45, 90, 180	Dynamický	Ne
12	4	0, 45, 90, 270	Pevný	-
13	4	0, 90, 135, 180	Dynamický	Ano
14	4	0, 90, 180, 270	Pevný	-
15	4	0, 45, 135, 315	Dynamický	Ne

Označení signálních plánů jako pevný nebo dynamický odkazuje na kapitolu o konfiguraci dopravních řadičů (viz 2), kde pevný signální plán znamená signální plán orientovaný na signální skupiny. Dynamický signální plán je potom signální plán orientovaný na etapy a má proměnnou délku trvání jednotlivých fází. U konfigurací s pevnými signálními plány tedy nejsou přítomny detektory, protože by v mém systému nenalezly využití.

7.2.2 Black box testování

Vytvořením testovacích případů a jejich vykonáním chci otestovat funkčnost systému z pohledu uživatele. Otestovány budou tedy ovládací prvky aplikace a jejich výstup. To vše bez ohledu na vnitřní zpracování na straně simulační aplikace a nebo klientské aplikace. Jedná se tedy o takzvané black-box testování.

Příklad testovacího případu pro black box testování je uveden v tabulce 7.2. Všechny testovací případy pro black-box testování jsou dostupné v příloze C. Těchto 31 testovacích případů popisuje základní a pokročilejší operace nad klientskou aplikací. Několikanásobným opakováním testování těchto testovacích případů a jejich variací jsem odhalil několik implementačních chyb. Tyto chyby jsou popsány v tabulce 7.3.

Z 8 nalezených chyb bylo 5 případů méně závažných chyb. V tabulce 7.3 jsou tyto nálezy podbarveny bílou barvou. Další 3 případy by se daly označit jako vážné. Hlášení nelogické chyby při přidávání některých testovacích konfigurací pod určitým úhlem a vozidla projíždějící křižovatkou na zelenou mají společnou příčinu a budu se jim věnovat v sekci o white-box testování (viz 7.2.3). Poslední kritickou chybou je zasekávání vizualizace při vysokém počtu vozidel. Příčinou této chyby se zabývám v sekci o výkonnostním testování (viz 7.3). Příčina této chyby opravena nebyla, bylo však provedeno opatření, které její následky eliminovalo. Aby se zamezilo projevům této chyby, maximální počet vozidel v simulaci byl omezen na 50. Doba animace jednoho simulačního kroku v klientské aplikaci byla navíc prodloužena na 1,5 násobek délky simulačního kroku, aby vznikl větší časový prostor pro generování simulačních dat ze simulační aplikace.

Tabulka 7.2: Testovací případ #11

ID: #11	Spojování křižovatek 1
Popis:	Základní test pro dvě nespojitelné křižovatky
Vstupní podmínky:	1. Simulační aplikace běží a klientská aplikace je na ní připojena. 2. Simulační situace je prázdná.
Postup:	1. V klientské aplikaci klikneme na tlačítko "Add first intersection". 2. Ve formuláři vynecháme pole "Angle", nahrajeme testovací konfiguraci číslo 4 a zaškrtneme políčko "Cars". 3. Počkáme na přidání a klikneme na střed nově přidané křižovatky. 4. Klikneme na tlačítko "Add neighbor intersection". 5. Ve formuláři vynecháme pole "Angle", jako relativní pozici vybereme "Left"a nahrajeme testovací konfiguraci číslo 4.
Očekávaný výsledek:	1. Křižovatka s ID 1 je vlevo od křižovatky s ID 0. 2. Křižovatky 0 a 1 nejsou nijak propojeny.

Tabulka 7.3: Chyby nalezené při funkčním testování

Činnost	**Chyba**	**Zdroj chyby**	**Vyřešeno**
Přidávání křižovatek	Některé křižovatky nejsou zobrazeny celé	Webová aplikace	Ano
Přidávání křižovatek	U některých křižovatek je hlášena nesmyslná chyba při přidávání.	Simulační aplikace	Ano
Odebírání křižovatek	Při smazání poslední křižovatky se neaktualizuje počet tras	Simulační aplikace	Ano
Odebírání křižovatek	Někdy zůstává po smazané křižovatce také spojovací rameno	Simulační aplikace	Ano
Zastavení simulace	Semafory se nevrací do původního stavu	Webová aplikace	Ano
Zastavení simulace	Zastavení simulace někdy zobrazí chybovou hlášku o chybě v simulaci	Simulační aplikace	Ano
Běh simulace	U některých křižovatek projíždějí vozidla na červenou a stojí na zelenou.	Simulační aplikace	Ano
Běh simulace	Při vyšším počtu vozidel než 50 se vizualizace zasekává	Simulační aplikace	Ano

7.2.3 White box testování

White-box testování slouží k otestování kritických součástí systému, kde na rozdíl od black-box testování jsem jako tester obeznámen s vnitřním fungováním programu.

Oblastí, na kterou se chci v této části testování zaměřit, je verifikace správnosti vizualizace přímým porovnáním s vestavěnou vizualizací v programu SUMO. Cílem je objevit případné nežádoucí odchylky. Mezi prvky které lze tímto ověřit patří:

- počet a souřadnice křižovatek,
- počet, směry a pořadí dopravních pruhů,
- počet aut,
- souřadnice aut,
- směr aut,
- směrová světla aut,
- brzdová světla aut,
- světelná signalizace křižovatek,
- signální plán a fáze křižovatek,
- simulační čas.

Tabulka 7.4: Testovací případ #103

ID: #103	Verifikace vizualizace 3
Popis:	Porovnání pozic vozidel ve vizualizaci a v SUMO vizualizaci
Vstupní podmínky:	1. Simulační aplikace běží a klientská aplikace je na ní připojena.
	2. Simulační situace je vytvořena podle testovacího případu #14.
	3. Simulační aplikace využívá grafické uživatelské rozhraní (debug mód).
Postup:	1. Spustíme simulaci podle testovacího případu #101.
	2. Po 30s simulačního času pozastavíme vizualizaci v klientské aplikaci.
	3. Manuálně zkontrolujeme počet vozidel v obou vizualizacích.
	4. Manuálně zkontrolujeme každé vozidlo: jeho ID, pozici, natočení a signalizaci.
Očekávaný výsledek:	1. Počet vozidel je stejný v obou vizualizacích.
	2. U všech vozidel odpovídá jejich ID, pozice, natočení a signalizaci.

Příklad testovacího případu pro verifikaci vizualizace je v tabulce 7.4. Celkem byly vytvořeny 4 obecné testovací případy. Jejich popis se nachází v příloze D. Tyto testovací

Tabulka 7.5: Chyby nalezené při verifikaci vizualizace

Činnost	Chyba	Vyřešeno
Porovnání křižovatek	Některá centra křižovatek v SUMO vizualizaci jsou větší než jejich ekvivalenty v klientské aplikaci. Efektem tak je, že se vozidla v některých specifických případech nacházejí mimo křižovatky při jejím průjezdu.	Ne
Porovnání světelné signalizace	U některých křižovatek se aktuální světelná signalizace mezi vizualizacemi rozchází.	Ano

případy byly testovány s různými variacemi rozsáhlejších dopravních situací (s pěti a více křižovatkami), pro nalezení veškerých možných nepřesností.

Nakonec byly nalezeny 2 odlišnosti mezi vizualizacemi. Tyto chyby jsou popsány v tabulce 7.2.3. První nalezená chyba se vyskytla při porovnávání center křižovatek. Tato odlišnost způsobuje, že se vozidlo ve webové aplikaci při přejíždění takto ochuzené křižovatky může dostat mimo její centrum, a tedy "mimo silnici". Tuto chybu jsem se rozhodl v rámci této práce neopravit. Důvodem je vysoká náročnost této opravy oproti relativně malé přidané hodnotě.

Druhá chyba se dá označit jako kritická. O této chybě byla už řeč při black-box testování (viz 7.2.2). Ve vizualizaci v klientské aplikaci se tato chyba projevuje jízdou vozidel při červené barvě na SSZ u takto nekorektních křižovatek.

7.3 Výkonnostní testování

Výkonnostní testování slouží za účelem nalezení limitů systému [8] z hlediska jeho výkonu a stability. Zajímá mne také vztah mezi zvyšováním náročnosti simulace a délkou provádění jednotlivých akcí jak na serveru, tak i na klientovi. Primárním kritickým faktorem zde bude počet vozidel, a to jak z pohledu vlivu na délku provádění simulačního kroku simulátorem, tak také na vykreslení simulace.

Z pohledu paměťové náročnosti simulační aplikace, ale i klienta v prohlížeči, bude zajímavý test dlouhodobé zátěže.

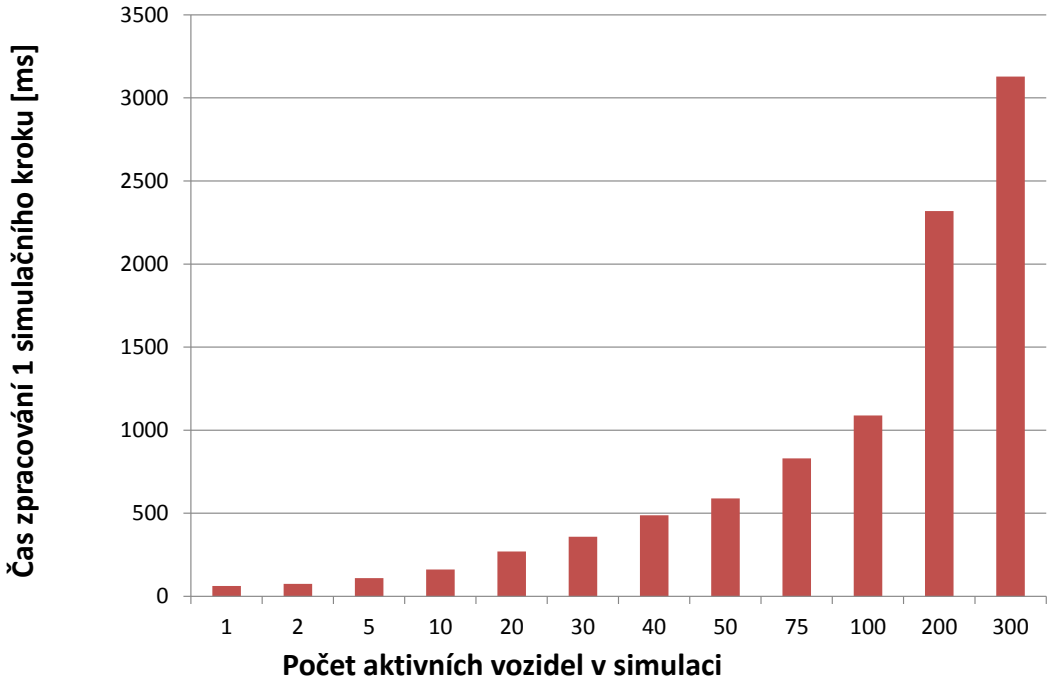
Pro testování výkonu byl použit notebook s dvoujádrovým procesorem Intel i5-4300M (2.60 GHz) s 8 GB RAM. Testování proběhlo v prostředí operačního systému Windows 7 Enterprise SP1 64bit.

7.3.1 Výkonnostní testování simulace

Cílem tohoto testování je nalézt horní hranici výkonu simulace s ohledem na počet aktivních vozidel.

Jak je vidno na obrázku 7.1, počet aktivních vozidel v běžící simulaci má velký vliv na celkový výkon simulace. S ohledem na použitou délku simulačního kroku 0.5s a stejně dlouhou vizualizaci na klientské straně se jako limitní počet aktivních vozidel ukazuje číslo 40. Což znamená, že např. při 50 aktivních vozidlech by vizualizace musela čekat na data a do té doby by se simulace pozastavila. Pokud bychom uvažovali délku simulačního kroku 1s a stejně dlouhou animaci vozidel, limitem by patrně bylo číslo kolem 90.

Testování probíhalo následovně:



Obrázek 7.1: Graf závislosti mezi počtem aktivních vozidel a délkou zpracování jednoho simulačního kroku [vlastní]

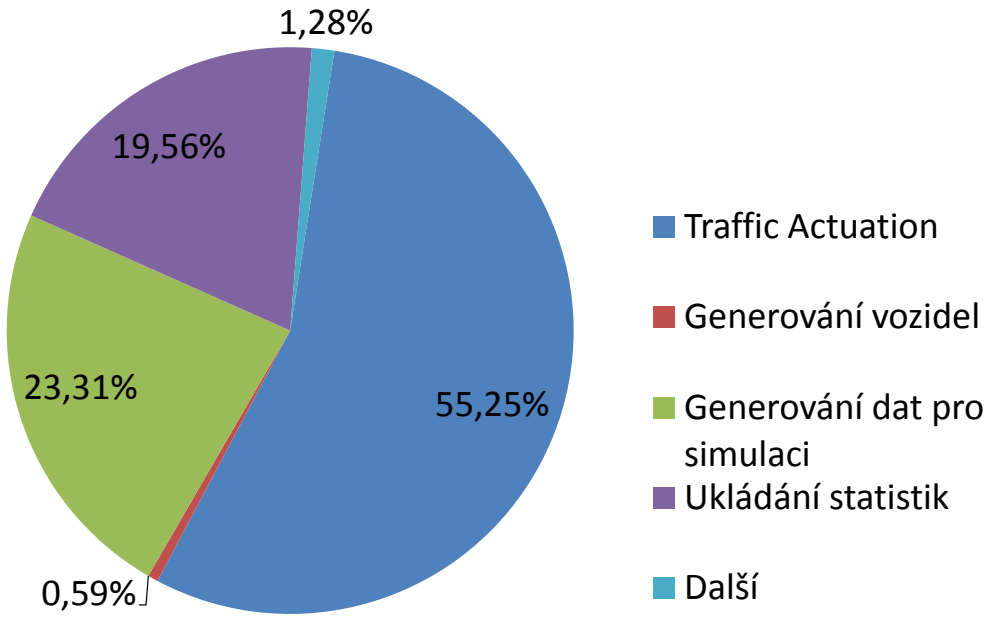
1. Simulační aplikace byla spuštěna s testovaným parametrem maximálního počtu vozidel.
2. V klientské aplikaci byla vytvořena příslušná dopravní situace (viz níže).
3. Simulace byla spuštěna.
4. Po dosažení limitu aktivních vozidel bylo vybráno 10 simulačních kroků. Rozestup mezi vybranými kroky byl minimálně 10 simulačních kroků.

Testování probíhalo na dostatečně velké dopravní situaci, kdy byla použita testovací konfigurace č. 14 (viz tabulka 7.1). Tato konfigurace byla použita celkem 16x, přičemž byla vytvořena matice 4x4. Počet možných cest v této situaci byl 240. Byl měřen rozdíl času mezi započetím vykonávání metody a jejím ukončením. Testování bylo pro ověření opakováno.

7.3.2 Testování jednotlivých úkonů simulačního kroku

V tomto testu chci zjistit délku zpracování jednotlivých úkonů aplikace v rámci jednoho simulačního kroku. Cílem je najít případná vhodná místa pro optimalizaci. Jedná se o tyto úkony"

- generování vozidel,
- dynamické řízení dopravy na křižovatkách,
- sběr dat pro vizualizaci,
- sběr statistik pro celkový výstup simulace.



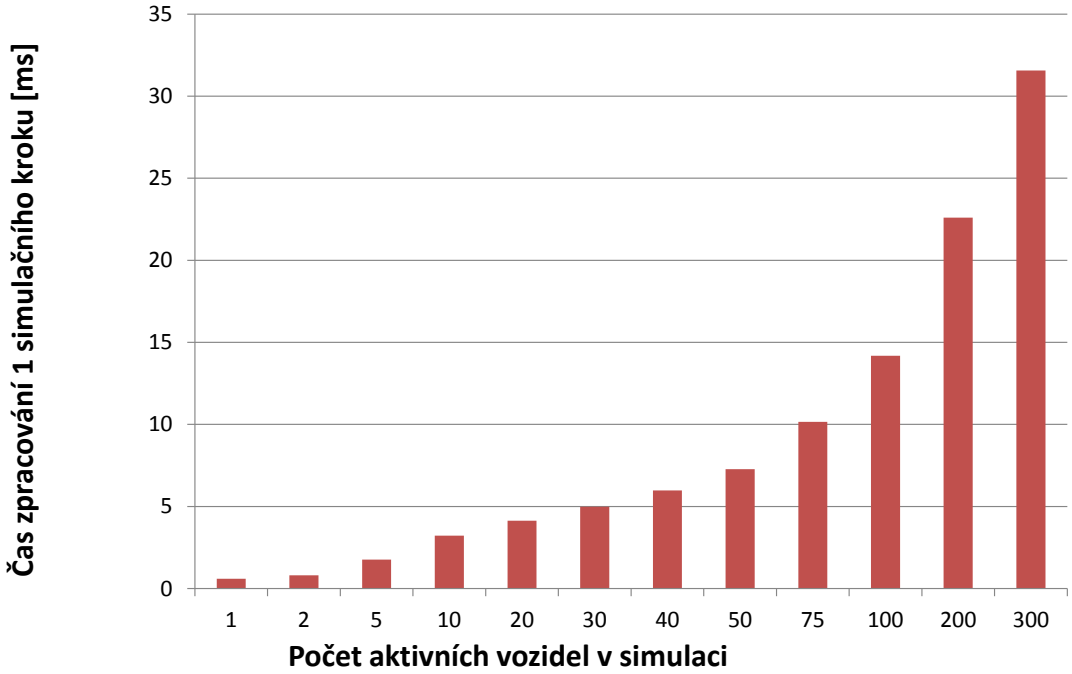
Obrázek 7.2: Poměr jednotlivých operací na času provádění simulačního kroku [vlastní]

Z grafu na obrázku 7.2 je patrné, že generování vozidel a jejich vkládání do simulace je z testovaných úkonů tím nejméně časově náročným. Důvodem bude četnost provádění této operace, která je řádově menší než četnost ostatních tří. Generování simulačních dat a ukládání statistik se provádí pro každý simulační krok tolikrát, kolik je v simulaci přítomných vozidel. Tyto úkony v sobě zahrnují mnoho volání TraCI rozhraní (viz 3.5.3). Zdaleka nejvíce času spotřebuje dynamické řízení provozu. Algoritmus dynamického řízení je náročný sám o sobě, a také získává data o simulaci pomocí TraCI rozhraní.

Testování probíhalo v dopravní situaci složené ze čtyř křižovatek. Všechny byly definovány konfigurací č. 13. Tyto byly propojeny do matice 2x2. Maximální počet vozidel byl stanoven na číslo 50. Simulace byla aktivní po dobu 2500 simulačních kroků a experiment byl 5x opakován. Časy jednotlivých úkonů byly postupně sčítány. Procentuální rozložení na obrázku 7.2 odpovídá průměru ze všech opakování testu.

7.3.3 Výkonnostní testování vizualizace

Zde chci zjistit, jaký vliv má vysoký počet vozidel na dobu vykreslení vizualizace. Výsledek bude porovnán také s generováním dat pro simulaci.



Obrázek 7.3: Graf závislosti mezi počtem aktivních vozidel a délkou zpracování jednoho simulačního kroku [vlastní]

Pokud porovnáme grafy na obrázcích 7.1 a 7.3, uvidíme, že nárůst délky zpracování simulačního kroku v závislosti na počtu aktivních vozidel roste podobně rychle v obou aplikacích. Časy zpracování na straně klientské aplikace jsou však o několik řádu kratší (5,9 ms u klientské aplikace proti 500 ms u simulační aplikace při 40 aktivních vozidel).

Testování probíhalo stejným způsobem jako výkonnostní testování simulace (viz 7.3.1). Byla také použita stejná dopraní situace.

7.4 Výsledky testování

Funkční testování odhalilo celkem 10 chyb, z toho 9 bylo opraveno. Z 10 chyb byly 4 chyby kritické, z nichž byly vyřešeny všechny. Pro funkční testování bylo vytvořeno celkem 35 testovacích případů.

Testování výkonu systému odhalilo kritická místa celého systému. Tato místa jsou na straně simulační aplikace. Hloubější analýza ukázala, že úzkým hrdlem jsou prvky Testování výkonu neodhalilo nedostatky s výkonem na straně klientské aplikace.

Pokud bych mohl nyní zpětně něco změnit na testování, asi bych obětoval čas vytvoření automatizované sady testů pro webového klienta, např. pomocí technologie Selenium[1]. Automatická sada by mi pomohla najít více chyb již při vývoji. Rovněž bych se pokusil o větší využití vývoje řízeného testy[2]. Využití této filozofie by mi ušetřilo čas při psaní jednotkových testů tříd, kdy jsem tyto testy psal pro již hotovou implementaci. Obrácením tohoto procesu (tedy psaní testů před vlastním vývojem implementace) bych pravděpodobně dosáhl čistšího návrhu jednotlivých modulů, a také bych nemusel zpětně na základě napsaných testů měnit implementaci.

[1] Selenium: https://www.seleniumhq.org/
[2] Test-driven developmnent, TDD

Kapitola 8

Závěr

Diplomová práce navazuje na semestrální projekt, kde bylo potřeba splnit následující: nastudovat si nejprve obecnou stránku konfigurace dopravního řadiče, poté už konkrétní konfiguraci řadiče sX, a navrhnout a implementovat systém, který bude konfigurace zpracovávat do jedné simulace a tu spouštět. Semestrální projekt byl v diplomové práci obohacen o klientskou aplikaci, která simulaci ovládá a zobrazuje, a navíc slouží jako jednoduchý editor dopravních situací.

Úvod práce pro mě tedy znamenal seznámení se s procesy moderního dynamického ovládání křižovatky, jakožto i s tím, jak se například konfigurace řadiče vytváří, a získat další poznatky. Některé z těchto znalostí jsem v práci nakonec nijak viditelně nevyužil, ale pomohly mi utvořit si o této komplexní oblasti potřebný kontext a obecný rozhled.

Po nastudování a prvních prototypech jsme s mým konzultantem definovali požadavky na výsledný systém, které se staly základem mého návrhu. V této fázi se objevilo několik problémů, které se mi nakonec podařilo vyřešit - například určení počtu pruhů v křižovatce, nebo vyhledávání cest v simulační síti. Na návrh navazuje samotná implementace simulačního systému, které využívá simulační platformu SUMO. V implementaci se mi podařilo splnit definované požadavky na systém.

Následoval návrh a implementace webové nástavby nad tímto systémem. Tato část přinesla několik dalších zajímavých problémů k vyřešení, především otázku efektivní komunikace mezi klientem a serverem, nebo celkové zajištění plynulé vizualizace. Pro budoucí práci na simulačním systému jsem se snažil při implementaci a návrhu zohledňovat možná rozšíření, která nejsou přímo definována v tomto zadání - například využití reálných dat z dopravních řadičů pro simulaci. Kladl jsem důraz na snadnou rozšiřitelnost jak serverové tak i klientské části.

Testování v simulovaném prostředí odhalilo několik chyb, z nichž většina byla ve finální odevzdávané verzi opravena. Výsledky výkonnostního testování potom odhalily limity simulační aplikace v její rychlosti generování simulačních dat.

Vzhledem k náročnosti zadání je tedy hlavním výstupem diplomové práce prototyp dopravního simulačního systému. Na něm bude založen další vývoj, aby v budoucnu mohl pomáhat například dopravním inženýrům při návrhu dopravních řešení se zahrnutím okolní dopravní situace. Dalším možným využitím může být testování a vylepšování stávajících konfigurací dopravních řadičů světelných křižovatek.

8.1 Budoucí rozšíření simulačního systému

Budoucí rozšíření lze rozdělit na dvě kategorie. První kategorií jsou vylepšení týkající se schopnosti systému lépe obsáhnout reálné situace. Mezi taková vylepšení patří podpora chodců. Důležitým prvkem, který stávající prototyp přiblíží reálnému využití, je vylepšení systému generování vozidel, který by měl být konfigurovatelný s ohledem na hlavní a vedlejší tahy. Možná ještě důležitější bude práce na optimalizaci současného kódu pro generování simulačních dat za účelem výrazného zvednutí počtu možných aktivních vozidel v situaci.

Do druhé kategorie spadají nové prvky a vylepšení mající za cíl zvýšit použitelnost celého systému pro jeho cílové uživatele. Jedním z nejdůležitějších vylepšení bude vytvoření možnosti připojit více klientských aplikací k jedné simulační aplikaci. Zde bude potřeba malá úprava na straně klienta a větší úprava na straně serveru, který bude muset zpracovávat několik běžících simulací najednou. Každá simulace by tedy operovala ve vlastním vláknu. Dalším prvkem spadajícím do druhé kategorie může být jednoduchá databáze pro ukládání často používaných dopravních situací.

Vylepšením spadajícím do obou kategorií potom bude možnost použít reálná data z dopravních řadičů pro vizualizaci, což umožní přehrávání situací skutečně existující dopravy v daném místě.

Literatura

[1] How Red-light Cameras Work. [Online; navštíveno 6.1.2018].
URL https://auto.howstuffworks.com/car-driving-safety/safety-regulatory-devices/red-light-camera1.htm

[2] ITS Knowledge Resources Portal. [Online; navštíveno 6.1.2018].
URL https://www.itsknowledgeresources.its.dot.gov/

[3] SUMO User Documentation. [Online; navštíveno 7.1.2018].
URL http://sumo.dlr.de/wiki/SUMO_User_Documentation

[4] The Sitraffic sX traffic controller. [Online; navštíveno 16.1.2018].
URL https://www.siemens.com/press/en/feature/2015/mobility/2015-01-sitraffic-sx.php

[5] TraaS - TraCI as a Service. [Online; navštíveno 6.1.2018].
URL http://traas.sourceforge.net/cms/

[6] *TP 81 - Navrhování světelných signalizačních zařízení pro řízení provozu na pozemních komunikacích*. Praha, třetí vydání, 2015.

[7] Barceló, J.: *Fundamentals of Traffic Simulation*. International Series in Operations Research & Management Science, Springer New York, 2011, ISBN 9781441961426.

[8] Farrell-Vinay, P.: *Manage Software Testing*. Boston, MA, USA: Auerbach Publications, 2007, ISBN 0849393833, 9780849393839.

[9] Fielding, R.; Gettys, J.; Mogul, J.; aj.: RFC 2616, Hypertext Transfer Protocol – HTTP/1.1. 1999.
URL http://www.rfc.net/rfc2616.html

[10] Kanagaraj, V.; Asaithambi, G.; Kumar, C. N.; aj.: Evaluation of Different Vehicle Following Models Under Mixed Traffic Conditions. *Procedia - Social and Behavioral Sciences*, ročník 104, 2013: s. 390 – 401, ISSN 1877-0428, doi:https://doi.org/10.1016/j.sbspro.2013.11.132, 2nd Conference of Transportation Research Group of India (2nd CTRG).
URL http://www.sciencedirect.com/science/article/pii/S1877042813045230

[11] Krajzewicz, D.; Erdmann, J.; Behrisch, M.; aj.: Recent Development and Applications of SUMO - Simulation of Urban MObility. *International Journal On Advances in Systems and Measurements*, ročník 5, č. 3&4, December 2012: s. 128–138.

[12] Kumar, V.; Lin, L.; Krajzewicz, D.; aj.: iTETRIS: Adaptation of ITS Technologies for Large Scale Integrated Simulation. In *2010 IEEE 71st Vehicular Technology Conference*, May 2010, ISSN 1550-2252, s. 1–5, doi:10.1109/VETECS.2010.5494182.

[13] Maciejewski, M.: A Comparison of Microscopic Traffic Flow Simulation Systems for an Urban Area. *Transport Problems : an International Scientific Journal*, ročník 5, č. 4, 2010: s. 27 – 38.

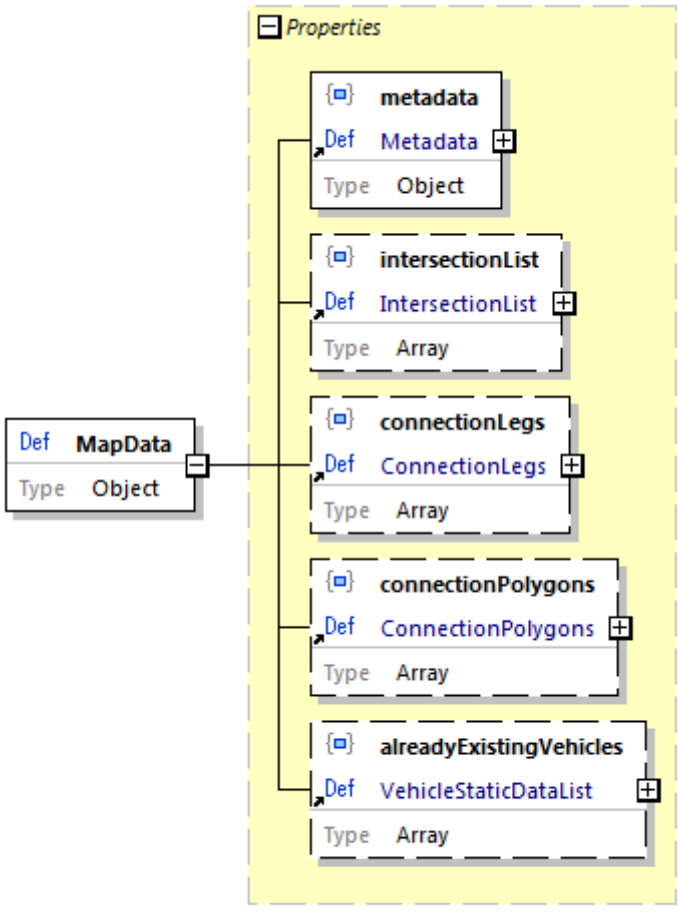
[14] Myers, G. J.; Sandler, C.: *The Art of Software Testing*. USA: John Wiley & Sons, Inc., 2012, ISBN 9781118031964.

Příloha A

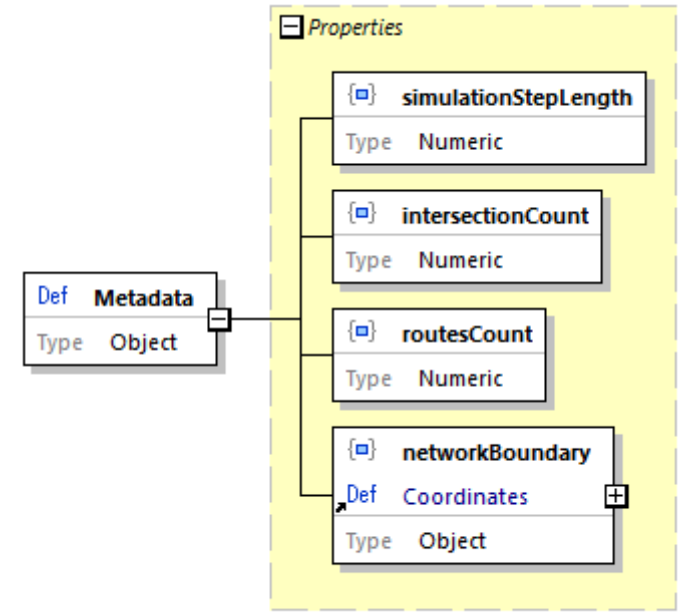
JSON schémata

V této příloze je ve formě obrázků uvedeno JSON schéma, a to pro dva případy, kdy očekáváme od simulátoru nějaká data.

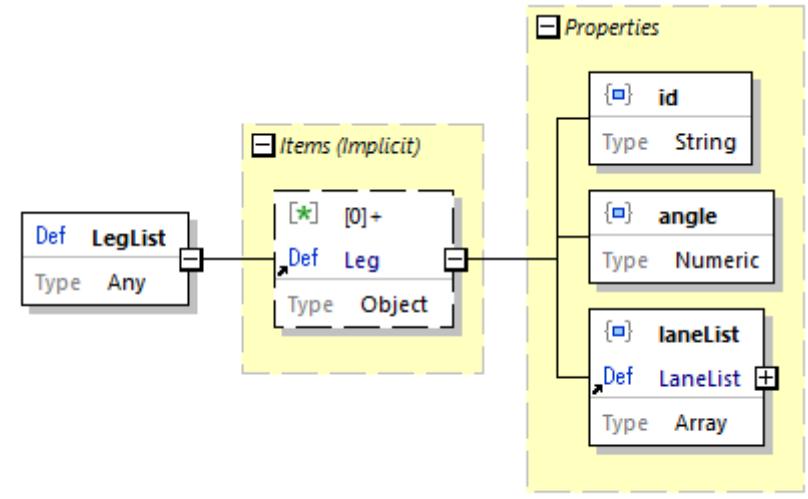
A.1 JSON schéma pro informace o mapě



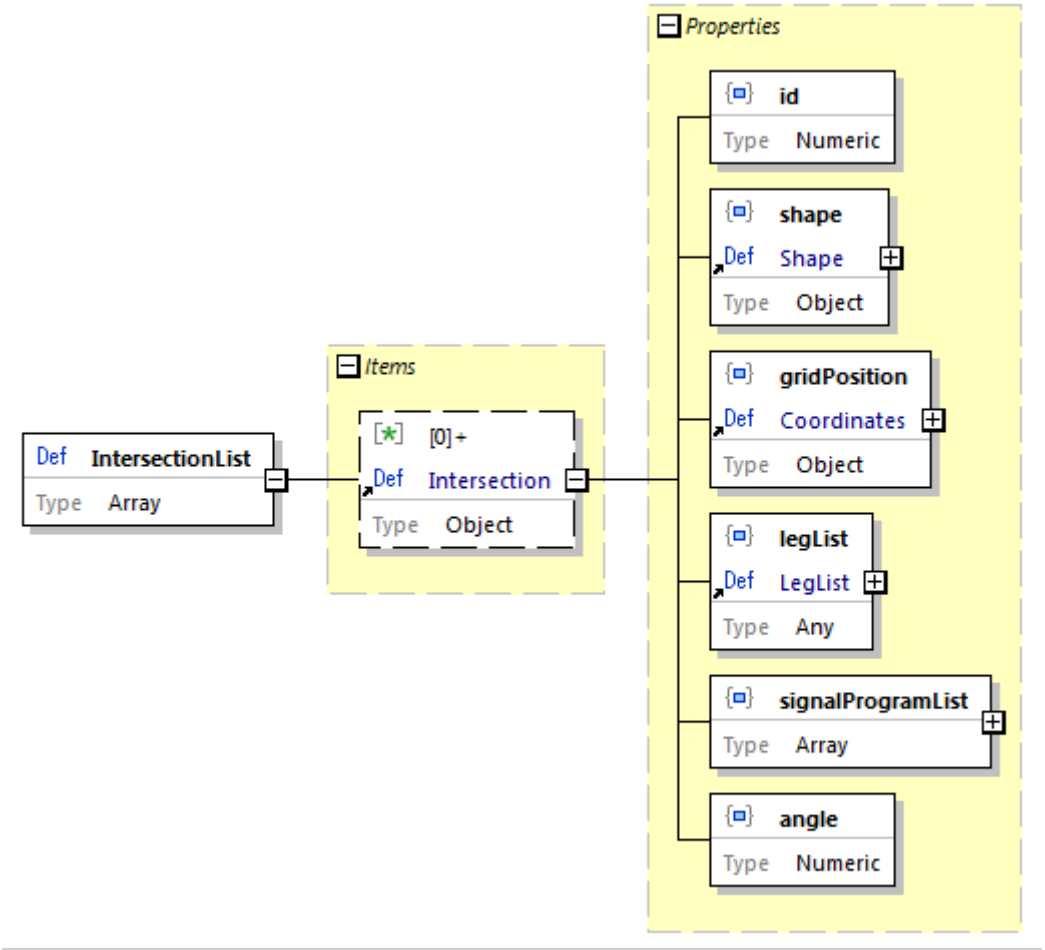
Obrázek A.1: Data pro vykreslení statické části vizualizace [vlastní]



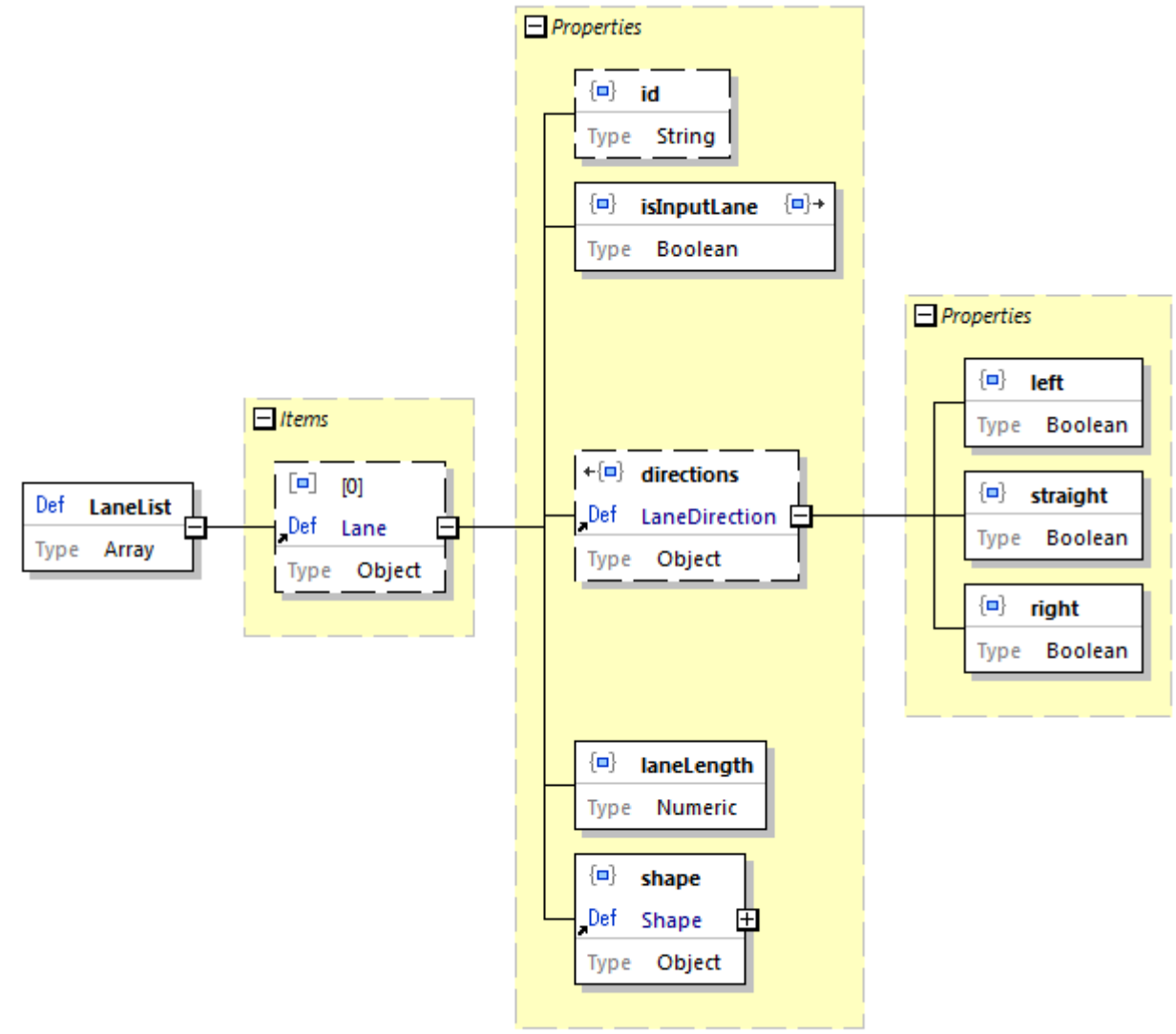
Obrázek A.2: Metadata [vlastní]



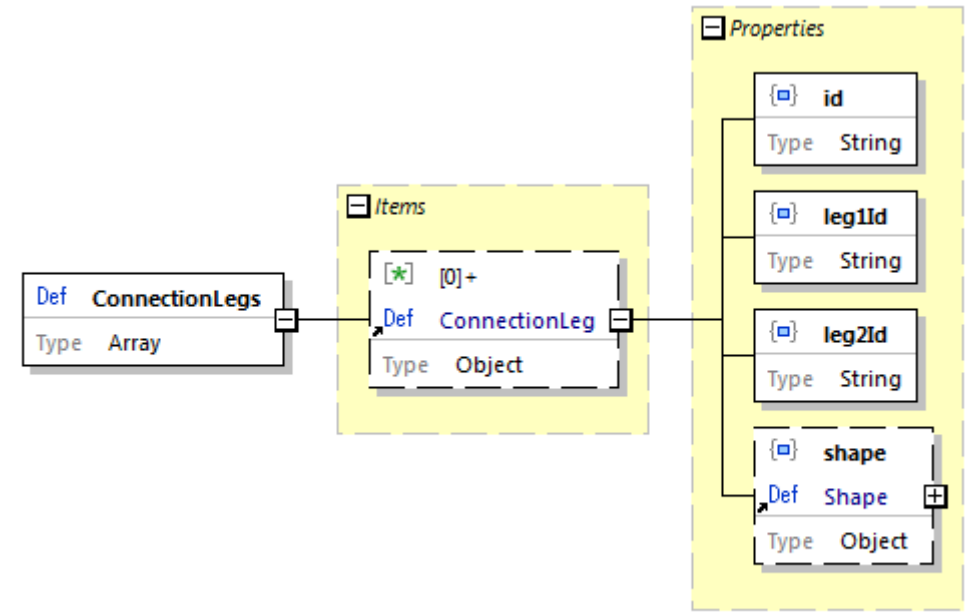
Obrázek A.3: Seznam ramen (pro křižovatku) [vlastní]



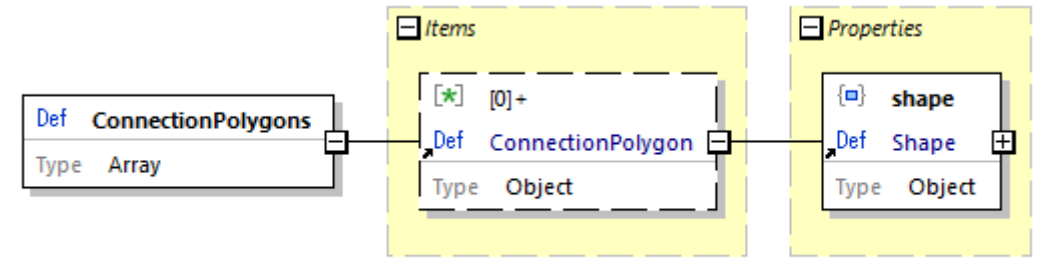
Obrázek A.4: Seznam křižovatek [vlastní]



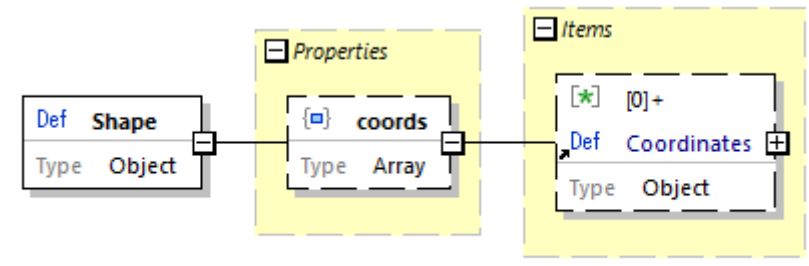
Obrázek A.5: Seznam dopravních pruhů (pro rameno křižovatky) [vlastní]



Obrázek A.6: Seznam propojovacích ramen mezi křižovatkami [vlastní]

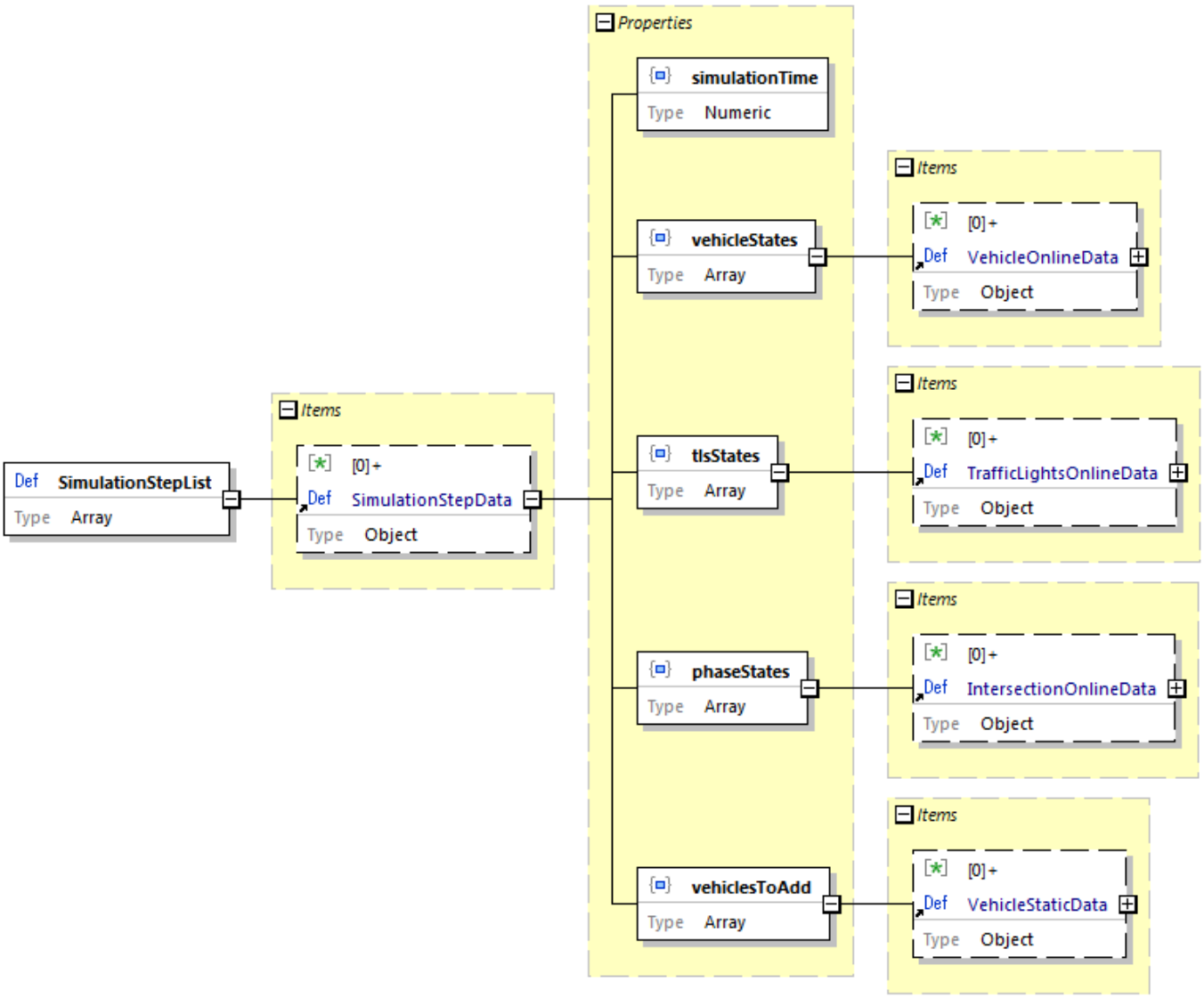


Obrázek A.7: Seznam doplňujících polygonů pro doplnění volného prostoru mezi spojeními [vlastní]

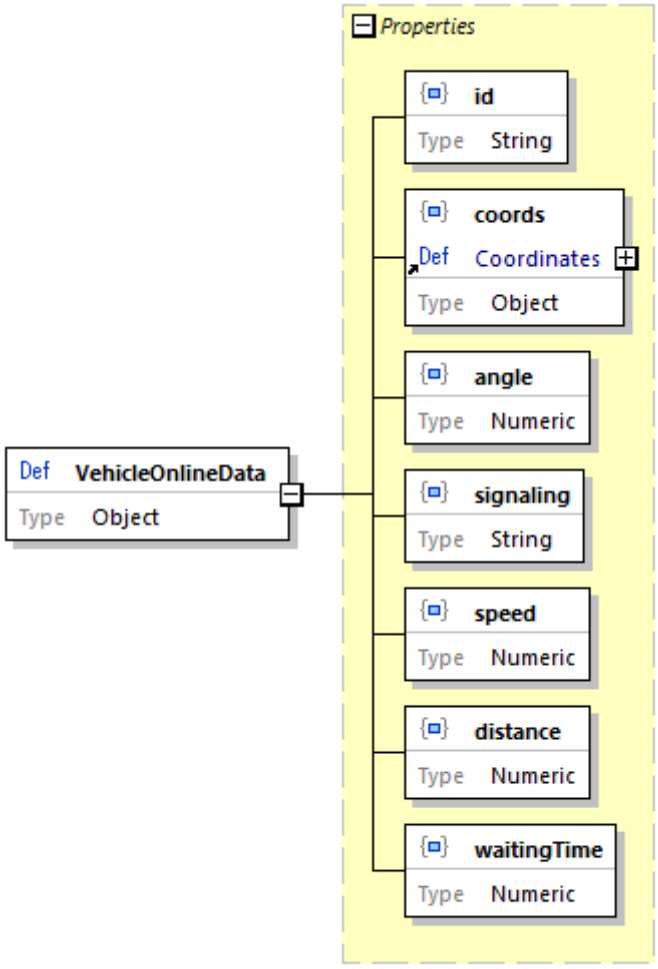


Obrázek A.8: Schéma pro tvar (křižovatky, pruhu) [vlastní]

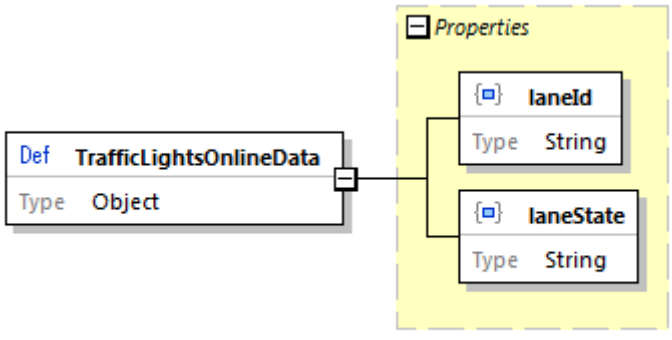
A.2 JSON schéma pro simulační on-line data



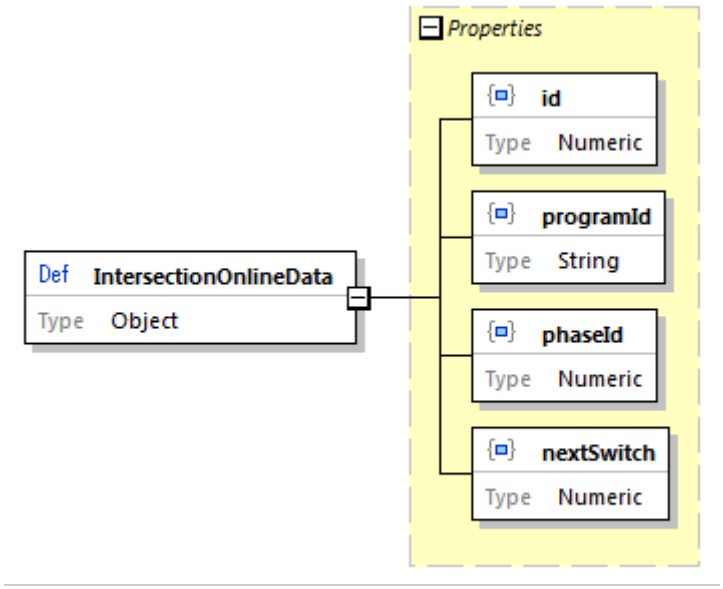
Obrázek A.9: Seznam simulačních kroků v jednom požadavku [vlastní]



Obrázek A.10: On-line data pro vozidla v rámci simulačního kroku [vlastní]

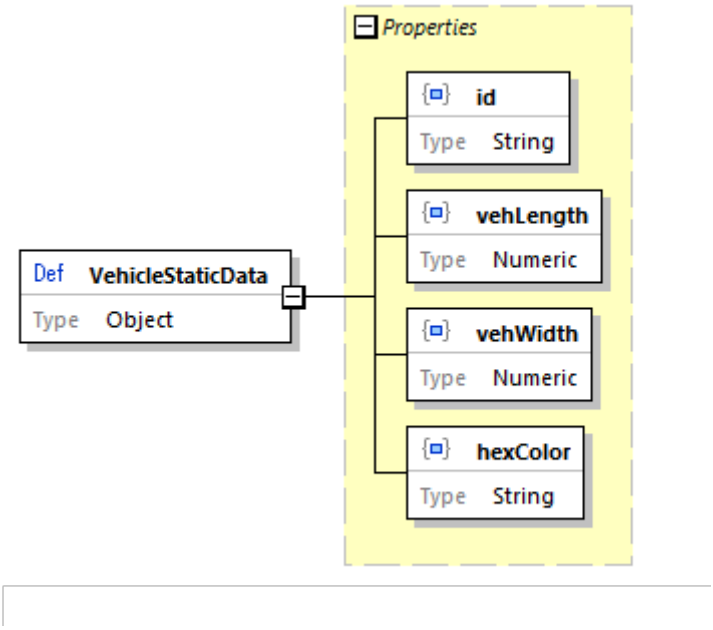


Obrázek A.11: On-line data pro semafory v rámci simulačního kroku [vlastní]

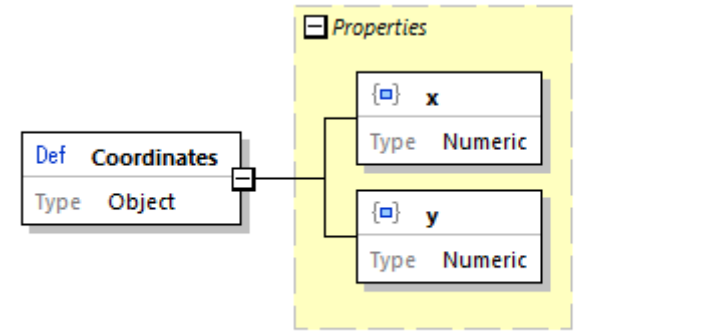


Obrázek A.12: On-line data pro kžižovatky v rámci simulačního kroku [vlastní]

A.3 Schémata společných prvků



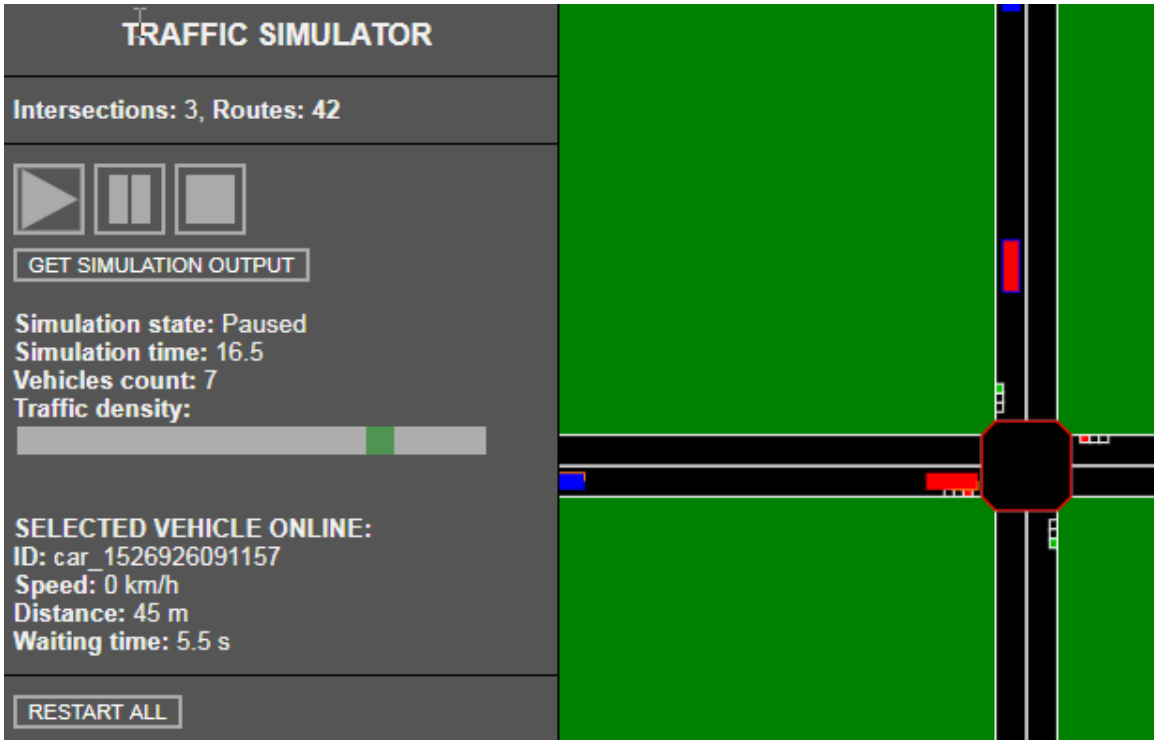
Obrázek A.13: Seznam vozidel k přidání do vizualizace [vlastní]



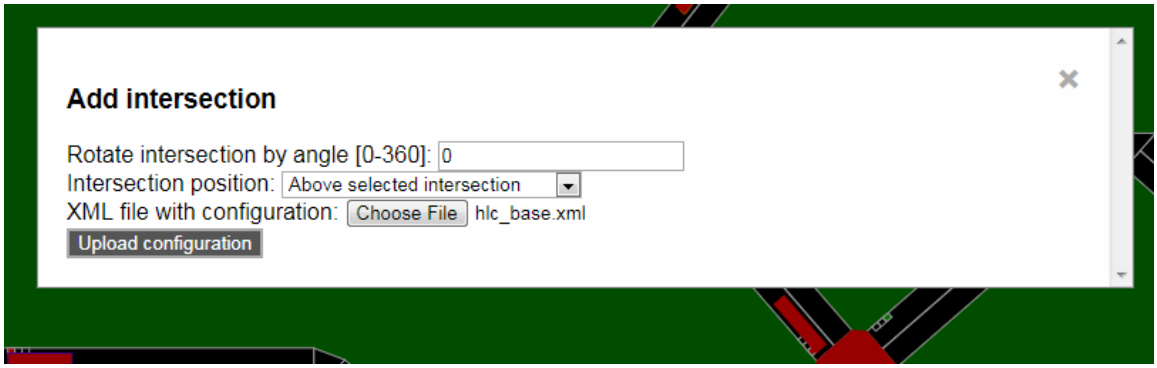
Obrázek A.14: Schéma pro souřadnice [vlastní]

Příloha B

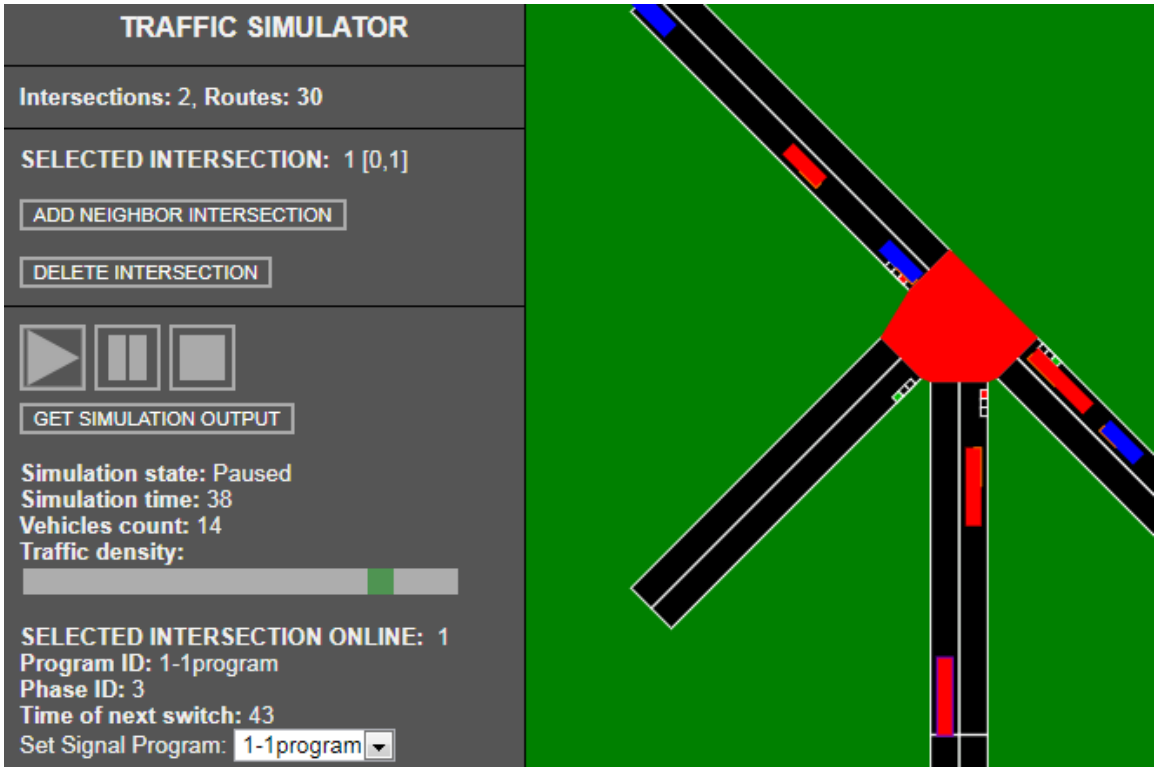
Ukázky z webové aplikace



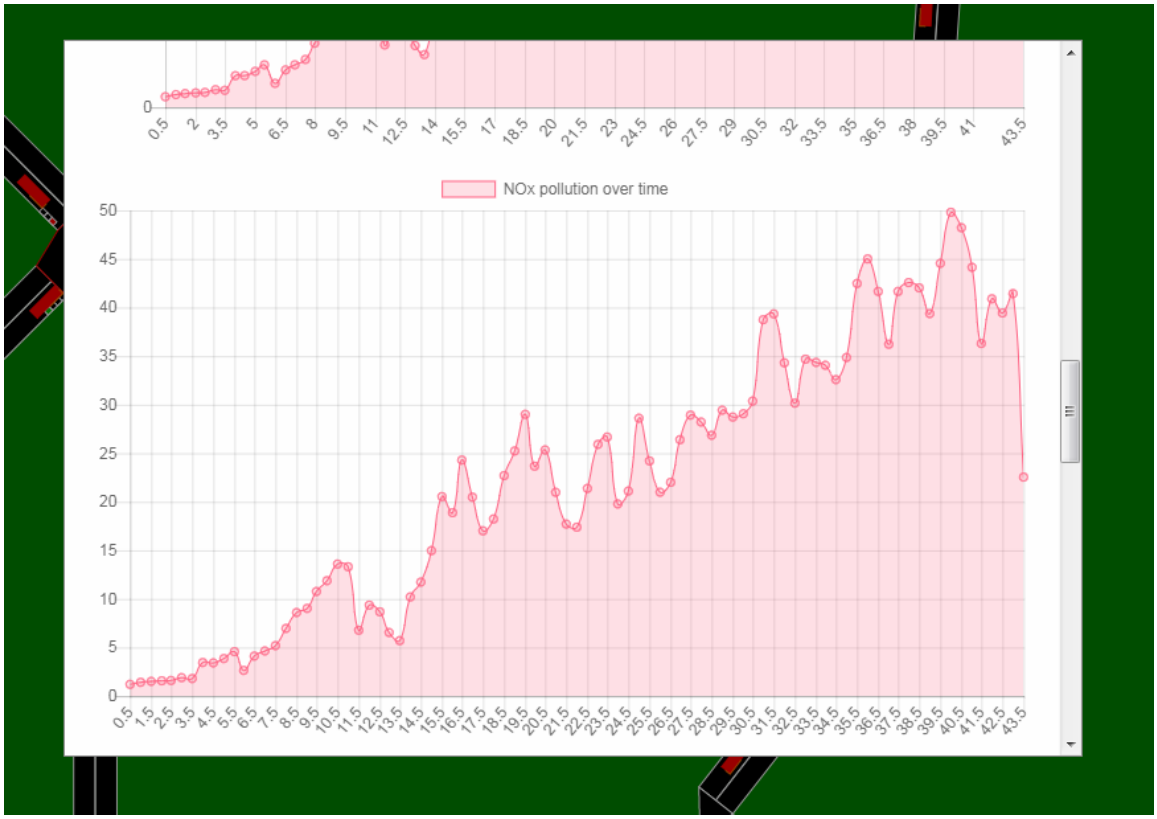
Obrázek B.1: Ukázka z vizualizace: vybrané vozidlo [vlastní, snímek obrazovky]



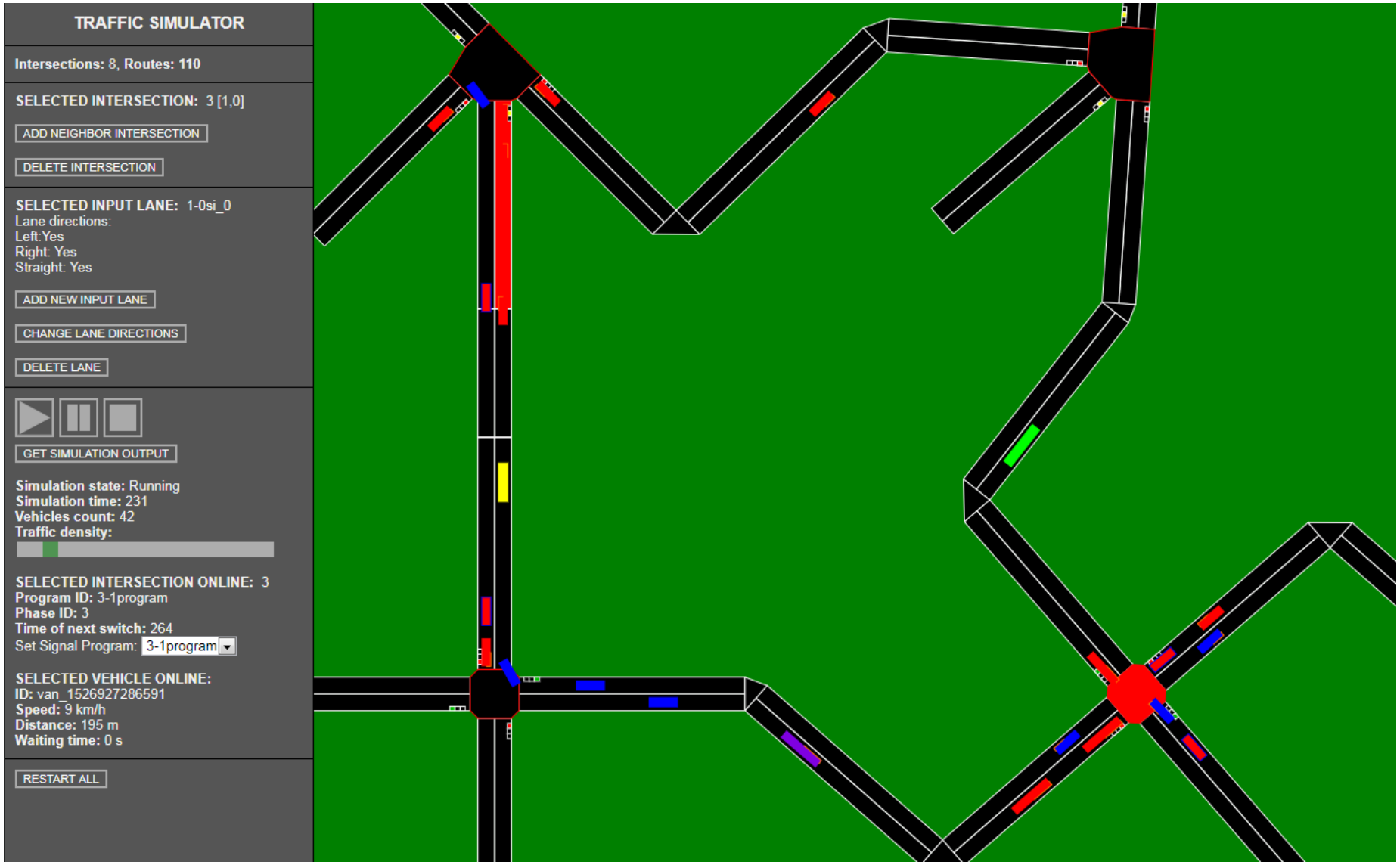
Obrázek B.2: Ukázka z vizualizace: přidávání křižovatky [vlastní, snímek obrazovky]



Obrázek B.3: Ukázka z vizualizace: vybraná křižovatka [vlastní, snímek obrazovky]



Obrázek B.4: Ukázka z vizualizace: výstup simulace [vlastní, snímek obrazovky]



Obrázek B.5: Ukázka z vizualizace: celkový pohled [vlastní, snímek obrazovky]

Příloha C

Testovací případy pro black-box testování

Tabulka C.1: Testovací případ #1

ID: #1	Přidání první křižovatky
Název:	Do prázdné simulační situace lze přidat první křižovatku.
Vstupní podmínky:	1. Simulační aplikace běží a klientská aplikace je na ní připojena.
	2. Simulační situace je prázdná.
Postup:	1. V klientské aplikaci klikneme na tlačítko "Add first intersection".
	2. Ve formuláři vynecháme pole "Angle", nahrajeme testovací konfiguraci č. 4 a zaškrtneme políčko "Cars".
Očekávaný výsledek:	1. V simulační situaci se nachází křižovatka s ID 0.
	2. Tlačítko "Add first intersection"není po přidání již viditelné.

Tabulka C.2: Testovací případ #2

ID: #2	Výběr křižovatky
Název:	Křižovatka může být vybrána
Vstupní podmínky:	1. Simulační aplikace běží a klientská aplikace je na ní připojena.
	2. Simulační situace je vytvořena podle testovacího případu #1.
Postup:	1. Klikneme na střed jediné křižovatky.
Očekávaný výsledek:	1. V postranním panelu lze zjistit ID křižovatky a její pozici.
	2. Jsou viditelná tlačítka "Add neighbor intersection"a "Delete intersection".

Tabulka C.3: Testovací případ #3

ID: #3	Přidání další křižovatky
Název:	Do neprázdné simulační situace lze přidat další křižovatku.
Vstupní podmínky:	1. Simulační aplikace běží a klientská aplikace je na ní připojena.
	2. Simulační situace je vytvořena podle testovacího případu #1.
Postup:	1. Vybereme křižovatku s ID 0 podle testovacího případu #2.
	2. Klikneme na tlačítko "Add neighbor intersection".
	3. Ve formuláři vynecháme pole "Angle", jako relativní pozici vybereme "Left"a nahrajeme libovolnou testovací konfiguraci.
Očekávaný výsledek:	1. Křižovatka s ID 1 je vlevo od křižovatky s ID 0.

Tabulka C.4: Testovací případ #4

ID: #4	Odstranění křižovatky
Popis:	Křižovatku lze odstranit.
Vstupní podmínky:	1. Simulační aplikace běží a klientská aplikace je na ní připojena.
	2. Simulační situace je vytvořena podle testovacího případu #1.
Postup:	1. Vybereme křižovatku s ID 0 podle testovacího případu #2.
	2. V postranním panelu klikneme na dlačítko "Delete intersection",
Očekávaný výsledek:	1. V simulační situaci není viditelná žádná křižovatka.
	2. Tlačítko "Add first intersection"je opět viditelné.

Tabulka C.5: Testovací případ #5

ID: #5	Výběr dopravního pruhu.
Popis:	Je možné vybrat vstupní dopravní pruh.
Vstupní podmínky:	1. Simulační aplikace běží a klientská aplikace je na ní připojena.
	2. Simulační situace je vytvořena podle testovacího případu #1.
Postup:	1. Klikneme na libovolný vstupní dopravní pruh.
Očekávaný výsledek:	1. V postranním panelu lze zjistit ID dopravního pruhu a jeho dostupné směry.
	2. Jsou viditelná tlačítka "Add new input lane", "Change lane"a "Delete lane".

Tabulka C.6: Testovací případ #6

ID: #6	Změna dopravního pruhu
Popis:	Pro vstupní dopravní pruh je možné změnit jeho směry.
Vstupní podmínky:	1. Simulační aplikace běží a klientská aplikace je na ní připojena.
	2. Simulační situace je vytvořena podle testovacího případu #1.
Postup:	1. Vybereme vstupní dopravní pruh s ID 0-0si_0 podle testovacího případu #5.
	2. Klikneme na tlačítko "Change lane".
	3. V modálním okně vybereme směry "Left"a "Straight".
Očekávaný výsledek:	1. V postranním panelu lze vidět nově nastavené směry.

Tabulka C.7: Testovací případ #7

ID: #7	Přidání dopravního pruhu.
Popis:	Do ramene křižovatky je možné přidat další dopravní pruh.
Vstupní podmínky:	1. Simulační aplikace běží a klientská aplikace je na ní připojena.
	2. Simulační situace je vytvořena podle testovacího případu #6.
Postup:	1. Vybereme vstupní dopravní pruh s ID 0-0si_0 podle testovacího případu #5.
	2. Klikneme na tlačítko "Add lane".
	3. V modálním okně vybereme směry "Right".
Očekávaný výsledek:	1. Spodní rameno křižovatky s ID 0 má nyní 2 vstupní dopravní pruhy: 0-0si_0 a 0-0si_1.
	2. Dopravní pruh s ID 0-0si_1 má směry: Left, Straight.
	2. Dopravní pruh s ID 0-0si_0 má směry: Right.

Tabulka C.8: Testovací případ #8

ID: #8	Odebrání dopravního pruhu.
Popis:	Z ramene křižovatky je možné odebrat dopravní pruh.
Vstupní podmínky:	1. Simulační aplikace běží a klientská aplikace je na ní připojena.
	2. Simulační situace je vytvořena podle testovacího případu #7.
Postup:	1. Vybereme vstupní dopravní pruh s ID 0-0si_0 podle testovacího případu #5.
	2. Klikneme na tlačítko "Delete lane".
Očekávaný výsledek:	1. Spodní rameno křižovatky s ID 0 má nyní 1 vstupní dopravní pruh: 0-0si_0.
	2. Dopravní pruh s ID 0-0si_0 má směry: Left, Straight.

Tabulka C.9: Testovací případ #9

ID: #9	Restartování situace.
Popis:	Všechny prvky simulační simulace je možné odebrat najednou.
Vstupní podmínky:	1. Simulační aplikace běží a klientská aplikace je na ní připojena.
	2. Simulační situace je vytvořena podle testovacího případu #7.
Postup:	1. Klikneme na tlačítko "Restart all".
Očekávaný výsledek:	1. Simulační situace je prázdná.
	2. Tlačítko "Add first intersection"je opět viditelné.

Tabulka C.10: Testovací případ #10

ID: #10	Postranní panel: křižovatky a trasy.
Popis:	Počet křižovatek a počet možných tras je možné sledovat v postranním panelu.
Vstupní podmínky:	1. Simulační aplikace běží a klientská aplikace je na ní připojena.
	2. Simulační situace je prázdná.
Postup:	1. Vložíme křižovatku s konfigurací č. 14 dle případu #1.
	2. Smažeme křižovatku s ID 0.
Očekávaný výsledek:	1. Před provedením bodu 1 je počet křižovatek 0 a počet tras 0.
	2. Po provedení bodu 1 je počet křižovatek 1 a počet tras 12.
	3. Po provedení bodu 2 je počet křižovatek 0 a počet tras 0.

Tabulka C.11: Testovací případ #11

ID: #11	Spojování křižovatek 1
Název:	Dvě křižovatky se při nevhodném natočení ramen nespojí.
Vstupní podmínky:	1. Simulační aplikace běží a klientská aplikace je na ní připojena.
	2. Simulační situace je prázdná.
Postup:	1. V klientské aplikaci klikneme na tlačítko "Add first intersection".
	2. Ve formuláři vynecháme pole "Angle", nahrajeme testovací konfiguraci číslo 4 a zaškrtneme políčko "Cars".
	3. Počkáme na přidání a klikneme na střed nové křižovatky.
	4. Klikneme na tlačítko "Add neighbor intersection".
	5. Ve formuláři vynecháme pole "Angle", jako relativní pozici vybereme "Left"a nahrajeme testovací konfiguraci číslo 4.
Očekávaný výsledek:	1. Křižovatka s ID 1 je vlevo od křižovatky s ID 0.
	2. Křižovatky 0 a 1 nejsou nijak propojeny.

Tabulka C.12: Testovací případ #12

ID: #12	Spojování křižovatek 2
Popis:	Dvě křižovatky se při vhodném natočení ramen spojí.
Vstupní podmínky:	1. Simulační aplikace běží a klientská aplikace je na ní připojena.
	2. Simulační situace je prázdná.
Postup:	1. V klientské aplikaci klikneme na tlačítko "Add first intersection".
	2. Ve formuláři vynecháme pole "Angle", nahrajeme testovací konfiguraci číslo 14 a zaškrtneme políčko "Cars".
	3. Počkáme na přidání a klikneme na střed nové křižovatky.
	4. Klikneme na tlačítko "Add neighbor intersection".
	5. Ve formuláři vynecháme pole "Angle", jako relativní pozici vybereme "Left"a nahrajeme testovací konfiguraci číslo 14.
Očekávaný výsledek:	1. Křižovatka s ID 1 je vlevo od křižovatky s ID 0.
	2. Křižovatky 0 a 1 jsou propojeny.

Tabulka C.13: Testovací případ #13

ID: #13	Spojování křižovatek 3
Popis:	Dvě křižovatky mohou mít pouze jedno propojení.
Vstupní podmínky:	1. Simulační aplikace běží a klientská aplikace je na ní připojena.
	2. Simulační situace je prázdná.
Postup:	1. V klientské aplikaci klikneme na tlačítko "Add first intersection".
	2. Do pole "Angle"napíšeme hodnotu "45", nahrajeme testovací konfiguraci číslo 14 a zaškrtneme políčko "Cars".
	3. Počkáme na přidání a klikneme na střed nové křižovatky.
	4. Klikneme na tlačítko "Add neighbor intersection".
	5. Do pole "Angle"napíšeme hodnotu "45", jako relativní pozici vybereme "Left"a nahrajeme testovací konfiguraci číslo 14.
Očekávaný výsledek:	1. Křižovatka s ID 1 je vlevo od křižovatky s ID 0.
	2. Křižovatky 0 a 1 jsou propojeny.
	3. Existuje pouze jedno propojovací rameno mezi křižovatkami 0 a 1.

Tabulka C.14: Testovací případ #14

ID: #14	Spojování křižovatek 4
Popis:	Složitější případ spojování 1: různé typy a různé úhly natočení
Vstupní podmínky:	1. Simulační aplikace běží a klientská aplikace je na ní připojena.
	2. Simulační situace je prázdná.
Postup:	1. Vložíme první křižovatku: č. Konfigurace 1, "Angle"55, zašktrneme "Cars".
	2. Vybereme křižovatku s ID 0 a podle testovacího případu #2 vložíme křižovatku s těmito parametry: "Direction": "Left", konfigurace č. 2, "Angle": 0.
	3. Vybereme křižovatku s ID 1 a vložíme křižovatku s těmito parametry: "Direction": "Left", konfigurace č. 3, "Angle":180.
	4. Vybereme křižovatku s ID 2 a vložíme křižovatku s těmito parametry: "Direction": "Above", konfigurace č. 4, "Angle":0.
	5. Vybereme křižovatku s ID 3 a vložíme křižovatku s těmito parametry: "Direction": "Above", konfigurace č. 5, "Angle":270.
	6. Vybereme křižovatku s ID 4 a vložíme křižovatku s těmito parametry: "Direction": "Right", konfigurace č. 6, "Angle":0.
	7. Vybereme křižovatku s ID 5 a vložíme křižovatku s těmito parametry: "Direction": "Below", konfigurace č. 7, "Angle":0.
Očekávaný výsledek:	1. Jsou propojeny následující křižovatky: 1 a 2, 2 a 3, 3 a 4, 4 a 5, 5 a 6.
	2. Křižovatka s ID 0 není spojena s žádnou křižovatkou.
	3. Spojení odpovídá obrázku.

Tabulka C.15: Testovací případ #15

ID: #15	Spojování křižovatek 5
Popis:	Složitější případ spojování 2: jedna křižovatka může být propojena s více křižovatkami
Vstupní podmínky:	1. Simulační aplikace běží a klientská aplikace je na ní připojena.
	2. Simulační situace je prázdná.
Postup:	1. Vložíme první křižovatku: č. Konfigurace 15, "Angle"0, zašktrneme "Cars".
	2. Vybereme křižovatku s ID 0 a vložíme křižovatku s těmito parametry: "Direction": "Left", konfigurace č. 15, "Angle":0.
	3. Vybereme křižovatku s ID 0 a vložíme křižovatku s těmito parametry: "Direction": "Right", konfigurace č. 15, "Angle":0.
	4. Vybereme křižovatku s ID 0 a vložíme křižovatku s těmito parametry: "Direction": "Above", konfigurace č. 15, "Angle":0.
	5. Vybereme křižovatku s ID 0 a vložíme křižovatku s těmito parametry: "Direction": "Below", konfigurace č. 15, "Angle":0.
Očekávaný výsledek:	1. Křižovatka s ID 0 je propojena se 4 křižovatkami.
	2. Každé rameno křižovatky ID 0 je napojeno na nějaké propojovací rameno.

Tabulka C.16: Testovací případ #16

ID: #16	Odstraňování křižovatek
Popis:	Křižovatka napojená na další křižovatky lze odstranit a poté opět přidat na stejné místo
Vstupní podmínky:	1. Simulační aplikace běží a klientská aplikace je na ní připojena.
	2. Simulační situace je vytvořena podle testovacího případu #15.
Postup:	1. Ostraníme křižovatku s ID 0 (dle testovacího případu #4).
	2. Vybereme křižovatku s ID 4 a vložíme křižovatku s těmito parametry: "Direction": "Above", konfigurace č. 15, "Angle":0.
Očekávaný výsledek:	Křižovatka s ID 5 je propojena se 4 křižovatkami.
	Každé rameno křižovatky ID 5 je napojeno na nějaké propojovací rameno.

Tabulka C.17: Testovací případ #17

ID: #17	Spuštění a pozastavení vizualizace
Popis	Vizualizace může být spuštěna a poté pozastavena.
Vstupní podmínky:	1. Simulační aplikace běží a klientská aplikace je na ní připojena.
	2. Simulační situace je vytvořena podle testovacího případu #1.
Postup:	1. Klikneme na tlačítko "Run simulation".
	2. Vyčkáme do simulačního času 20.
	3. Klikneme na tlačítko "Pause simulation".
	4. Klikneme na tlačítko "Run simulation".
Očekávaný výsledek:	1. Po provedení kroku 1. začnou do situace postupně vjíždět vozidla. Semafory mění své stavy.
	2. Po pozastavení vizualizace se vozidla přestanou hýbat. Semafory nemění své stavy. V postranním panelu se u údaje "Simulation time"zobrazuje hodnota "20.0".
	3. Po opětovném spuštění simulace se vozidla opět rozpohybují ze svých původních pozic. Semafory mění své stavy.

Tabulka C.18: Testovací případ #18

ID: #18	Spuštění a zastavení vizualizace
Popis	Vizualizace může být spuštěna a poté pozastavena.
Vstupní podmínky:	1. Simulační aplikace běží a klientská aplikace je na ní připojena.
	2. Simulační situace je vytvořena podle testovacího případu #1.
Postup:	1. Klikneme na tlačítko "Run simulation".
	2. Vyčkáme do simulačního času 20.
	3. Klikneme na tlačítko "Stop simulation".
	4. Klikneme na tlačítko "Run simulation".
Očekávaný výsledek:	1. Po provedení kroku 1. začnou do situace postupně vjíždět vozidla. Semafory mění své stavy.
	2. Po zastavení vizualizace vozidla zmizí. Semafory se nastaví do původního stavu. V postranním panelu se u údaje "Simulation time"zobrazuje hodnota "0".
	3. Po opětovném spuštění simulace se simulace spustí znovu.

Tabulka C.19: Testovací případ #19

ID: #19	Nastavení hustoty dopravy
Popis	Z klientské aplikace lze měnit hustotu za běhu simulace.
Vstupní podmínky:	1. Simulační aplikace běží a klientská aplikace je na ní připojena.
	2. Simulační situace je vytvořena podle testovacího případu #1.
Postup:	1. Klikneme na tlačítko "Run simulation".
	2. Vyčkáme přibližně 20 sekund simulačního času.
	3. Posuvník pro nastavení hustoty dopravy posuneme zcela doprava.
	4. Vyčkáme přibližně 10 sekund simulačního času.
	5. Posuvník pro nastavení hustoty dopravy posuneme zcela doleva.
	6. Vyčkáme přibližně 50 sekund simulačního času.
Očekávaný výsledek:	1. Po provedení kroku 4. je patrné, že je v simulaci více vozidel než před prvním nastavením hustoty a rychleji přibývají.
	2. Po provedení kroku 6. je patrné, že je v simulaci méně vozidel než před prvním nastavením hustoty a pomaleji přibývají.

Tabulka C.20: Testovací případ #20

ID: #20	Nastavení signálního programu
Popis	Z klientské aplikace lze měnit signální program křižovatky za běhu simulace.
Vstupní podmínky:	1. Simulační aplikace běží a klientská aplikace je na ní připojena.
	2. Simulační situace je prázdná.
Postup:	1. Vložíme první křižovatku: č. konfigurace 1, "Angle"55, zaškrtneme "Cars".
	2. Klikneme na tlačítko "Run simulation".
	3. Ve výběru signálního programu vybereme signální program jiný než aktuálně běžící.
Očekávaný výsledek:	1. V postranním panelu pozorujeme změnu signálního programu na námi vybraný.
	2. Změna signálního programu není okamžitá.

Tabulka C.21: Testovací případ #21

ID: #21	Automatické odebírání vozidel z vizualizace
Popis	Po dokončení své trasy se vozidlo z vizualizace odstraní.
Vstupní podmínky:	1. Simulační aplikace běží a klientská aplikace je na ní připojena.
	2. Simulační situace je vytvořena podle testovacího případu #1.
Postup:	1. Klikneme na tlačítko "Run simulation".
	2. Vybereme si libovolné vozidlo a pozorujeme jeho pohyb.
Očekávaný výsledek:	1. Jakmile se vozidlo přiblíží ke jednomu z otevřených konců dopravní situace, je odebráno.

Tabulka C.22: Testovací případ #22

ID: #22	On-line informace o vozidle
Popis	V klientské aplikaci lze sledovat on-line informace o vybraném vozidle.
Vstupní podmínky:	1. Simulační aplikace běží a klientská aplikace je na ní připojena.
	2. Simulační situace je vytvořena podle testovacího případu #1.
Postup:	1. Klikneme na tlačítko "Run simulation".
	2. Klikneme na libovolné vozidlo a v postranním panelu pozorujeme jeho on-line informace.
Očekávaný výsledek:	1. On-line informace o vozidlu (rychlost, čekací doba na semaforu, ujetá vzdálenost) odpovídají pozorovanému vozidlu.

Tabulka C.23: Testovací případ #23

ID: #23	On-line informace o křižovatce
Popis	V klientské aplikaci lze sledovat on-line informace o vybrané křižovatce.
Vstupní podmínky:	1. Simulační aplikace běží a klientská aplikace je na ní připojena.
	2. Simulační situace je vytvořena podle testovacího případu #1.
Postup:	1. Klikneme na tlačítko "Run simulation".
	2. Vybereme křižovatku s ID 0 a v postranním panelu pozorujeme její on-line informace.
Očekávaný výsledek:	1. On-line informace o křižovatce (aktuální fáze, čas příštího přepnutí fáze) odpovídají pozorované křižovatce.

Tabulka C.24: Testovací případ #24

ID: #24	Připojení k simulační aplikaci
Popis:	Pokud se nedaří připojit k serveru, klient nabídne změnu údajů pro připojení.
Vstupní podmínky:	1. Simulační aplikace běží a klientská aplikace je na ní připojena.
Postup:	1. Vypneme simulační aplikaci.
	2. Aktualizujeme stránku s webovou aplikací klávesou F5.
	3. Do otevřeného modálního okna zkusíme zadat libovolné údaje a kliknout na tlačítko "Connect".
	4. Spustíme simulační aplikaci lokálně s konfigurací, kde je port specifikován jako 8080.
	5. V klientské aplikaci zadáme údaje "localhost"a "8080".
Očekávaný výsledek:	1. Po provedení kroku 2. se zobrazí se chybová hláška která značí, že se k serveru nedalo připojit.
	2. Po odkliknutí se zobrazí modální okno pro připojení na specifickou IP a port.
	3. Se simulační situací nebude možné provádět žádné akce dokud není připojení obnoveno.
	4. Po provedení kroku 5. se klientská aplikace korektně připojila k simulační aplikaci.

Tabulka C.25: Testovací případ #25

ID: #25	Chybový stav 1: Žádné typy vozidel
Popis:	Nelze přidat první křižovatku bez specifikace typů vozidel.
Vstupní podmínky:	1. Simulační aplikace běží a klientská aplikace je na ní připojena.
	2. Simulační situace je prázdná.
Postup:	1. Klikneme na tlačítko "Add first intersection".
	2. V modálním okně vybereme libovolnou konfiguraci a klikneme na tlačítko "Add intersection".
Očekávaný výsledek:	1. Zobrazí se chybová hláška oznamující, že je potřeba před odesláním specifikovat typy vozidel pro simulaci.
	2. Křižovatka se do situace nepřidá (počet křižovatek v situaci je stále 0).
	3. Modální okno pro přidání křižovatky zůstane otevřené s původními údaji.

Tabulka C.26: Testovací případ #26

ID: #26	Chybový stav 2: XML konfigurace je prázdný soubor
Popis:	Nelze přidat křižovatku bez vybrané XML konfigurace.
Vstupní podmínky:	1. Simulační aplikace běží a klientská aplikace je na ní připojena.
	2. Simulační situace je prázdná.
Postup:	1. Klikneme na tlačítko "Add first intersection".
	2. V modálním okně vybereme libovolnou kombinaci typů vozidel a klikneme na "Add intersection".
Očekávaný výsledek:	1. Zobrazí se chybová hláška oznamující, že je potřeba před odesláním specifikovat konfiguraci.
	2. Křižovatka se do situace nepřidá (počet křižovatek v situaci je stále 0).
	3. Modální okno pro přidání křižovatky zůstane otevřené s původními údaji.

Tabulka C.27: Testovací případ #27

ID: #27	Chybový stav 3: Chyba v XML konfiguraci
Popis:	Nelze přidat použít nevalidní XML soubor jako konfiguraci křižovatky.
Vstupní podmínky:	1. Simulační aplikace běží a klientská aplikace je na ní připojena.
	2. Simulační situace je prázdná.
Postup:	1. Klikneme na tlačítko "Add first intersection".
	2. V modálním okně vybereme libovolnou kombinaci typů vozidel.
	3. Vybereme libovolný XML soubor, který však není XML konfigurací dopravního řadiče sX, a klikneme na tlačítko "Add intersection".
Očekávaný výsledek:	1. Zobrazí se chybová hláška oznamující, že zaslaný soubor s konfigurací nebylo možné zpracovat.
	2. Křižovatka se do situace nepřidá (počet křižovatek v situaci je stále 0).
	3. Modální okno pro přidání křižovatky zůstane otevřené s původními údaji.

Tabulka C.28: Testovací případ #28

ID: #28	Chybový stav 4: Přidání křižovatky na obsazené místo
Popis:	Nelze přidat křižovatku na obsazené místo v situaci.
Vstupní podmínky:	1. Simulační aplikace běží a klientská aplikace je na ní připojena.
	2. Simulační situace je vytvořena podle testovacího případu #11.
Postup:	1. Vybereme křižovatku s ID 0 a vložíme křižovatku s těmito parametry: "Direction": "Left", konfigurace libovolná, "Angle":libovolný úhel.
Očekávaný výsledek:	1. Zobrazí se chybová hláška oznamující, že je uvažované místo již obsazené jinou křižovatkou.
	2. Křižovatka se do situace nepřidá (počet křižovatek v situaci je stále 2).
	3. Modální okno pro přidání křižovatky zůstane otevřené s původními údaji.

Tabulka C.29: Testovací případ #29

ID: #29	Chybový stav 5: Neplatné směry vstupního dopravního pruhu
Popis:	Nelze mít v rámci jednoho ramene dopravní pruhy, které by mohly způsobit kolizní situaci.
Vstupní podmínky:	1. Simulační aplikace běží a klientská aplikace je na ní připojena.
	2. Simulační situace je vytvořena podle testovacího případu #7.
Postup	1. Vybereme jeden z dopravních pruhů s ID 0-0si_0 nebo 0-0si_1.
	2. Klikneme na tlačítko "Add new input lane".
	3. V modálním okně zaškrtneme všechny směry a klikneme na tlačítko "Add lane".
Očekávaný výsledek:	1. Zobrazí se chybová hláška oznamující, že nově přidaný pruh by vytvořil možnou kolizní situaci.
	2. Dopravní pruh se do situace nepřidá (počet vstupních dopravních pruhů v tomto rameni je stále 2).
	3. Modální okno pro přidání vstupního dopravního pruhu zůstane otevřené s původními údaji.

Tabulka C.30: Testovací případ #30

ID: #30	Chybový stav 6: Odebrání posledního dopravního pruhu pro danou signální skupinu.
Popis:	Pro každou signální skupinu musí být přítomen alespoň jeden vstupní dopravní pruh.
Vstupní podmínky:	1. Simulační aplikace běží a klientská aplikace je na ní připojena.
	2. Simulační situace je vytvořena podle testovacího případu #1.
Postup	1. Vybereme jeden z dopravní pruh s ID 0-0si_0.
	2. Klikneme na tlačítko "Delete input lane".
Očekávaný výsledek:	1. Zobrazí se chybová hláška oznamující, že námi odstraňovaný dopravní pruh je jediný pro svou signální skupinu.
	2. Dopravní pruh se ze situace neodstraní (počet vstupních dopravních pruhů v tomto rameni je stále 1).

Tabulka C.31: Testovací případ #31

ID: #31	Výstup simulace
Popis:	Výstup simulace je dostupný po započetí simulace.
Vstupní podmínky:	1. Simulační aplikace běží a klientská aplikace je na ní připojena.
	2. Simulační situace je vytvořena podle testovacího případu #1.
Postup	1. Klikneme na tlačítko "Run simulation".
	2. Klikneme na tlačítko "Get simulation output".
Očekávaný výsledek:	1. Po provedení vizualizace prvního simulačního kroku se zobrazí tlačítko "Get simulation output".
	2. Po provedení bodu 2. se zobrazí modální okno se statistickými údaji a grafy, které korespondují s dosavadním průběhem Vizualizace .
	3. Vizualizace je pozastavena.

Příloha D

Testovací případy pro white-box testování

Tabulka D.1: Testovací případ #101

ID: #101	Verifikace vizualizace 1
Popis:	Základní test statických dat pro verifikaci vizualizace proti SUMO vizualizaci
Vstupní podmínky:	1. Simulační aplikace běží a klientská aplikace je na ní připojena.
	2. Simulační situace obsahuje testovací konfiguraci č. 14 na pozici [0,0] a konfiguraci č. 15 na pozici [1,0]
	3. Simulační aplikace využívá grafické uživatelské rozhraní (debug mód).
Postup:	1. V klientské aplikaci klikneme na tlačítko "Start simulation".
	2. Spustí se grafické uživatelské rozhraní SUMO. (Simulace se nespustí bez stisknutí tlačítka "Start simulation"v grafickém uživatelském rozhraní SUMO.)
	3. Manuálně zkontrolujeme vzájemnou polohu křižovatek pomocí ID ve webové a SUMO vizualizaci.
	4. Pro každý dopravní pruh: Manuálně zkontrolujeme polohu a směry dopravního pruhu pomocí ID ve webové a SUMO vizualizaci.
Očekávaný výsledek:	1. Křižovatka 14 je vlevo od křižovatky 15 v SUMO vizualizaci.
	2. Dopravní pruhy jsou na stejných pozicích v SUMO vizualizaci i ve webové vizualizaci.
	3. Vstupní dopravní pruhy mají stejné směry v SUMO vizualizaci i ve webové vizualizaci.

Tabulka D.2: Testovací případ #102

ID: #102	Verifikace vizualizace 2
Popis:	Porovnání spojení křižovatek ve vizualizaci a v SUMO vizualizaci
Vstupní podmínky:	1. Simulační aplikace běží a klientská aplikace je na ní připojena.
	2. Simulační situace je vytvořena podle testovacího případu #14.
	3. Simulační aplikace využívá grafické uživatelské rozhraní (debug mód).
Postup:	1. Spustíme simulaci podle testovacího případu #101.
	2. Manuálně zkontrolujeme vzájemnou polohu křižovatek pomocí ID ve webové a SUMO vizualizaci.
	3. Manuálně zkontrolujeme spojení křižovatek z pohledu ramen které spojují.
	4. Manuálně zkontrolujeme spojení křižovatek z pohledu jejich polohy a tvaru.
Očekávaný výsledek:	1. Počet křižovatek je stejný v obou vizualizacích.
	2. Poloha všech křižovatek je stejná v obou vizualizacích
	3. Všechna spojení křižovatek spojují stejná ramena svých křižovatek a tvar v obou vizualizacích.
	4. Všechna spojení křižovatek mají stejnou polohu a tvar v obou vizualizacích.

Tabulka D.3: Testovací případ #103

ID: #103	Verifikace vizualizace 3
Popis:	Porovnání pozic vozidel ve vizualizaci a v SUMO vizualizaci
Vstupní podmínky:	1. Simulační aplikace běží a klientská aplikace je na ní připojena.
	2. Simulační situace je vytvořena podle testovacího případu #14.
	3. Simulační aplikace využívá grafické uživatelské rozhraní (debug mód).
Postup:	1. Spustíme simulaci podle testovacího případu #101.
	2. Po 30s simulačního času pozastavíme vizualizaci v klientské aplikaci.
	3. Manuálně zkontrolujeme počet vozidel v obou vizualizacích.
	4. Manuálně zkontrolujeme každé vozidlo: jeho ID, pozici, natočení a signalizaci.
Očekávaný výsledek:	1. Počet vozidel je stejný v obou vizualizacích.
	2. U všech vozidel odpovídá jejich ID, pozice, natočení a signalizaci.

Tabulka D.4: Testovací případ #104

ID: #104	Verifikace vizualizace 4
Popis:	Porovnání pozic vozidel ve vizualizaci a v SUMO vizualizaci
Vstupní podmínky:	1. Simulační aplikace běží a klientská aplikace je na ní připojena.
	2. Simulační situace je vytvořena podle testovacího případu #14.
	3. Simulační aplikace využívá grafické uživatelské rozhraní (debug mód).
Postup:	1. Spustíme simulaci podle testovacího případu #101.
	2. Po 30s simulačního času pozastavíme vizualizaci v klientské aplikaci.
	3. Manuálně zkontrolujeme světelné signalizace na křižovatkách.
	4. Manuálně zkontrolujeme aktuální fáze jednotlivých křižovatek.
	5. Manuálně zkontrolujeme aktuální signální programy jednotlivých křižovatek.
Očekávaný výsledek:	1. Všechny světelné signalizace na všech křižovatkách jsou stejné v obou vizualizacích.
	2. Všechny křižovatky mají stejné aktuální fáze v obou vizualizacích.
	3. Všechny křižovatky mají stejné aktuální signální programy v obou vizualizacích.

Příloha E

Obsah přiloženého CD

Obsah přiloženého CD má následující adresářovou strukturu:

- **client/** - zdrojové soubory potřebné pro běh klientské aplikace
- **configurations/** - testovací konfigurační soubory dopravních řadičů ve formátu XML
- **simulator/** - zdrojové a binární soubory potřebné pro běh simulační aplikace
- **tex-src/** - zdrojové soubory textové části diplomové práce
- **thesis/** - tato diplomová práce ve formátu PDF
- **README.txt** - návod ke spuštění aplikace