Univerzita Pardubice

Fakulta Elektrotechniky a Informatiky

Management mikroskopické dopravní simulace s využitím Aimsun Next API

Diplomová práce

2025 Bc. Vladimír Josefy

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2024/2025

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Vladimír Josefy**
Osobní číslo:	**I23275**
Studijní program:	**N0613A140007 Informační technologie**
Téma práce:	**Management mikroskopické dopravní simulace s využitím Aimsun Next API**
Zadávající katedra:	**Katedra softwarových technologií**

Zásady pro vypracování

Cílem diplomové práce je vytvořit externí (simulační) aplikaci pro řízení dopravní mikroskopické simulace v nástroji Aimsun Next s využitím Aimsun Next API.
Teoretická část práce představí prostředí simulačního nástroje Aimsun Next, dále bude provedena rešerše možností nástroje Aimsun s ohledem na realizaci mikroskopických simulací zahrnující simulaci stochastických jevů představujících dopravní nehody, dlouhodobé uzávěry a aplikaci možných dopravních opatření. Cílem analýzy je vyhodnotit možnosti automatizovaného provádění simulací za účelem analýzy řady různých situací a aplikování důsledků různých dopravních opatření.
V praktické části bude realizována externí řídící aplikace, která bude spolupracovat s Aimsun Next API (resp. s rozhraním interagujícím s Aimsun Next API), která představí základní možnosti vyvolání nehod a omezení v probíhající mikroskopické simulace a následné provedení dopravních opatření za účelem snížení negativních důsledků vzniklých omezení. Realizovaná aplikace musí být použitelná a parametrizovatelná pro libovolný mikroskopický simulační model.

Rozsah pracovní zprávy: **50 – 60 stran**
Rozsah grafických prací:
Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

PRATA, Stephen. Mistrovství v C++. 4., aktualiz. vyd. Brno: Computer Press, 2013, 1176 s. Bestseller (Computer Press). ISBN 978-80-251-3828-1.
PECINOVSKÝ, Rudolf. *Python: kompletní příručka jazyka pro verzi 3.9*. Praha: Grada Publishing, 2020. Knihovna programátora (Grada). ISBN 978-80-271-1269-2.

Vedoucí diplomové práce: **Ing. Roman Diviš, Ph.D.**
Katedra softwarových technologií

Datum zadání diplomové práce: **10. dubna 2025**
Termín odevzdání diplomové práce: **22. srpna 2025**

L.S.

prof. Ing. Petr Doležel, Ph.D. v.r.
děkan

prof. Ing. Antonín Kavička, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 10. dubna 2025

Prohlašuji:

Práci s názvem „Management mikroskopické dopravní simulace s využitím Aimsun Next API“ jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 7/2019 Pravidla pro odevzdávání, zveřejňování a formální úpravu závěrečných prací, ve znění pozdějších dodatků, bude práce zveřejněna prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 22. 8. 2025

Bc. Vladimír Josefy v.r.

PODĚKOVÁNÍ

Tímto chci poděkovat vedoucímu diplomové práce, panu Ing. Romanu Divišovi, Ph.D. za pomoc a cenné připomínky, které mi pomohly při zpracování práce. Dále chci poděkovat své rodině za podporu v průběhu studia a svojí přítelkyni za trpělivost a podporu při vypracovávání této práce.

ANOTACE

Cílem práce je návrh a realizace externí aplikace pro řízení mikroskopické dopravní simulace v nástroji Aimsun Next s využitím Aimsun Next API. Výsledná aplikace musí být parametrizovatelná a použitelná pro libovolný mikroskopický simulační model. Teoretická část práce poskytuje vhled do mikrosimulačních metodik a představuje prostředí Aimsun Next společně s rozhraním Aimsun Next API. Praktická část navazuje vlastním návrhem a implementací aplikace.

KLÍČOVÁ SLOVA

mikrosimulace, aimsun next API, externí řídící aplikace, řízení dopravy

TITLE

Management of microscopic traffic simulation using the Aimsun Next API

ANNOTATION

The aim of this thesis is to design and implement an external control application for managing a microscopic traffic simulation using the Aimsun Next API. The resulting application must be parameterizable and applicaple for any microscopic simulation model. The theoretical part surveys microsimulation methodologies and introduces the Aimsun Next environment together with the Aimsun Next API. The practical section details the actual design and implementation of the developed application.

KEYWORDS

microsimulation, aimsun next API, external control application, traffic control

OBSAH

SEZNAM ILUSTRACÍ A TABULEK

SEZNAM VÝPISŮ

SEZNAM ZKRATEK A ZNAČEK

API	Application Programming Interface
ASGI	Asynchronous Server Gateway Interface
CEM	Cross-Entropy Method
COM	Component Object Model
DTA	Dynamic Traffic Assignment
DTO	Data Transfer Object
DUE	Dynamic User Equilibrium
GIL	Global Interpreter Lock
HTML	Hypertext Markup Language
HTPP	Hypertext Transfer Protocol
IDM	Intelligent Driver Model
JSON	Javascript Object Notation
LSE	Least Squared Errors
MFC	Microscopic Free-Flow Acceleration
ML	Maximum Likelihood
MOBIL	Minimizing Overall Braking Induced By Lane Changes
MVC	Model-View-Controller
OD	Origin-Destination
REST	Representational State Transfer
SRC	Stochastic Route Choice
SWIG	Simplified Wrapper and Interface Generator
UI	User Interface

ÚVOD

Rostoucí nároky kladené na kapacitu a spolehlivost silniční infrastuktury si žádají nástroje, které umožňují ověřovat návrhy a dopravní strategie ještě před jejich vlastním nasazením do reálného provozu. Mikroskopická simulace představuje v tomto ohledu dobrý přístup, umožňující modelování chování jednotlivých vozidel i jejich vzájemných interakcí v rámci definované dopravní sítě. Díky tomu je možné zkoumat dopady různých dopravních jevů jako jsou dopravní nehody či dlouhodobé uzavírky a testovat různé varianty žádaných opatření.

Aimsun Next patří mezi široce používané komerční simulační platformy, které tyto úlohy podporují, současně nabízí programové rozhraní pro integraci s externími nástroji a pro automatizaci experimentů. Díky zmiňovanému rozhraní je tak možno testovat různé přístupy k rozličným úlohám řízení dopravy a zasahovat do běžící simulace.

Cílem diplomové práce je návrh a implementace externí aplikace pro řízení mikroskopické dopravní simulace v prostředí Aimsun Next za využití rozhraní Aimsun Next API. Teoretická část poskytuje technicky orientovaný přehled simulačních metodik s detailním zaměřením na problematiku mikrosimulace, představuje prostředí Aimsun Next a nastiňuje interakci s nástrojem pomocí rozhraní. Praktická část se věnuje návrhu architektury externí řídící aplikace společně s její implementací, poslední kapitola také poskytuje krátký popis integrace externí aplikace s nástrojem Aimsun Next.

1 Přehled dopravních simulací

Tato kapitola se zaměřuje na vymezení základních pojmů oblasti modelování a simulace a následným seznámením čtenáře s problematikou dopravních simulací, přičemž pozornost je věnována zejména tzv. mikrosimulaci.

1.1 Vymezení základních pojmů

Podkapitola je zpracována dle zdroje [1].

1.1.1 Systém

Narozdíl od běžného jazyka má v oblasti modelování a simulací pojem „systém“ odlišný a přesněji vymezený význam. Označuje abstrakci reálného světa, tvořenou souborem vzájemně propojených prvků, které společně vykazují určité chování. Takový systém se může sestávat z různorodých komponent (technických, biologických aj.) a jejich vztahů. Představuje tedy určitý výsek reality vymezený pro účely zkoumání jeho struktury či funkce.

Zásadní roli při práci se systémem hraje jeho vymezení, tj. určení hranic sloužících pro rozlišení mezi systémem a jeho okolím. Klíčové je dále určení vstupů a výstupů, což slouží pro zachycení toho, jak systém ovlivňuje okolí a také jak je svým okolím ovlivňován.

S ohledem na výměnu informací, energie či materiálu s okolím je možné systémy rozlišit na otevřené a uzavřené dle přítomnosti nebo absence zmiňovaného chování. Systémy lze také kategorizovat dle přístupu k abstrahování času, je-li od tohoto úkonu upuštěno a čas nehraje žádnou roli, hovoří se pak o systému statickém. V případě, kdy vzniká snaha zachytit systém v sérii okamžiků, je pak možné označit systém jakožto systém dynamický, přičemž dynamické systémy jsou klíčové právě pro oblast simulace.

1.1.2 Model

Představuje cíleně vytvořenou reprezentaci reálného systému, která s ním udržuje přesně definovaný vztah. Mezi prvky modelovaného systému a odpovídajícími prvky modelujícího systému existuje bijekce zahrnující prvky i jejich vlastnosti (atributy), hodnoty těchto atributů pak odpovídají konkrétním vztahům či zákonitostem.

Pro potřeby simulace se však uplatňují výhradně simulační modely, kde modelovaný i modelující systém jsou systémy dynamické. Simulační model je dále typický tím, že zachycuje časový průběh chování systému, tj. modelující systém zachycuje (modeluje) každý okamžik modelovaného systému a zachovává také posloupnost těchto okamžiků.

V případě simulačních modelů je vhodnější nazývat modelovaný systém jakožto systém simulovaný a modelující systém označovat systémem simulujícím. V běžné řeči se však často termín model nesprávně vztahuje právě na modelující systém. Tato terminologická nepřesnost se vztahuje i na simulující systém, který bývá běžně označován pojmy simulační model případně simulátor.

1.1.3 Simulace

Pojem simulace lze vymezit následovně: *„Simulace je výzkumná technika, jejíž podstatou je náhrada zkoumaného dynamického systému jeho simulátorem s tím, že se simulátorem se experimentuje s cílem získat informace o původním zkoumaném dynamickém systému.“* [1]

Je důležité poznamenat, že právě snaha získat o simulovaném systému informace je pro metodiku simulace stěžejní. Prosté nahrazení modelovaného (simulovaného) systému systémem modelujícím (simulujícím) nestačí, tuto situaci je však možné označit termínem emulace.

Nezbytné tedy je, aby simulace vedla k poznání vlastností původního systému, to však předpokládá, že simulátor věrně zachycuje realitu a výsledky experimentů umožňují formulovat validní a spolehlivé závěry. Z tohoto důvodu je součástí procesu simulace také verifikace modelu, tj. systematické ověření, že model byl sestaven správně a tudíž odpovídá zamýšlené reprezentaci reálného systému.

1.1.4 Dopravní simulace

Jde o klíčový nástroj dopravního inženýrství sloužící k analýze a predikci chování dopravních systémů. Umožňuje zkoumat situace, které jsou pro analytické výpočty příliš složité a bezpečně testovat dopravní strategie či infrastrukturu v digitálním prostředí namísto reálného provozu. Dopravní simulace se rozvíjí již od 50. let 20. století a staly se nepostradatelným pomocníkem při plánování, návrhu a řízení dopravních sítí. [2]

1.2 Klasifikace dopravních modelů

Jedním z hlavních způsobů, jak je možné klasifikovat dopravní modely je klasifikace dle úrovně agregace, tj. do jaké míry detailu popisují dopravu. Modely se dle úrovně agregace člení na modely makroskopické, mezoskopické a (sub-)mikroskopické. Tyto kategorie se odlišují způsobem, jak reprezentují dopravu a také používají rozdílné matematické přístupy. [3]

1.2.1 Makroskopické modely

Makroskopické modely popisují dopravní proud analogicky k fyzikálním systémům, jako jsou kapaliny či plyny, proto mohou být také označovány pojmem hydrodynamické modely. Základní veličiny, se kterými tyto modely pracují, jsou lokálně agregované charakteristiky dopravního proudu: hustota dopravy $\rho(x,t)$, dopravní tok $Q(x,t)$, střední rychlost $V(x,t)$ a případně také rozptyl rychlosti $\sigma_V^2(x,t)$. Tyto veličiny nejsou vztaženy k jednotlivým vozidlům, avšak k určitým prostorovým úsekům infrastruktury a typicky se mění v čase a prostoru, čímž tvoří dynamická pole. Toto umožňuje modelům zachytit kolektivní jevy jako je například vývoj dopravní kongesce daného regionu nebo rychlost šíření dopravních vln. [3]

Jsou vhodné zejména v případech, kdy:

- není nutné zohledňovat efekty, které jsou obtížně popsatelné makroskopicky (např. změny jízdních pruhů, různé typy vozidel a řidičů),
- předmětem zkoumání jsou výhradně makroskopické veličiny,
- je rozhodující výpočetní čas simulace (např. v real-time aplikacích),
- vstupní data pocházejí z heterogenních nebo nekonzistentních zdrojů.

Výpočetní čas a schopnost integrace heterogenních zdrojů dat jsou důležité zejména pro odhad a predikci dopravního stavu. V tomto procesu je budoucí stav dopravy předpovídán na konkrétním časovém horizontu a predikce jsou v kratších časových intervalech aktualizovány. Výstup predikcí je pak možné dále distribuovat prostřednictvím dopravního zpravodajství, proměnných dopravních značek nebo pomocí navigačních zařízení. [3]

1.2.2 Mikroskopické modely

Simulují pohyb každého jednotlivého vozidla a řidiče v síti. Jsou založeny na modelování individuálního chování, tj. každé vozidlo je autonomní entitou s vlastními charakteristikami a tyto entity pak kolektivně tvoří dopravní proud. [3]

Mikroskopický simulátor typicky postupuje v malých diskrétních časových krocích a v každém kroku aktualizuje pozici a rychlost všech vozidel dle zvolených modelů chování řidiče. [4] Díky tomu je možné zachytit detailní dynamiku dopravního proudu. Narozdíl od makromodelů jsou stavové proměnné mikroskopické povahy. Pro každé vozidlo α jsou sledovány:

- $x_\alpha(t)$ – pozice vozidla,
- $v_\alpha(t)$ – rychlost vozidla,
- $\dot{v}_\alpha(t)$ – akcelerace vozidla. [3]

Mikrosimulace představuje nejdetailnější a zpravidla nejpřesnější metodu, avšak za cenu nejvyšší výpočetní náročnosti a rozsáhlých požadavků na kalibraci vstupních parametrů. Proto nachází využití zejména pro lokální a taktické studie, např. hodnocení návrhu křižovatky či vedení jízdních pruhů, kde je tato úroveň detailu nepostradatelná. Díky schopnosti simulovat i silně populované situace a složité konfigurace, které jsou nad možnosti makromodelů, je mikrosimulace vhodná i pro analýzu dopadů nehod, řízení dopravy a dalších opatření na plynulost a bezpečnost provozu. [5]

Na obecnější rovině lze mikrosimulace vymezit jakožto vhodný nástroj v případech kdy:

- je snaha o modelování vlivu jednotlivých vozidel na dopravní proud,
- důležitým aspektem zkoumání je heterogenita provozu, tj. například specifická pravidla pro nákladní vozy a osobní automobily,
- je žádoucí zachycení lidského faktoru v rámci dopravy – reakční čas, chybné odhady, nepozornost nebo předvídání,
- vizualizuje se interakce mezi jednotlivými účastníky provozu (automobily, nákladní vozy, cyklisté, chodci),
- je potřeba modelovat okolní provoz, například pro využití ve vědeckých jízdních simulátorech za účelem fyziologické či psychické studie chování řidiče nebo pro využití ve hrách či v trenažérech. [3]

1.2.3 Mezoskopické modely

Představují stupeň mezi mikro- a makromodely, kombinují prvky obou přístupů. Obvykle sledují jednotlivá vozidla (obdobně jako mikromodely), jejich pohyb a interakce jsou však modelovány zjednodušeně či agregovaně. Dle zvolené metodiky existuje vícero variant mezoskopických modelů. [3] [4] [5]

Některé přístupy využívají k reprezentaci přesunu vozidla v rámci regionu fronty a fungují na principu systému diskrétních událostí, tj. čas simulace „skočí“ k nejbližší události, kterou bývá příchod vozidla na konec či začátek úseku nebo do uzlu. Příkladem může být přístup použitý v nástroji Aimsun Next, kde se vozidla po úsecích pohybují s předpokládanou průměrnou rychlostí a detailní poloha uvnitř úseku se neřeší. Simulace postupuje od události k události a jenom vozidla v čele zmiňovaných front jsou aktualizována častěji což snižuje výpočetní náročnost simulace. [4]

Dalším z přístupů mezoskopické simulace může být modelování chování skupin vozidel (tzv. vehicle platoon behaviour), kdy se vozidla pohybují ve „formaci“ při stejné rychlosti. Tento přístup umožňuje agregovat chování jednotlivých vozidel do kompaktních celků (platoon). To snižuje počet individuálních interakcí v modelu a zjednodušuje simulaci dopravního proudu. [6]

Ve srovnání s mikroskopickými modely jsou mezoskopické přístupy výpočetně méně náročné nejen z hlediska času, ale také s ohledem na množství potřebných dat pro kalibraci. Představují tak vhodný kompromis mezi výpočetní efektivitou a úrovní zachycených jevů v dopravním proudu. [7]

Tato efektivita umožňuje použití modelů i při simulaci rozsáhlých městských či regionálních sítí v akceptovatelném čase, díky menším nárokům na vstupní data jsou také výhodné pro aplikace, kde není k dispozici detailní informace o individuálním chování řidičů, ale je třeba analyzovat dopravní zatížení rozsáhlé sítě. [8]

Oproti mikroskopickým modelům mezoskopické přístupy často pracují s agregovanými veličinami a diskrétní aktualizací stavu, což snižuje úroveň zachycených detailů chování řidičů. V důsledku tak modely nemusí plně reprodukovat jemnější aspekty provozu, jako jsou heterogenita řidičů nebo dynamika plynulé změny rychlosti. [7] [8]

Naopak ve srovnání s makroskopickými modely poskytují mezomodely vyšší úroveň detailu, včetně individuálních trajektorií vozidel, které jsou však založené na zjednodušených pravidlech pohybu, toto zjednodušení však může vést k určitým odchylkám vůči realitě. [7] V modelu DTALite je například každé vozidlo definováno individuálně, ale jeho pohyb se řídí makroskopickými vztahy. [8]

Historicky byly mezoskopické modely méně rozšířené a to zejména kvůli technickým a metodologickým omezením. Zůstávaly spíše v doméně specializovaných výzkumných projektů bez dostupnosti pro praktické využití. [7] S rozvojem výpočetní techniky a dopravních simulátorů se však tyto modely dostávají do praktického využití. [4] [9]

2 Mikrosimulace dopravy

Mikroskopická simulace (mikrosimulace) modeluje provoz na úrovni jednotlivých vozidel a řidičů v čase. Vstupem mikrosimulátoru je podrobný popis dopravní sítě společně s parametry definujícími chování různých typů řidičů a vozidel. Tato kapitola podrobně popisuje, jak mikrosimulace funguje, jaké algoritmy využívá a jaké aspekty musí být zohledněny.

2.1 Vstupní data mikrosimulace

Aby byl mikrosimulační model schopen věrně reprodukovat skutečnou dopravní dynamiku, je potřeba jej budovat na základě dostatečně bohatého a detailního souboru vstupů. Přestože konkrétní formát a struktura dat se může dle voleného nástroje a zaměření studie lišit, praxe ukazuje, že valná většina mikrosimulačních analytických studií využívá z hlediska vstupních dat tři základní pilíře:

- geometrická data infrastruktury,
- data dopravního řízení,
- data dopravní poptávky.

K doplňujícím, avšak nezbytným vstupům patří rovněž údaje popisující vlastnosti vozidel a chování řidičů (délka vozidla, maximální akcelerace, míra agrese řidiče). Většina simulačních nástrojů dodává pro dané charakteristiky výchozí hodnoty, jelikož sběr dat v terénu může být pro tyto jemné charakteristiky náročný. Pro zachycení variability vozového parku jsou výchozí hodnoty podkládány statistickou distribucí. [10]

2.1.1 Geometrická data infrastruktury

Do modelu je potřeba zanést počet a délku jízdních pruhů, společně s návrhovou rychlostí. U křižovatek dále hrají důležitou roli data o odbočovacích pruzích a jejich akumulačních délkách společně s daty o poloměrech zaoblení nároží. Zdrojem těchto dat většinou bývají projektové výkresy, ortofotomapy, geografické informační systémy nebo případně terénní studie. [10]

2.1.2 Data dopravního řízení

Data obsahují zejména informace o poloze všech zařízení dopravního řízení (semafory, značky, přejezdová zabezpečovací zařízení aj.) společně s informací o nastavení jejich signálních plánů. Nejspolehlivější způsob, jak tyto data získat, představují databáze organizací, které jsou za provoz daného systému zodpovědné. Vždy, kdy je to možné, je žádoucí získání

signálních plánů s časovými razítky, je však nutno podotknout, že spoléhání se výhradně na obdržené plány není dobrý přístup. Doporučuje se vyčlenění kapacit na místní kontrolu v terénu, jelikož i drobné rozdíly mezi databází a skutečným systémem mohou simulaci značně ovlivnit. [10]

2.1.3 Data dopravní poptávky

Pro tvorbu výchozího modelu je nezbytné shromáždění dat o dopravní poptávce. Základem jsou průjezdné objemy vstupních profilů modelované oblasti a odbočovací proudy v křižovatkách, obvykle agregované do intervalů nepřesahujících 15 minut. Pro komplexnější scénáře (např. modelování situací odklonu dopravy) se tvoří matice původ-cíl – matice OD (origin-destination). [10]

2.1.4 Charakteristiky vozidel

Simulační software sice disponuje výchozím vozovým parkem, nicméně národní či regionální reprezentace se od zásobených výchozích hodnot může značně lišit. Zvláště podíl nákladních vozů může v závislosti na oblasti výrazně kolísat, od 2 % v městských ulicích během dopravní špičky až ke 40 % na tranzitních dálnicích. [10]

2.1.5 Charakteristiky řidičů

Modelování dopravního proudu vyžaduje i parametry lidského chování: agresivita, reakční čas, cílová rychlost, akceptované minimální mezery (při zipování, změnách pruhu apod.). Některé nástroje navíc dovolují definovat kooperativnost řidičů, jejich informovanost a míru dodržování pravidel dopravního provozu. [10]

2.1.6 Data o provozu a řízení dopravy

Lze je rozdělit do čtyř skupin: výstražné prvky, regulační prvky, informační prvky a monitorovací prvky. U všech prvků je nutné znát jejich typ a přesnou polohu. [10]

2.1.7 Data o cestovních podmínkách

V případě že jde o data relevantní pro cílovou studii, mohou se společně k datům dopravní poptávky přiřadit i další informace. Patří sem například meteorologická data (srážky, rychlost větru nebo viditelnost) nebo podrobné incidentní záznamy. [10]

2.2 Modelovací jádro mikrosimulace

Nejvýznamnější skupinu mikrosimulačních modelů tvoří modely sledu vozidel. Dynamiku provozu popisují z pohledu jednotlivých dvojic řidič–vozidlo. V užším pojetí se tyto modely

zaměřují pouze na situace, kdy vozidlo interaguje s jinými vozidly, přičemž volná jízda bez vlivu okolí je řešena samostatným modelem. V širším pojetí však zahrnují všechny provozní stavy, od volné jízdy až po stacionární provoz. [3]

Dle modelovaného chování je možné modely členit na:

- modely podélného řízení (modely sledu vozidel),
- modely příčného řízení,
- rozhodovací modely.

Ačkoliv uvedené členění zdůrazňuje rozdílné funkce jednotlivých modelů, moderní komplexní přístupy je navzájem propojují, důsledkem pak bývají komplexní integrované modely, tvořící výpočetní jádro většiny komerčních mikrosimulačních nástrojů. [3] [11]

2.2.1 Modely car-following

Prvotní modely sledu vozidel se datují do 50. let 20. století, obecně modelují pohyb a chování daného vozidla v závislosti na vozidle, které je v daném pruhu začleněno před daným vozidlem, tj. v závislosti na vozidlu vedoucím. [3] V případě, že by volná jízda požadovanou rychlostí vedla ke kolizi, následující vozidlo se považuje za omezené vozidlem vedoucím, v opačných situacích je následující vozidlo ve volném režimu. [12]

Většina modelů popisuje reakci následovníka pomocí zrychlení, některé využívají přímo rychlost vozidla. Některé formulace se soustředí pouze na situace skutečného sledu, jiné mohou pokrývat veškeré stavy. Úkolem modelu obecně bývá jednak určit v jakém režimu se vozidlo nachází a zároveň stanovit akci, kterou je nutné provést. [12]

Většina modelů používá k určení chování následovníka několik režimů, většinou existuje rozlišení do alespoň tří režimů:

- volná jízda – vozidlo v tomto režimu není omezeno a snaží se docílit požadované rychlosti,
- běžný sled – vozidlo v tomto režimu uzpůsobuje svoji rychlost s ohledem na vozidlo před ním,
- nouzové brzdění – režim prudkého brzdění za účelem vyvarování kolize. [12]

V této sekci budou rozebrány dva modely, zjednodušený Gippsův model a Intelligent Driver Model.

2.2.1.1 Gippsův model

Zpracováno dle zdroje [3]. Důležitým konceptem pro udržování adekvátní rychlosti s ohledem na prevenci kolizí je zavedení tzv. „bezpečné rychlosti“ $v_{safe}(s, v_l)$, ta závisí na vzdálenosti s od vedoucího vozidla a také na rychlosti vedoucího vozidla v_l. K odvození bezpečné rychlosti využívá model několik předpokladů:

1. Brzdění se provádí s konstatním zpomalením b.
2. Existuje čas reakce Δt, který je konstatní.
3. V případě, kdy vedoucí vozidlo kompletně zastaví, nedojde k redukci vzdálenosti mezi vozidly pod minimální hodnotu s_0.

Jak bylo zmíněno tento model je v porovnání s originální publikací od Gippse zjednodušený, první zjednodušení představuje zavedení předpokladu 3, což je nutné z důvodu diskretizace. Druhým zjednodušením je předpoklad konstatní rychlosti v době reakce, tj. ke změně rychlosti dochází až po uplynutí času reakce.

Za využití prvního předpokladu je možné odvodit brzdnou dráhu vedoucího vozidla.

$$\Delta x_l = \frac{v_l^2}{2b}$$

Druhý předpoklad pak upravuje brzdnou dráhu následujícího vozidla, konkrétně jí prodlužuje o reakční vzdálenost $v\Delta t$, kterou vozidlo urazí, než jeho řidič adekvátně zareaguje.

$$\Delta x = v\Delta t + \frac{v^2}{2b}$$

Předpokladu 3 je vyhověno v případě, že vzdálenost s mezi vozidly je větší než minimální vzdálenost s_0 a větší než rozdíl brzdných drah $\Delta x - \Delta x_l$, tudíž:

$$s \geq s_0 + v\Delta t + \frac{v^2}{2b} - \frac{v_l^2}{2b}$$

Za využití těchto vztahů je pak možno odvodit maximální bezpečnou rychlost:

$$v_{safe}(s, v_l) = -b\Delta t + \sqrt{b^2\Delta t^2 + v_l^2 + 2b(s - s_0)}$$

Výpočet aktualizované rychlosti je pak prováděn následujícím způsobem:

$$v(t + \Delta t) = min\,[v + a\Delta t, v_0, v_{safe}(s, v_l)]$$

Přičemž a představuje maximální zrychlení vozidla a v_0 je jeho cílová rychlost. Tento vztah reflektuje následující vlastnosti modelu:

- Čas simulačního kroku je roven reakčnímu času řidiče Δt.
- Pokud aktuální rychlost přesahuje hodnotu $v_{safe} - a\Delta t$ nebo $v_0 - a\Delta t$, pak v dalším simulačním kroku dosáhne rychlost vozidla $min\ (v_{safe},\ v_0)$.
- V ostatních případech vozidlo zrychluje dle hodnoty a, dokud není dosaženo $min\ (v_{safe},\ v_0)$.

V porovnání s minimalistickými modely je Gippsův model založen na několika relativně jednoduchých předpokladech a využívá parametry, které jsou jednoduché k interpretaci a pro které je snadné přiřazení realistických hodnot. Model je také možno považovat za robustnější, jelikož je schopen produkovat smysluplné výsledky pro širokou škálu hodnot parametrů.

Významnou nevýhodou tohoto přístupu je nerealistický model zrychlování a zpomalování. Z uvedených vztahů může zrychlení nabývat pouze tří hodnot: 0, a nebo $-b$. Výsledné chování řidiče je pak robotické a změny zrychlení jsou drastické a nerealistické. Brzdící manévry jsou navíc prováděny „plnou silou“, jelikož v modelu není rozlišena maximální míra zpomalení a komfortní míra zpomalení, kterou by za dané situace brzdil člověk. Pokud by však b bylo nastaveno jakožto pohodlná míra zpomalení a do modelu byl zaveden vícepruhový provoz či heterogenní vozový park, pak by model mohl produkovat kolize v případech, kdy zpomalení vedoucího vozidla přesahuje hodnotu b.

Gippsův model lze sumarizovat jakožto model, který je navzdory své jednoduchosti schopen produkovat relativně dobré výsledky, modifikované verze tohoto modelu jsou tak využity v některých komerčních simulačních nástrojích (např. Aimsun Next).

2.2.1.2 Intelligent Driver Model

Podkapitola je vypracována dle zdroje [3]. Intelligent driver model (IDM) představuje nejspíše nejjednodušší, avšak z hlediska úplnosti a prevence kolizí plnohodnotný časově spojitý model, který generuje realistické profily zrychlení a zároveň zaručuje věrohodné chování vozidel v naprosté většině situací jednoproudého silničního provozu.

Obdobně jako Gippsův model je založen na sadě předpokladů, charakterizují jej následující požadavky:

1. Zrychlení splňuje obecné podmínky úplnosti:

- Klesající závislost na vlastní rychlosti, tj. s rostoucí rychlostí akcelerace klesá.
- Rostoucí závislost na odstupu s, čím větší odstup tím více může vozidlo zrychlovat.
- Klesající závislost na rychlosti přibližování $\Delta v = v - v_l$ k vedoucímu vozidlu.
- Zajištění minimální mezery s_0, při porušení minimální mezery nedochází ke zpětnému pohybu.

2. Udržuje se bezpečný odstup $s_0 + vT$, kde s_0 reprezentuje minimální vzdálenost od nárazníku k nárazníku a T představuje časovou mezeru k vedoucímu vozidlu.
3. Existuje inteligentní brzdící strategie, která definuje, jak se vozidlo přibližuje k překážkám (vedoucí vozidlo, semafory aj.). Za normálních podmínek strategie pracuje s pohodlnou hodnotou zpomalení b, která plynule roste i klesá. V kritických situacích může hodnota zpomalení překročit hodnotu b, dokud není situace napravena, zbytek brzdícího manévru je opět prováděn s komfortní hodnotou b.
4. Přechody mezi jednotlivými režimy jsou plynulé, akcelerační funkce neobsahuje žádné skoky, tj. je spojitě diferencovatelná pro všechny tři proměnné s, v, v_l.
5. Parametry modelu by měly popisovat ideálně pouze jeden konkrétní aspekt chování řidiče, což je vhodné s ohledem na kalibraci modelu.

Zmíněné požadavky jsou realizované pomocí následujícího vztahu zrychlení:

$$\frac{dv}{dt} = a\left[1 - \left(\frac{v}{v_0}\right)^{\delta} - \left(\frac{s^*(v, \Delta v)}{s}\right)^2\right]$$

Akcelerace se v modelu skládá ze dvou klíčových komponent. Jednou z nich je porovnání aktuální rychlosti v se chtěnou rychlostí v_0. Druhou představuje porovnání aktuální vzdálenosti s se chtěnou vzdáleností s^*, přičemž chtěná vzdálenost od vedoucího vozidla je reprezentována jako:

$$s^*(v, \Delta v) = s_0 + \max\left(0, vT + \frac{v\Delta v}{2\sqrt{ab}}\right)$$

Kde $s_0 + vT$ je rovnovážný člen a výraz $\frac{v\Delta v}{2\sqrt{ab}}$ implementuje „inteligentní“ brzdící strategii.

Parametry modelu

Parametry modelu je možné osvětlit pomocí tří standardních situací. Při rozjezdu na volné vozovce vozidlo startuje maximálním zrychlením a, které s rostoucí rychlostí vozidla klesá a při přiblížení k požadované rychlosti v_0 se blíží nule. Kde exponent δ vyjadřuje míru

ubývání ($\delta \to \infty$ zachycuje akcelerační profil Gippsova modelu, hodnota $\delta = 1$ naopak produkuje nerealisticky plynulý akcelerační profil). Při jízdě za vedoucím vozidlem je odstup přibližně roven bezpečné vzdálenosti $s_0 + vT$, kterou určují minimální vzdálenost s_0 a minimální časová mezera T. Při dojíždění pomalejších vozidel většinou brzdění nepřevyšuje komfortní hodnotu b a akcelerační funkce zůstává plynulá i během přechodu mezi těmito situacemi. Tabulka 1 poskytuje přehled typických hodnot parametrů modelu pro dálniční a městský provoz.

Tabulka 1: Porovnání typických hodnot parametrů modelu IDM, zdroj: [3]

Parametr	**Typická dálniční hodnota**	**Typická městská hodnota**
Chtěná rychlost v_0	120 km/h	54 km/h
Časová mezera T	1,0 s	1,0 s
Minimální mezera s_0	2 m	2 m
Akcelerační exponent δ	4	4
Akcelerace a	1,0 m/s^2	1,0 m/s^2
Pohodlné zpomalení b	1,5 m/s^2	1,5 m/s^2

Jelikož model neudržuje explicitně reakční čas a akcelerační funkce je spojitá, reprodukuje spíše chování adaptivního tempomatu nežli lidského řidiče. Pro reprodukci chování lidských řidičů je však možné dodat aspekty jako jsou například chyby v odhadech, reakční čas apod.

2.2.2 Modely lane-changing a diskrétní rozhodování

Zpracováno dle zdroje [3]. Architektura mikroskopické simulace musí být schopna zachytit rozmanité dopravní situace, které vyžadují diskrétní rozhodování ze strany řidiče. Typickými příklady jsou manévry, jako je změna jízdního pruhu nebo volba vhodného chování při přibližování k semaforu, který má brzy přepnout na červenou.

Řidič ovlivňuje dynamiku provozu prostřednictvím akcelerace, brzdění a řízení směru jízdy. Detailní modelování řízení směru jízdy však spadá do oblasti submikroskopických modelů, které se zaměřují na fyzikální popis pohybu vozidla s větší mírou detailu. Pro účely mikroskopické simulace je proto uplatňována jistá míra abstrakce a příčné manévry jsou reprezentovány jako diskrétní rozhodnutí. Parametry jako délka trvání manévru či boční zrychlení nejsou brány v potaz; z hlediska simulace je manévr okamžitě považován za dokončený.

Pro rozhodování řidiče je možné vymezit čtyři úrovně:

- strategická úroveň, např. volba cíle a cesty,

- taktická úroveň, tj. příprava na budoucí manévry operativní úrovně,
- operativní úroveň, tj. vlastní okamžité rozhodování,
- post-decizní fáze, kdy jsou akce související s rozhodnutím simulovány.

Modelování taktické úrovně je velmi složitá záležitost, které se věnují primárně sofistikované komerční nástroje, další rozbor se tedy bude věnovat zejména operativní a post-decizní úrovni.

Jedním z přístupů k modelování rozhodování řidiče je chápat volbu manévru jakožto snahu o maximalizaci užitkové funkce. Užitková hodnota dané volby roste s hypotetickým podélným zrychlením, kterého by bylo možné dosáhnout. Tím, že se jako míra užitku používá přímo zrychlení, vzniká přirozené propojení mezi podélným modelem a modelem diskrétní volby bez nutnosti zavádět další samostatná kritéria a parametry. Příkladem může být model agresivního řidiče, jehož agresivita se projeví nejen při podélné akceleraci ale také při manévrech změny jízdního pruhu bez nutnosti zavedení nových parametrů.

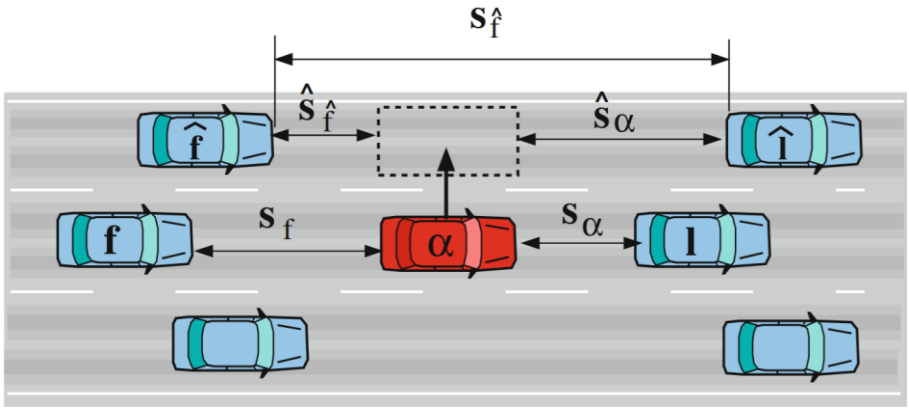
Obecně se tak každý prvek, který byl zahrnut do podélného modelu, automaticky přenáší i do rozhodovací části. Rozdíly rychlostí, reakce na brzdová světla a jiné jsou tak při úvahách změny pruhu zohledněny právě tehdy, pokud jsou přítomny i v samotném akceleračním modelu.

Tento přístup se však potýká s problémy v případech, kdy je důležité zohlednit taktické a kooperativní chování, tj. např. při zařazování do ucpaného cílového pruhu.

2.2.2.1 Obecný užitkově-bezpečnostní rámec

Zpracováno dle zdroje [3]. Za předpokladu, že v daný moment si řidič α vybírá manévr k z diskrétní množiny alternativ K a za předpokladu, že je řidič schopný anticipovat důsledky svého rozhodnutí (tj. rychlosti a mezery ovlivněných vozidel), je možné pro každé ovlivněné vozidlo vypočítat užitek daného manévru pomocí vztahu zrychlení.

V procesu rozhodování pak řidič maximalizuje užitkovou funkci také zvanou motivační kritérium, avšak za předpokladu, že je daná akce bezpečná, k čemuž se využívá bezpečnostní kritérium. Typickým příkladem manévru, pro který je nutno brát zřetel na okolní ovlivněný provoz může být změna jízdního pruhu (viz obrázek 1).



Obrázek 1: Změna jízdního pruhu, zdroj: [3]

Bezpečnostní kritérium

Žádný z řidičů β (včetně rozhodujícího řidiče α) nesmí být rozhodnutím k nucen k vykonání kritického manévru. Kritický manévr je takový manévr, který zahrnuje brzdění přesahující hodnotu bezpečného brzdění b_{safe}:

$$a_{mic}^{(\beta,k)} = -b_{safe}$$

Přičemž hodnota b_{safe} je blízká hodnotě komfortního zpomalení, již zmiňovaného v Gippsově modelu a v IDM. V obrázku 1 pak tedy pro nového následníka řidiče α platí:

$$\hat{a}_{\hat{f}} = a_{mic}(\hat{s}_{\hat{f}}, v_{\hat{f}}, v_{\alpha}) > -b_{safe}$$

Motivační kritérium

Řidič α volí takovou alternativu k', která dosahuje maximálního užitku U:

$$k_{selected} = \arg\max_{k'} U^{(\alpha,k')}$$

Přičemž nejjednodušší motivační kritérium volí přímo vztah zrychlení, motivační kritérium zároveň předpokládá, že rozhodující řidič dospívá k racionálním rozhodnutím ve vlastní prospěch.

Obecný aparátus modeluje řidiče jakožto sobeckého jedince, tj. zřetel na okolní provoz je brán pouze s ohledem na prevenci nehod a zamezení kritickým manévrům. Motivační kritérium pro „egoistického“ řidiče tedy představuje vztah:

$$\hat{a}_{\alpha} - a_{\alpha} > \Delta a + a_{bias}$$

kde

$$a_{\alpha} = a_{mic}(s_{\alpha}, v_{\alpha}, v_{l})$$

$$\hat{a}_\alpha = a_{mic}(\hat{s}_\alpha, v_\alpha, v_l)$$

Přičemž prahová hodnota změny jízdního pruhu Δa zamezuje změně jízdního pruhu v případě, kdy je vnímaná výhoda příliš nízká. Váha a_{bias} vnáší do chování lehkou asymetrii, např. při dálničním provozu by se vozidla měla více řadit do pravého pruhu. Hodnota váhy by neměla být příliš vysoká ($|a_{bias}| \ll b_{safe}$), ale zároveň by měla převyšovat hodnotu Δa.

2.2.2.2 MOBIL – Minimizing Overall Braking Induced by Lane Changes

V překladu model snažící se o minimalizování brzdění způsobeného změnou jízdního pruhu. Výše uvedená kritéria vystihují chování nadměrně sebestředného účastníka provozu. V situacích, kdy je změna pruhu nutná, je toto chování akceptovatelné, pokud je však změna pruhu volitelná, většina řidičů zohlední nejen aspekt prevence nehod, ale také disproporční „znevýhodnění“ ostatních účastníků provozu, zejména pokud je vnímaná výhoda změny pruhu dostatečně malá. Tuto skutečnost je možné reflektovat v motivačním kritériu pomocí parametru p – politeness factor neboli faktor zdvořilosti.

$$\hat{a}_\alpha - a_\alpha + p\left(\hat{a}_{\hat{f}} - a_{\hat{f}} + \hat{a}_f - a_f\right) > \Delta a + a_{bias}$$

V případech, kdy je zdvořilostní faktor daného řidiče nastaven roven 1 (a zároveň prahová hodnota i bias nabývají zanedbatelně malé hodnoty) a pomineme-li bezpečnostní složku modelu, pak je možné říci, že takový řidič by volil změnu jízdního pruhu pouze v situacích, které by vedly k navýšení sumy akcelerací všech ovlivněných vozidel. Aby model lépe odrážel skutečnost, zdvořilostní faktor se doporučuje volit zhruba kolem hodnoty 0,2.

3 Kalibrace a validace simulačního modelu

Kapitola je zpracována dle zdroje [3]. Kalibrace a validace představují klíčové aspekty tvorby realistických dopravních simulací. Tato kapitola popisuje proces kalibrace simulačního modelu společně s jeho následnou validací.

3.1 Kalibrace

Kalibrací simulačního modelu se rozumí proces ladění jeho parametrů tak, aby výsledné simulace co nejlépe odpovídaly reálnému chování dopravního proudu a řidičů. Jde o nezbytný krok, neboť i nejlepší model není možné aplikovat na konkrétní dopravní situaci s výchozími a příliš obecnými hodnotami parametrů. Cílem tedy je nalézt takové nastavení parametrů modelu, které bude co nejefektivněji využívat popisnou sílu modelu za účelem reprodukce lokálního chování řidičů či reprodukce charakteristik kolektivního dopravního proudu. Popisná síla se specifikuje pomocí vhodně zvolené objektivní funkce, která se minimalizuje či maximalizuje dle voleného matematického rámce.

S ohledem na kalibraci modelu se většinou volí jeden ze dvou matematických přístupů: metoda nejmenších čtverců (LSE – Least Squared Errors) nebo metoda maximální věrohodnosti (ML – Maximum Likelihood). V případech, kdy je žádoucí kalibrovat průběžně za běhu simulace nebo existuje nutnost zpracování dat v reálném čase, je možné využít variantu metody maximální věrohodnosti: Kalmanův filtr.

V případě LSE je cílová funkce definována jako součet čtverců rozdílů mezi simulovanými hodnotami a referenčními daty:

$$S(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left(y_i^{sim}(\boldsymbol{\beta}) - y_i^{data}\right)^2$$

Cílem je pak nalezení vektoru parametrů $\boldsymbol{\beta}$, který tuto funkci minimalizuje.

Druhou variantu představuje metoda maximální věrohodnosti, která funguje na pravděpodobnostním přístupu. Předpokládá se přítomnost stochastických složek (např. náhodný šum v akceleraci) a je definována pravděpodobnost, že model s daným vektorem parametrů $\boldsymbol{\beta}$ generuje pozorovaná data:

$$\mathcal{L}(\boldsymbol{\beta}) = \prod_{i=1}^{n} p\left(\boldsymbol{y}_i^{data} | \boldsymbol{\beta}\right)$$

Pro numerickou stabilitu a zjednodušení výpočtu se obvykle využívá logaritmická transformace:

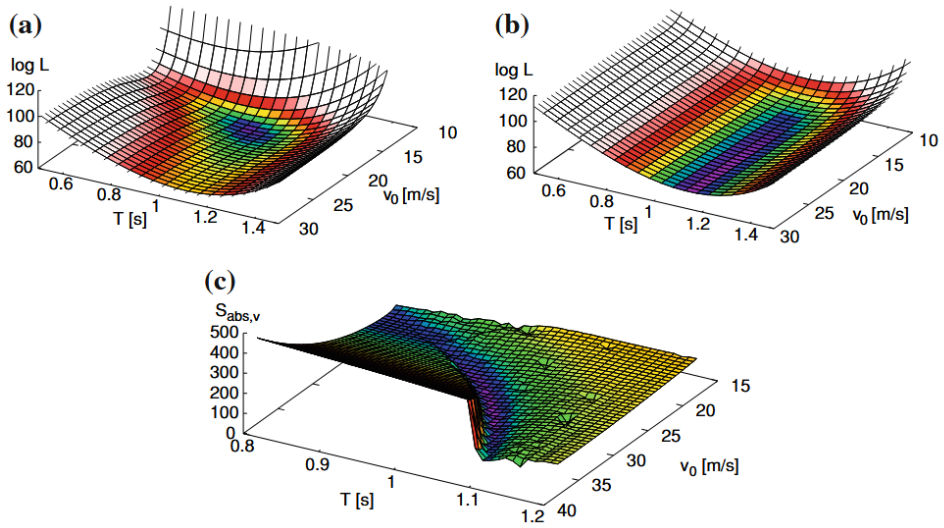
$$\tilde{\mathcal{L}}(\boldsymbol{\beta}) = \sum_{i=1}^{n} \ln p\left(\boldsymbol{y}_i^{data} \middle| \boldsymbol{\beta}\right)$$

Výsledná kalibrace pak hledá maximum logaritmické věrohodnostní funkce. Oba zmíněné přístupy tedy vedou na problém nelineární optimalizace.

3.2 Optimalizační metody

Nelineární optimalizace bývá stěžejním krokem kalibrace, jelikož využívané objektivní funkce často vykazují větší množství lokálních extrémů. Obecně jde o složitý úkon, pro který neexistuje jednotný předepsaný způsob, jak přistupovat ke všem typům funkcí. Funkce je však možné hrubě rozdělit do tří kategorií dle charakteru dat použitých při kalibraci (viz obrázek 2):

- Hladké unimodální funkce – vznikají při kalibraci trajektorií vozidel pomocí technik LSE či ML, zejména pokud jsou kalibrační data kompletní, tj. obsahují všechny dopravní situace. Mají jedno globální optimum.
- Hladké funkce bez jednoznačného optima – vznikají zejména v důsledku chybějících dat.
- Multimodální a kolísavé funkce – typické pro kalibraci na základě agregovaných dat, například z detektorů.



Obrázek 2: Různé typy objektivních funkcí, zdroj: [3]

Zvolené optimalizační metody musí odpovídat typu objektivní funkce. Mezi deterministické metody patří: Newtonova metoda, Gauss-Newton, Gradientní sestup, Levenberg-Marquardt.

Tyto metody však vyžadují hladkou unimodální funkci. Pro optimalizaci multimodálních funkcí je nezbytné využít stochastických metod, jelikož deterministické metody mají tendenci uváznout v lokálních extrémech, ze kterých jsou stochastické metody schopné uniknout. Mezi využívané stochastické metody patří genetické algoritmy případně varianty simulovaného žíhání, specificky metoda křížové entropie – CEM (Cross-Entropy Method).

3.3 Robustnost a vyhodnocení modelu

Kvalitu modelu není možné posuzovat pouze dle hodnoty objektivní funkce, důležitá je také robustnost modelu, tj. jeho odolnost vůči malým změnám hodnot parametrů. Modely o nízké robustnosti nejsou pro účely predikce žádoucí, jelikož i malá změna v hodnotách parametrů může vést k drastickým rozdílům ve výstupech.

Odhad vlivu parametrů na výstupy je předmětem analýzy citlivosti. Kolísají-li výrazně výsledky v závislosti na parametrech, pak model není dostatečně robustní. Mimo toho je žádoucí také tzv. parametrická ortogonalita, každý parametr by měl ideálně upravovat jeden konkrétní aspekt chování řidiče.

V případě, že existují dva modely se srovnatelnou vhodností i robustností, preferuje se model o menším množstvím parametrů, neboli model o vyšší úspornosti. Model s větším množstvím parametrů není automaticky lepším modelem, přestože může být lépe uzpůsoben cílovým datům, ztrácí schopnost generalizace, což je pro účely predikce nežádoucí. Pro kvantifikaci vztahu mezi počtem parametrů a vhodností modelu existují statistické testy jako například F-test či test poměru věrohodností.

Variabilita lidského chování limituje přesnost kalibrace, tato variabilita může nastávat jednak mezi jednotlivými řidiči (interindividuální variabilita) ale také v chování jednoho řidiče v průběhu času (intraindividuální variabilita). Existují modely, které se snaží toto chování vystihnout například pomocí efektu frustrace, kdy řidič navyšuje svoji agresivitu po dlouhém čekání v koloně nebo po neúspěšném pokusu zařadit se do jiného pruhu.

Kalibrační studie však ukazují, že i nejlepší modely dosahují minimálně 20 % chyby. Důležitým faktorem je zmiňovaná variabilita lidského chování, přičemž snaha o vystižení tohoto chování pomocí rozšíření modelu může paradoxně vést ke zhoršení jeho prediktivních schopností, jelikož model bývá v důsledku rozšiřován o dodatečné parametry.

3.4 Validace

Validace představuje nezávislé ověření kalibrovaného modelu na odlišné množině dat, která nebyla využita při procesu kalibrace. Cílem je zjistit, je-li model dostatečně schopen predikovat chování systému v nových situacích. Zatímco kalibrace se zaobírá schopností modelu přizpůsobit se (kalibračním) datům, validace zkoumá jeho predikční sílu. Z tohoto důvodu je směrodatným ukazatelem kvality modelu ne jeho vhodnost, ale zejména jeho predikční síla. [3]

Dobrý ukazatel spolehlivosti modelu představuje jeho robustnost, pokud model vykazuje obdobné výstupy při drobných změnách parametrů, je pravděpodobné, že při zasazení do nového, neznámého prostředí (predikce) bude fungovat dostatečně kvalitně. [3]

Hlavním postupem je rozdělení dat na kalibrační a validační množinu. Po odladění parametrů pomocí kalibračních dat jsou výstupy modelu porovnávány s validačními daty. K vyhodnocení přesnosti se využívají standardní statistické ukazatele jako střední kvadratická chyba, průměrná absolutní chyba, případně korelační koeficient či míra věrohodnosti pro kalibrace ML. [3]

Model s dobrou hodnotou objektivní funkce nemusí nutně poskytovat kvalitní výsledky v predikci, validace pomáhá odhalit případy přeučení (overfitting), kdy model sleduje šum v datech namísto podstatné dynamiky daného systému.

4 Aimsun Next

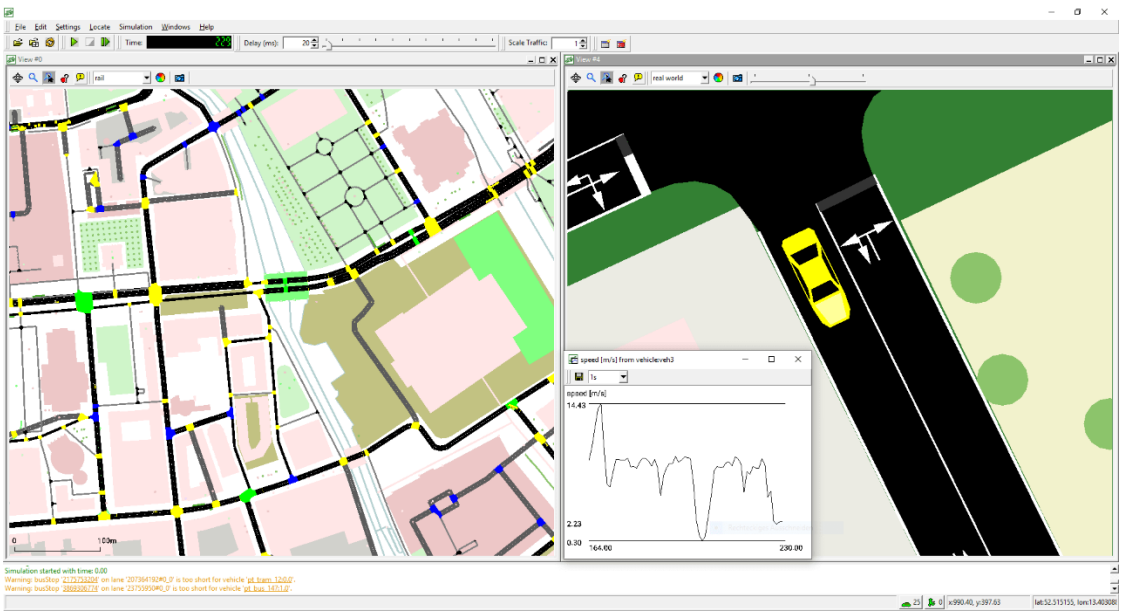
Nástroj Aimsun Next představuje plně integrovanou platformu pro simulaci dopravních sítí o libovolné velikosti a komplexitě. Díky možnosti mikroskopického, mezoskopického a hybridního modelování umožňuje popisovat provoz od vyhrazených autobusových pruhů až po regionální makro-systémy.

4.1 Alternativní nástroje

4.1.1 SUMO

Jde o otevřený mikroskopický dopravní simulátor dodávaný pod licencí Eclipse Public License 2.0, který modeluje každé vozidlo i jeho dynamiku zvlášť. V rámci simulace je v něm možné zahrnout i chodce, cyklisty či hromadnou dopravu, díky čemuž nachází využití pro mono- i multimodální analýzy (rozhraní nástroje viz obrázek 3). [13]

Narozdíl od komerčních nástrojů jako Aimsun či PTV Vissim se nástroj nešíří jakožto jedna monolitická aplikace, nýbrž jakožto modulární balík samostatně spustitelných nástrojů. Pro import a tvorbu sítí je například dostupný nástroj *netconvert*, přičemž je umožněn import z různé řady jiných simulačních nástrojů, včetně nástrojů Aimsun a Vissim. Pro doplnění dané sítě je pak k dispozici grafický nástroj *netedit* a pro vizuální ověření běhu slouží nástroj *sumo-gui*. Sadu nástrojů je tak možné řetězit či skriptovat dle potřeb uživatele. [13]



Obrázek 3: Rozhraní simulačního nástroje SUMO, zdroj: [14]

4.1.2 PTV Vissim

Komerční simulační nástroj vyvíjený skupinou PTV Group, je distribuován jakožto součást sady nástrojů PTV Vision Traffic Suite. V jednom modelu je schopný reprezentovat všechny účastníky provozu, osobní i nákladní vozidla, veřejnou dopravu, cyklisty i chodce, včetně jejich vzájemných interakcí (rozhraní viz obrázek 4). [15]

Otevřená rozhraní včetně generického rozhraní COM (Component Object Model) usnadňují propojení s externími aplikacemi a s dalšími nástroji dodávanými jakožto součást balíku PTV Vision Traffic Suite. [15]

Pro dosažení věrného chování nástroj využívá tři pokročilé modely:

- Psycho-fyzikální model sledování vozidel od prof. R. Wiedemanna, který popisuje čtyři stavové režimy (volná jízda, přibližování, následování a brzdění) a pomocí parametrů je možné jej přizpůsobit lokálním podmínkám.
- Model změny jízdního pruhu, který rozlišuje volitelné a nutné přejíždění mezi pruhy a zohledňuje agresivitu řidičů s ohledem na bezpečnou vzdálenost.
- Model laterálního chování v pruhu, který umožňuje simulovat jízdu vedle sebe či předjíždění v konkrétním pruhu, což je typické například pro cyklostezky či pro heterogenní provoz. [16]

Obrázek 4: Rozhraní nástroje PTV Visim, zdroj: [16]

4.2 Architektura

Architektura Aimsun Next se skládá ze dvou hlavních komponent: kernel (simulační jádro) a UI (user interface – uživatelské rozhraní). V paradigmatu MVC (model-view-controller) je možné simulační jádro chápat jakožto nosiče modelu, zatímco komponenta UI zodpovídá za části controller a view. Toto rozdělení umožňuje nezávisle vyvíjet algoritmickou a vizualizační část projektu a zároveň nástroji dopřává značnou míru rozšiřitelnosti. [18]

Simulační jádro je možné spouštět buď interaktivně s uživatelským rozhraním (pro animaci a vizualizaci) nebo v dávkovém režimu pomocí nástroje *aconsole.* To umožňuje experimenty automatizovat a spouštět bez nutnosti dohledu uživatele. [19]

Aimsun Next také podporuje integraci s různými nástroji využívanými v kontextu dopravního inženýrství. Podporuje například integraci se simulačním nástrojem SCANeR nebo s nástroji dopravního plánování Emme a SATURN. Software také disponuje rozhraním pro integraci s různými adaptivními systémy řízení dopravy a podporuje také integraci s nástrojem SIDRA Intersection, který slouží pro optimalizaci světelné signalizace. [20]

4.3 Datová a souborová struktura

Jeden projekt se v rámci nástroje může skládat z většího množství různých souborů, proto je zaveden koncept projektového adresáře s definovanou strukturou podsložek, díky čemuž je usnadněná organizace všech dat náležících danému projektu. [21]

Při tvorbě nového projektu si uživatel zvolí složku, v níž bude automaticky vytvořen adresář dle jména projektu. V projektovém adresáři je pak vytvořen podadresář Model společně se souborem <*projekt*>.*ang* (dle jména projektu). [21] Soubor s příponou .*ang* reprezentuje hlavní projektový dokument, který je komprimován v proprietárním binárním formátu. [22]

V adresáři Model je dále vytvořena složka Resources, do které jsou ukládány všechny ostatní potřebné soubory, soubory jsou organizovány do složek, které jsou také automaticky tvořeny během práce na modelu. Složka obsahuje následující podsložky:

- Animations – Pro soubory zpětného přehrání (replay) simulace, obsahuje soubory s příponou .*arf* .
- APIs – Obsahuje uživatelem zásobené externí knihovny, umožňuje rozšíření či změnu chování programu v rámci simulace.
- Backgrounds – Vizualizační soubory ve formátu .*cad*.
- Backup – Záložní soubory.

- FZP – Obsahuje informace o vozidlech generovaných v rámci mikroskopické simulace, tyto informace je pak možné exportovat do jiných nástrojů za účelem tvorby kvalitní vizualizace ve 3D. Soubory s připonou *.fzp*.
- Initial States – Umožňuje externě skladovat soubory s příponoou *.aip*, které popisují výchozí stav systému, tento přístup adresuje nedostatek simulace, kdy její počáteční stav neobsahuje vozidla a není tak dostatečně reprezentativní s ohledem na skutečný stav.
- Link Analysis – Soubory s příponou *.ala*, které obsahují analytické informace o směrování řidičů.
- Logs – Logovací soubory generované rozhraním pro integraci s nástroji adaptivního řízení dopravy. Většinou mají připonu *.log* či *.txt*, slouží pro účely ladění.
- Maps – Obrazové či vektorové grafické soubory, které umožňují importovat geometrii dopravní sítě do nástroje Aimsun.
- Matrices – Adresář obsahuje matice OD ve formátu *.txt*.
- Outputs – Adresář pro výstupy simulace.
- Path Assignments – Soubory ve formátu *.apa* popisující, jak byly v simulaci voleny trasy.
- Real Data Sets – Soubory formátu *.csv*, slouží zejména pro úkony kalibrace a validace, obvykle obsahují skutečná měřená data.
- Scripts – Skripty v jazyce python (*.py*).
- SSAM – Výstupní soubory pro nástroj Surrogate Safety Assesment Model, obsahuje poziční data vozidel.
- Traffic Arrivals – Soubory s příponou *.ata*, obsahují seznam všech generovaných vozidel v rámci běhu dynamické simulace. [21]

4.4 Tvorba modelu

V každém projektu v Aimsun Next se vyskytují dva základní stavební kameny. Prvním z nich je samotná základní síť, která obsahuje geometrii infrastruktury se všemi uzly a úseky, k nimž se vážou typy vozidel a účely cest (v kombinaci pak zvané „uživatelská třída“). Druhým klíčovém vstupem je dopravní poptávka, která definuje charakteristiky populace vozidel v rámci simulace. [23]

Na tyto povinné základy je možné vrstvit rozšiřující komponenty, např. řízení světelných křižovatek či plány hromadné dopravy. Pro zjednodušení správy rozličných konfigurací

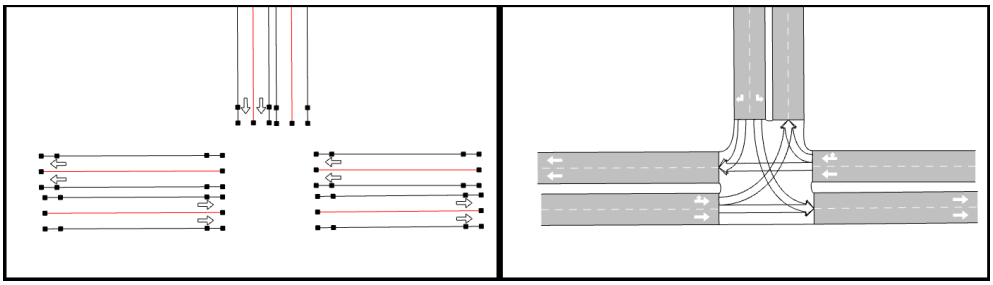
potřebných pro provádění experimentů existuje také infrastruktura pro lokalizované změny v modelu:

- Překrytí atributů umožňuje změnit vlastnosti stávajících objektů, konkrétní scénář tak může upravovat povolené rychlosti, reakční čas řidičů aj.
- Geometrické konfigurace umožňují definovat změnu topologie v porovnání s výchozím modelem, lze tak např. pro konkrétní scénář testovat úpravu křižovatky apod. bez nutnosti redefinice kompletní geometrie sítě. [23]

4.4.1 Dopravní síť

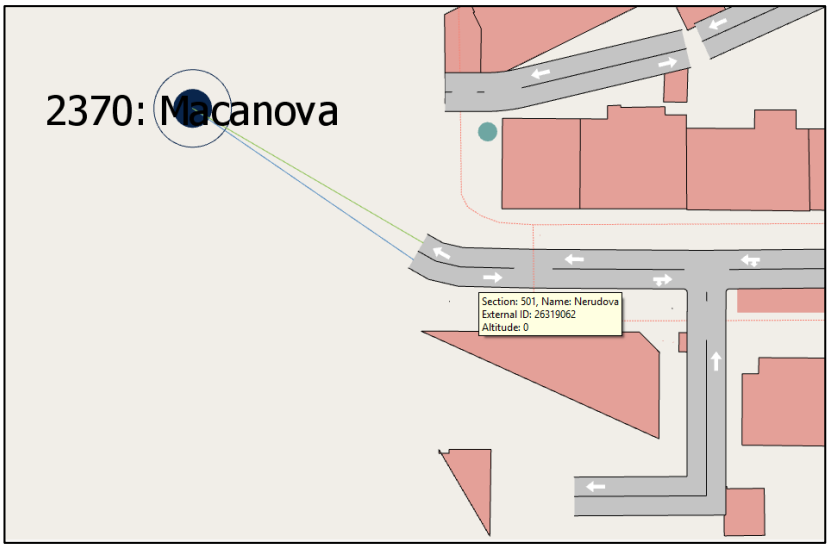
Dopravní síť představuje soubor silničních úseků a uzlů které společně tvoří topologii, má následující komponenty:

- Silniční úseky (sekce) – Představuje souvislou skupinu jízdních pruhů, v nichž se vozidla pohybují jedním směrem. Každá sekce nese atributy (návrhová rychlost, šířka, kapacita aj.) zděděné od výchozího typu cesty avšak otevřené pro případnou modifikaci uživatelem. [24]
- Uzly – Reprezentují spojný bod mezi dvěma a více sekcemi (viz obr. 5), každý uzel má alespoň jednu vstupní a výstupní sekci, přičemž přechod mezi sekcemi je vymezen konkrétní odbočkou. [25]



Obrázek 5: Sekce, uzly a odbočky, zdroj: [25]

- Centroidy – představují počáteční a cílové body dopravní sítě, je možné je propojit jak se sekcemi tak s uzly. Slouží jakožto generátor, modré spojení, dopravního toku do sítě a také jakožto atraktor dopravního toku ze sítě pryč (zelené spojení; viz obr. 6). [26]



Obrázek 6: Centroid napojený na sekci vozovky, zdroj: autor

4.4.2 Dopravní poptávka

Reprezentuje počet vozidel, které projíždějí jednotlivými silničními úseky. Nástroj Aimsun Next umožňuje k definici dopravní poptávky využít (ne však kombinovat) dva přístupy:

- Matice OD (origin-destination) – každá matice nese informace o počtu jízd mezi zdrojovým a cílovým bodem (centroid) pro daný časový interval, typ vozidla a účel cesty. Buňka $m_{i,j}$ matice M tedy obsahuje počet vozidel jedoucí z bodu i do bodu j. [27]
- Objekty dopravního stavu (traffic state) – zde je poptávka definována pomocí vstupních toků pro každý centroid v síti společně s poměry odbočení ve všech uzlech sítě (přičemž obě tyto složky je možno definovat individuálně pro dané typy vozidel). Tuto variantu je však možné aplikovat pouze v simulacích založených na jednotlivých vozidlech, tj. mezo- či mikrosimulacích a pouze s využitím stochastických modelů volby trasy. [28]

Sada těchto objektů poté definuje objekt dopravní poptávky, přičemž každý scénář má přiřazen jeden konkrétní objekt dopravní poptávky. Jednu definici poptávky je možno přiřadit vícero scénářům bez nutnosti individuální úpravy. [29]

4.5 Modelování pohybu vozidel

Věrohodnost a kvalita simulace silně závisí na použitých modelových mechanismech, které popisují chování jednotlivých vozidel. Nástroj Aimsun Next implementuje modely běžně známé z literatury společně s vlastními rozšířeními.

4.5.1 Příjezd vozidla

Časový interval mezi příjezdy po sobě jdoucích vozidel je vzorkován z náhodného rozdělení. Při načítání dopravní poptávky do simulace mohou pro generaci rozestupů být použitý různé modely: exponenciální, rovnoměrný, normální, externí aj. Přičemž výchozím modelem je model exponenciální. [30]

Při rozhodování, zdali má vozidlo dostatek místa pro vstup do sítě jsou zohledňovány:

- parametry vstupního úseku (rychlostní limit),
- parametry vedoucího vozidla (pozice, rychlost, délka, brzdná dráha),
- parametry vstupujícího vozidla (maximální rychlost, ochota překračovat limity aj.)

Vozidlo do sítě může vstoupit pouze tehdy, pokud je dostupná mezera větší nebo rovna minimální požadované vzdálenosti. [31] V případě, kdy je pro vozidlo naplánován vstup do sítě, ale není pro něj dostatek místa, je vozidlo zařazeno do virtuální fronty pro daný vstupní bod. [31]

4.5.2 Modely chování řidiče

Jakožto základ modelu sledu vozidel byl v rámci nástroje Aimsun Next využit model Gippsův, který je dále rozšířen o řadu lokálních parametrů reflektující různé typy řidičů a vozidel, geometrii vozovky, vliv vozidel v sousedních jízdních pruzích a další faktory. [31]

Model byl dále upraven pro věrnější chování v kolonách na dálnici, původní model měl tendenci nadhodnocovat rychlost. Aimsun využívá upravenou verzi, která lépe vystihuje závislost rychlosti na hustotě provozu, používá upravený vztah pro mezeru mezi vozidly a rychlost, který je v původním modelu lineární. [31]

Příčné chování (předjíždění, zařazování) je řešeno diskrétním rozhodovacím modelem, který je také založen na práci Gippse, konceptu potřebnosti a možnosti změny jízdního pruhu. V rámci modelu jsou uplatněny také kooperativní prvky, kdy vozidla v cílovém pruhu mohou mírně zpomalovat, aby umožnila zařazení. [31]

Zajímavostí nástroje je tzv. Two-Lane Car-Following model (dvoupruhový model sledu vozidel), který explicitně modeluje vzájemnou interakci ve dvou souběžných pruzích. Cílem modelu je zpřesnit predikci rychlosti tím, že vozidla v rychlejším pruhu zohledňují pomalejší provoz v sousedním pruhu (typicky z důvodu náhlého připojení). Chtěná rychlost se upravuje dle průměrné rychlosti několika vozidel sousedního pruhu společně s ohledem na maximální

přípustný rozdíl rychlostí (závisí na geometrii vozovky, odlišné pro situace s připojovacím pruhem). [31]

4.5.3 Model akcelerace

Pro výpočet akcelerace vozidla je v simulátoru použito několik modelů, přičemž všechny modely jsou zpětné orientované, což znamená, že akceleraci odvozují z aktuální rychlosti vozidla. Jednodušší akcelerační modely zohledňují pouze vliv sklonu, zatímco složitější modely akcelerace zohledňují dodatečné parametry, kterými jsou například vlastnosti vozidla, vozovky či řidiče. [31]

Základní model (default slope model) upravuje akceleraci a brzdění podle sklonu avšak nezohledňuje hmotnost vozidla. Model TWOPAS pak přidává vliv výkonu motoru, hmotnosti a odporu vzduchu. Model sjíždění svahu pro těžká vozidla simuluje vliv klesání na rychlost, model je navržen specificky pro těžká vozidla, která při sjíždění využívají nižší převodový stupeň, aby předešla přehřátí brzd. [31]

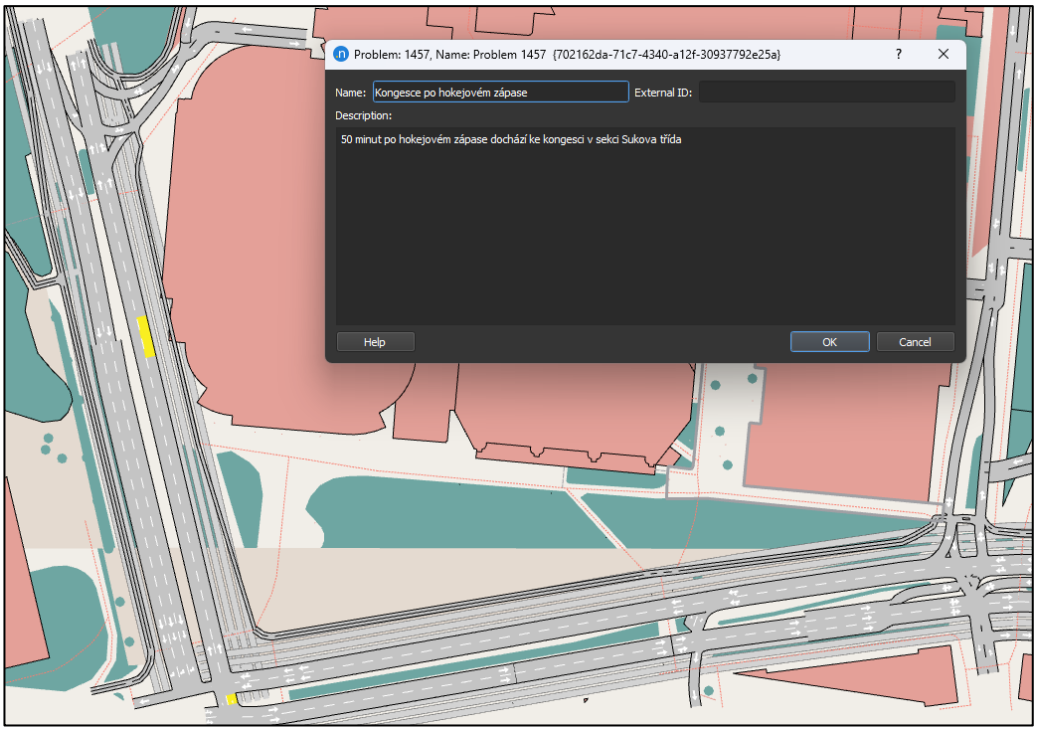
Nejkomplexnější akcelerační model nástroje Aimsun představuje model MFC (Microscopic Free-flow Acceleration), který při výpočtu akcelerace zohledňuje veškeré aspekty provozu vozidla. Model předpokládá volný tok (bez interakce s vedoucím vozidlem) a nehraje v něm roli reakční čas (problematika modelu car-following). Model zohledňuje například: typ vozidla, typ motoru (rozdílné akcelerační profily), hmotnost vozidla, styl jízdy řidiče, typ převodovky, podmínky na vozovce a odporové síly. [31]

4.6 Možnosti řízení dopravy

Zpracováno dle [32]. V rámci simulace založené na vozidlech podporuje nástroj rozličné operace řízení dopravy, které upravují podmínky v síti, ovlivňují chování řidičů či reprezentují dopravní události.

4.6.1 Problémy, strategie, autority, politiky

Problémy slouží jakožto popis nežádoucí dopravní situace, k jejíž řešení budou aplikovány různé strategie, které mohou být v rámci simulace testovány (příklad definice problému v prostředí viz obrázek 7). Objekty typu problém slouží čistě informačnímu a dokumentačnímu účelu a na běh simulace nemají žádný vliv.



Obrázek 7: Definice dopravního problému, zdroj: autor

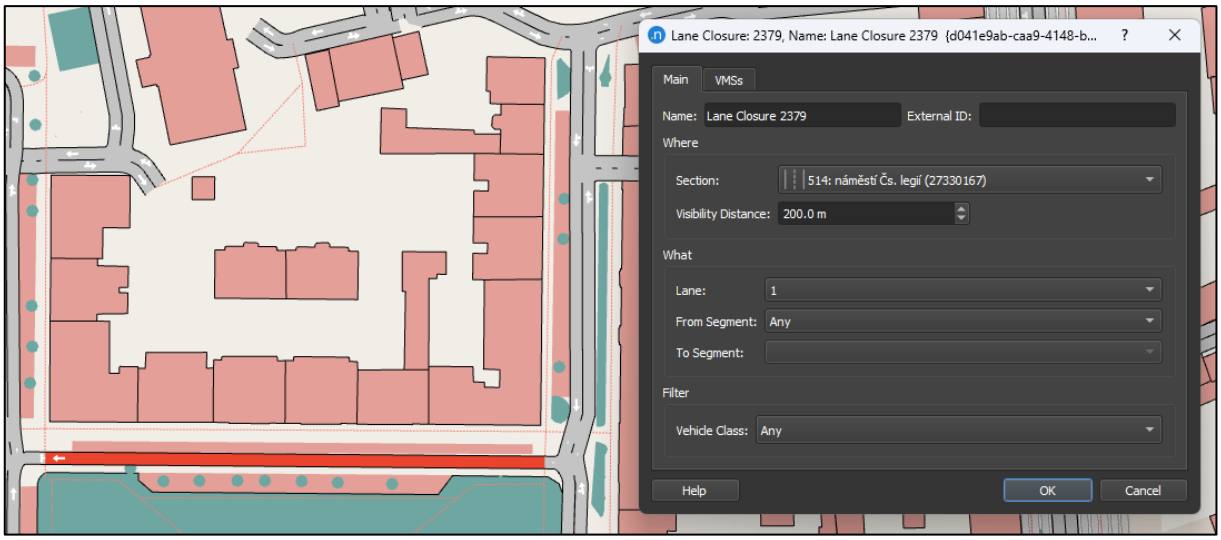
Dopravní strategie je soubor politik, přiřazený konkrétnímu problému, přičemž konkrétní problém spadá pod zodpovědnost dané autority; strategie může obsahovat vícero politik, které mohou spadat pod režii různých dopravních autorit. Autorita je chápána jakožto orgán odpovědný za schválení dané politiky. S ohledem na zmíněné rozdělení je pak možné simulovat i „neideální“ součinnost různých dopravních orgánů což může vyústit v neúplnou implementaci dané strategie (tj. aplikace jen části jejích definovaných politik).

Dopravní politika představuje kolekci akcí, které jsou aktivovány souběžně. Každá akce má pouze jeden důsledek a slouží tak jakožto základní jednotka komplexnějších opatření. Politika může být aktivována a deaktivována na základě času, podmínky (spoušť) či externího zásahu (skrze rozhraní).

4.6.2 Vybrané dopravní akce

4.6.2.1 Uzávěr pruhu

Akce slouží k uzavření jízdního pruhu v rámci konkrétní sekce (definice v prostředí viz obrázek 8). Mezi parametry patří sekce příslušného jízdního pruhu; počet ovlivněných segmentů sekce; třídy vozidel, pro které bude pruh uzavřen, a viditelnost, tj. vzdálenost od které začínají vozidla dbát a reagovat na dané opatření změnou jízdního pruhu.



Obrázek 8: Definice uzávěru pruhu, zdroj: autor

4.6.2.2 Uzávěr odbočky

Tato akce znemožňuje odbočení v daném směru. V případě, že se pro definici dopravní poptávky využívají matice, pak uzávěr může působit na celou odbočku, případně podmnožinu zdrojových pruhů. Reprezentuje-li se poptávka pomocí objektu dopravního stavu, pak se uzavírka musí vztahovat na konkrétní zdrojové pruhy.

4.6.2.3 Změna kooperačního modelu odbočování

Akce umožňuje změnit přednosti v křižovatkách a odbočkách se zaměřením na konkrétní typy vozidel. V případě aplikování tak vozidla s předností v jízdě umožní průjezd vybraným cílovým vozidlům, i když by tak za normálních okolností neučinila. Míra kooperace je nastavitelným parametrem.

4.6.2.4 Rychlostní omezení

Akce umožňuje změnit maximální rychlost na:

- konkrétním úseku,
- všech úsecích daného typu vozovky,
- konkrétní odbočce,
- nebo na skupině objektů.

Omezení je možné aplikovat selektivně s ohledem na typ vozidla.

4.6.2.5 Vynucené odbočení

Akce slouží k vynucení konkrétního odbočení vozidla v závislosti na jejich počátečním a cílovém bodu cesty (při použití matic OD) nebo jejich zamýšleném odbočení (matice

i dopravní stavy). Akci je také možné aplikovat pouze na část vozidel (pomocí definice poměrů) a simulovat tak částečný odklon dopravy.

4.6.2.6 Vynucené přehodnocení cesty

V simulaci využívající matice OD umožňuje u vybraných vozidel (dle typu) přepočítat trasu k cíli na základě aktuální pozice. Ovlivněná vozidla jsou vybírána podle jejich aktuální cesty, kterou definují cílový a počáteční centroid. Nová trasa může být zvolena buď jako nejkratší, nebo pomocí daného modelu diskrétního rozhodování.

4.6.2.7 Změna cíle

Pro simulaci s poptávkou na bázi matic OD umožňuje změnit destinaci daného vozidla. Nová destinace může být konkrétní centroid, případně i jejich seznam společně s pravděpodobnostmi jejich volby.

4.6.2.8 Parkování a přeprava

Pouze pro simulaci využívající poptávku definovanou maticemi OD, umožňuje pro daný vůz změnit cíl na alternativní parkovací plochu, kde cesta daného vozidla končí a zbývající část původní cesty je řešena pomocí hromadné dopravy.

4.6.2.9 Dopravní incident

Způsobí blokaci dané oblasti specifického pruhu sekce vozovky, umožňuje definovat doprovodné rychlostní omezení společně s viditelností incidentu.

4.6.2.10 Periodický dopravní incident

Uvnitř specifikované oblasti náhodně produkuje incidenty s ohledem na specifické hodnoty doby trvání, množství ovlivněných pruhů, periodicity a přijatelné odchylky.

4.6.2.11 Zrušení vyhrazeného pruhu

Pomocí této akce je možné zrušit dosavadní rezervaci vybraného jízdního pruhu a zpřístupnit jej tak pro běžný silniční provoz.

Je nutné zmínit, že aplikovatelnost jednotlivých dopravních akcí se liší podle volené úrovně detailu simulačního modelu (viz tabulka 2).

Tabulka 2: Přehled použitelnosti možných dopravních akcí, zdroj: [32]

Akce	**Mikro**	**Mezo**	**Makro-Mezo**
Uzávěr pruhu	Ano	Ano	Ano
Uzávěr odbočky	Ano	Ano	Ano

Změna kooperačního modelu odbočování	Ano	Ne	Ano
Rychlostní omezení	Ano	Ano	Ne
Vynucené odbočení	Ano	Ano	Ano
Vynucené přehodnocení cesty	Ano	Ano	Ano
Změna cíle	Ano	Ano	Ano
Parkování a přeprava	Ano	Ano	Ne
Dopravní incident	Ano	Zjednodušené	Ano
Periodický dopravní incident	Ano	Zjednodušené	Ano
Zrušení vyhrazeného pruhu	Ano	Ano	Ano

4.7 Modularita simulace

Jakožto komplexní integrovaná platforma podporuje nástroj Aimsun Next různé úrovně detailu dopravní simulace a umožňuje také jejich kombinaci. [33] Pro simulaci vozidel jsou k dispozici čtyři různé varianty úrovně detailu, dle potřeb uživatele:

- **Mikroskopická simulace** – detailní simulace individuálních vozidel v malých časových krocích. Je zohledňována heterogenita vozového parku, pro rozdílná vozidla jsou uplatňovány specifické dynamické charakteristiky. Parametrizována je také variabilita chování řidičů. Mikrosimulátor disponuje také modulem pro pěší dopravu, díky kterému je v rámci simulace možné modelovat i vzájemnou interakci chodců a vozidel.
- **Mezoskopická simulace** – vozidla opět reprezentována jakožto jednotlivé entity, jejich pohyb je však zjednodušen pomocí dělení na úseky, na kterých se vozidla pohybují s předpokládanou průměrnou rychlostí.
- **Mezo-mikro simulace** – forma hybridní simulace, pro klíčové oblasti je aplikován přístup mikrosimulace, nevyznačené oblasti pracují v režimu mezoskopické simulace. Přístup je doporučen zejména v případech, kdy je žádoucí modelovat rozsáhlou dopravní síť, která obsahuje lokality vyžadující detailní rozlišení (např. modelování pěších, řízení křižovatek, uplatnění adaptivních systému řízení dopravy), přičemž celková analýza zůstává zachována na úrovni celé sítě. Využití mezoskopického přístupu v lokalitách nevyžadujících detailní přístup znamená výrazné úspory s ohledem na výpočetní čas simulace.
- **Makro-mezo simulace** – druhá forma hybridní simulace, simulace stále zůstává založena na sledování individuálních vozidel. Do makroskopických úseků jsou vozidla přiřazena prostým navýšením objemu dopravy, v mezoskopických zónách setrvává

detailnější přístup (v porovnání s makroskopickým). Přístup je doporučován v situacích, kdy modelovaná oblast představuje velmi rozsáhlou dopravní síť (například celý region případně i stát), ale zároveň je potřeba v jistých lokalitách ponechat úroveň detailu, kterou makroskopický model nevystihuje. Vhodné případy užití tedy jsou: snaha o snížení výpočetního času, absence detailní geometrie či řídících plánů v některé části modelu. [34]

Scénář je v prostředí primárně vymezen zvoleným způsobem provozu modelu, může jít o makroskopické přiřazování, čtyřstupňový plánovací model, mikroskopickou, mezoskopickou či hybridní simulaci. Po výběru příslušné modelovací metody je pro scénář definována výchozí dopravní síť a konfigurace poptávky. Následně jsou vytvořeny jednotlivé experimenty obsahující konkrétní kombinace testovaných parametrů.

4.8 Organizace simulačních experimentů

Veškeré síťové, poptávkové i řídící vstupy jsou v rámci nástroje ukládány do projektového souboru (.ang). Nad tímto dokumentem je tedy možno stavět vícero simulačních experimentů čehož je zprostředkováno zavedením konceptu scénáře, experimentu a replikace. [35]

4.8.1 Scénář

Objekt scénář stanovuje hlavní vstupy a výstupy simulace, určuje úroveň detailu simulace, nastavuje dopravní poptávku, plán veřejné dopravy, přiřazení tras, řídící plán a případně také soubor reálných dat sloužící pro účely validace. [36]

Může také definovat strategie řízení dopravy či alternativní konfigurace geometrie. Každý scénář vyžaduje určité vstupy, přičemž pro dynamický scénář (tj. fungující na bázi individuálních vozidel) jsou minimální požadované vstupy definice dopravní sítě a definice dopravní poptávky. [36]

4.8.2 Experiment

Představuje nástroj pro detailní analýzu a testování voleného scénáře. Neslouží k modifikaci fyzikálních či infrastrukturálních aspektů simulace, nýbrž k určení specifické kombinace dílčích modelů a algoritmů přiřazení dopravy. Pro jeden scénář může být definováno libovolné množství experimentů. [36]

Metoda přiřazení definuje způsob volby tras a přiřazení dopravy. U dynamických modelů, kde je poptávka zadána maticemi OD a síť nabízí alternativní trasy, hraje nastavení a kalibrace metod volby trasy důležitou roli při validaci modelu. [36]

V rámci dynamického experimentu jsou k dispozici dvě varianty algoritmů dynamického přiřazení provozu (DTA – Dynamic Traffic Assignment):

- SRC (Stochastic Route Choice) – tj. algoritmus stochastické volby trasy, na konci každého odjezdového intervalu, jehož délku volí uživatel, dopočítá aktuálně nejlevnější trasu a pomocí diskrétní volby rozdělí vozidla mezi tuto trasu a nejlevnější trasy z předchozích intervalů. Trasa je vozidlu přiřazena v momentu kdy započne danou cestu. [37]
- DUE (Dynamic User Equilibrium) – tj. algoritmus dynamické uživatelské rovnováhy. Představuje iterativní proces, snažící se o rovnováhu a minimalizaci dob jízdy vozidel pro každý pár původ-cíl a každý odjezdový interval. [37]

Různé algoritmy DTA odrážejí v modelu rozdílné typy chování řidiče. Algoritmus DUE reprezentuje řidiče, kteří trasu volí dle ustálených zkušeností a historických znalostí provozu, zatímco SRC modeluje řidiče s přístupem k dopravním informacím jak před výjezdem, tak i v průběhu jízdy. [37]

4.8.3 Replikace a výsledek

Pro dynamický experiment je dále vyžadována definice replikace či výsledku. Jde o objekt reprezentující jeden konkrétní běh daného experimentu, reprezentuje nosiče výstupních informací experimentu společně s inicializačními hodnotami generátorů náhodných čísel. [36]

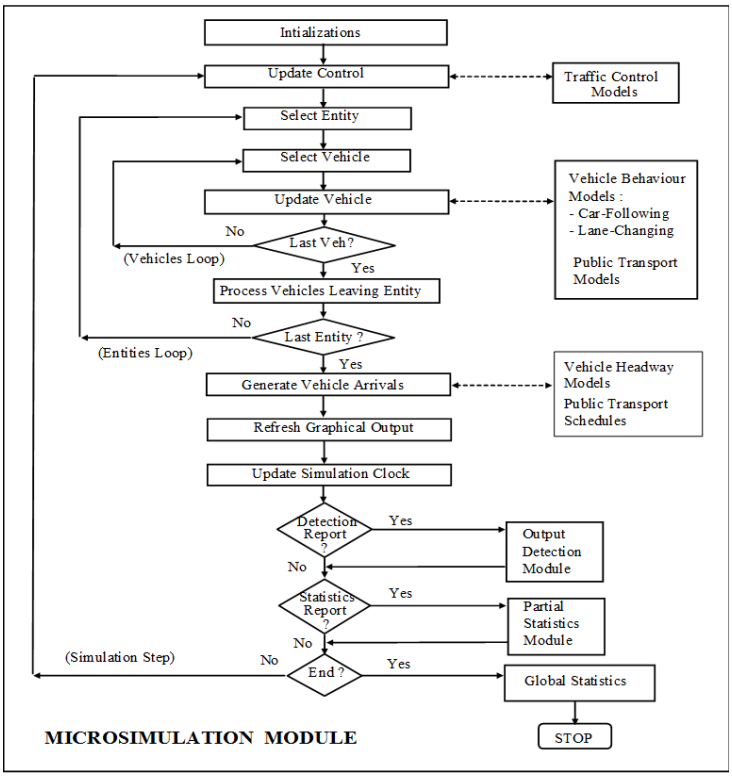
V případě využití algoritmu SRC jde konkrétně o objekt replikace, přičemž každá replikace využívá jedinečnou inicializační hodnotu (seed – semínko) generátorů náhodných čísel, sloužící k reprezentaci variability stochastických jevů. [36]

Pokud je využit algoritmus DUE, pak je výstup experimentu zapouzdřen do objektu „výsledek“, který také pro reprezentaci variability využívá náhodnou inicializační hodnotu. [36] Volba metodiky DUE také umožňuje postupné vybudování výsledku pomocí několika iterací s rostoucím zastoupením definované poptávky. To umožňuje zúčelovat prvotní iterace pro nalezení tras v nepříliš zatížené síti a teprve s rostoucí poptávkou (tj. v dalších iteracích) trasy pozvolna přizpůsobovat dopravním podmínkám. Cílem je vyvarovat se generaci tras, které jsou zkreslené přetížením dopravní sítě a vylepšit tak proces konvergence. [38]

4.9 Proces mikrosimulace

Zpracováno dle [39]. Mikrosimulátor funguje jakožto hybridní proces kombinující princip plánování událostí s průběžným skenováním aktivit (viz obrázek 9). V každém časovém

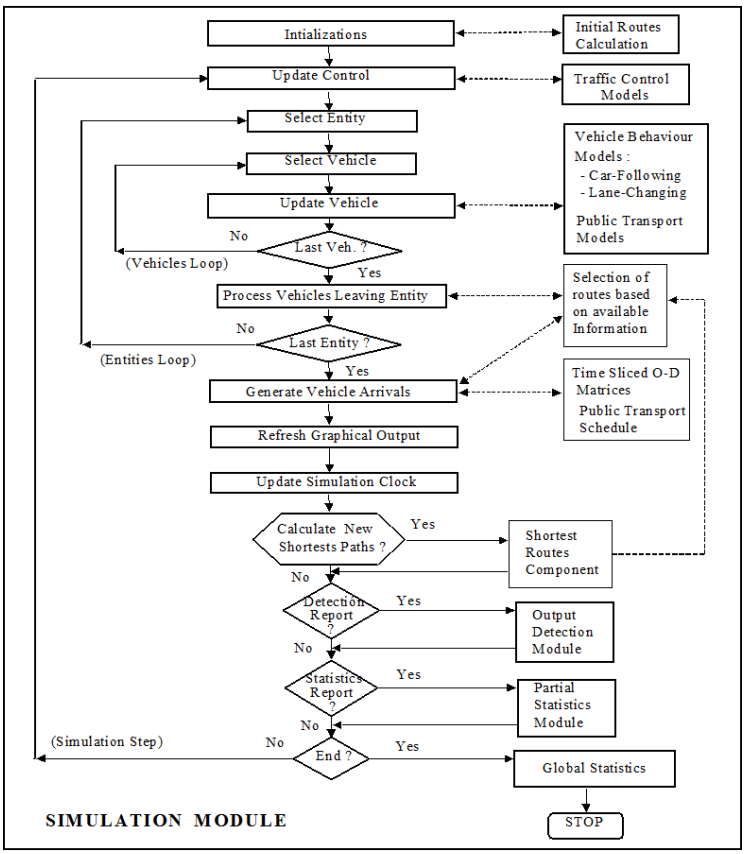
intervalu (simulační krok) je aktualizován seznam nepodmíněných událostí (události nepodmíněné dokončením jiných, např. změna světelné signalizace). Po aktualizaci seznamu následuje smyčka procházející všechny entity modelu (silniční sekce a úseky na úrovni pruhů) a u každého vozidla je aktualizován jeho stav. Po zpracování poslední entity provádí simulátor zbývající operace simulačního kroku: vkládání nových vozidel, kolekce dat a statistik aj.



Obrázek 9: Diagram procesu mikrosimulace, zdroj: [39]

Obrázek 10 pak popisuje proces mikrosimulace v případě, kdy je pro definici poptávky využito matic OD. V inicializačním kroku je pak obsažen také výpočet výchozích tras. Komponenta nejkratších cest následně periodicky přepočítává nejkratší cesty na základě aktuálních cestovních časů, nově generovaným vozidlům jsou pak za využití modelu přiřazení dopravy přidělené specifické trasy. Vozidla si zpravidla ponechávají přidělenou cestu, výjimkou jsou následující situace:

- vozidlo je identifikováno jakožto schopné přepočtu aktuální trasy,
- vozidlo minulo odbočku,
- vozidlo bylo odkloněno akcí řízení dopravy.



Obrázek 10: Diagram simulačního procesu s využitím matic OD, zdroj: [39]

5 Aimsun Next API

Rozhraní pro programování aplikace (API – Application Programming Interface) představuje v prostředí Aimsun Next most mezi dynamickou simulací a externí řídící aplikací. Přístup k tomuto rozhraní vyžaduje samostatnou licenci, bez které nejsou funkce API přístupné. [40]

Motivací pro vznik rozhraní je skutečnost, že dopravní telematika je obor, který se vyvíjí velmi dynamicky a nedrží se jednotných komunikačních standardů. Existuje velké množství komunikačních protokolů, přičemž málo z nich disponuje interoperabilitou napříč aplikacemi. Což znesnadňuje jejich integraci v prostředí mikroskopického simulátoru. [40]

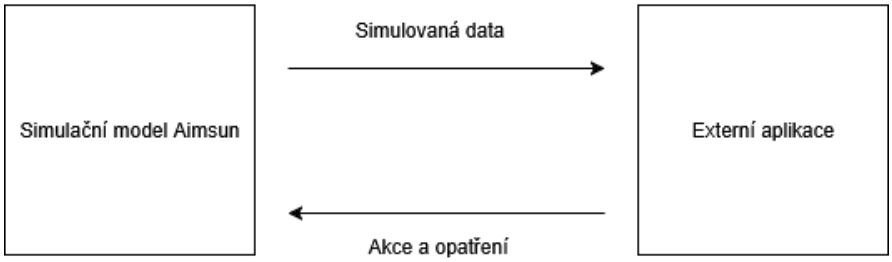
Vestavění řídícího algoritmu přímo do prostředí simulátoru je však jen krátkodobým řešením, jelikož s vývojem nových přístupů a modifikací stávajících by se údržba takto vestavěných systému stala neproveditelná. [40]

Typický provozní scénář moderních řídících systémů funguje na principu osazení dopravní sítě řadou rozličných detektorů, které v pravidelných intervalech poskytují (dle chtěné míry) agregovaná dopravní data, která jsou v lokálním řadiči či dispečerském centru využívány za účelem řízení dopravy pomocí specifických algoritmů. [40]

Aby bylo v prostředí Aimsun možné takovéto systémy vyhodnocovat a ověřovat, je nutné, aby byl mikrosimulátor do modelu schopen začlenit daná zařízení a detektory. Musí tedy emulovat jejich funkcionalitu, tj. nabízet rozhraní, pomocí kterého si externí aplikace mohou vyžádat data a zároveň do modelu vracet povely. Právě tuto funkci zastupuje Aimsun Next API, které mikrosimulátor otevírá externím aplikacím. [40]

5.1 Architektura rozhraní

Na řídící algoritmus telematického systému je pohlíženo jakožto na externí aplikaci, která je na simulaci napojena. Simulátor prostřednictvím rozhraní průběžně poskytuje údaje o stavu sítě, externí modul tato data zpracuje vlastním výpočetním jádrem, vyhodnotí dopravní situaci a do modelu může vrátit vhodné zásahy či opatření (viz obrázek 11). [41]



Obrázek 11: Architektura rozhraní, zdroj: [41]

Rozhraní zároveň umožňuje detailní přístup k datům o jednotlivých vozidlech, toho lze využít například jako vstup pro externí modely spotřeby paliva a produkce emisí nebo ke sledování vybraného vozidla napříč sítí pomocní externího sledovacího systému. [41]

Rozhraní obsahuje mikroskopické i mezoskopické funkce, s ohledem na zaměření práce bude popsáno výhradně mikrosimulační rozhraní.

5.2 Mikrosimulační rozhraní

Zpracováno dle zdroje [41]. Mikrosimulační rozhraní disponuje šesti hlavními funkcemi, skrze které probíhá komunikace mezi modulem rozhraní a simulačním modelem, jedná se o funkce: *AAPILoad*, *AAPIInit*, *AAPISimulationReady*, *AAPIManage*, *AAPIPostManage*, *AAPIFinish* a *AAPIUnLoad*.

Dalších šest funkcí slouží pro zpracování konkrétních událostí, specificky jde o funkce: *AAPIEnterVehicle*, *AAPIExitVehicle*, *AAPIEnterVehicleSection*, *AAPIExitVehicleSection*, *AAPIPreRouteChoiceCalculation*, *AAPIVehicleStartParking*.

5.2.1 Inicializační funkce

- **AAPILoad –** Volána v momentě, kdy je externí modul načten nástrojem.
- **AAPIInit –** Volána v momentě, kdy v nástroji začíná simulace, může být využito pro inicializaci modulu.
- **AAPISimulationReady –** Volána, jakmile Aimsun dokončí incializaci simulace a vozidla jsou připravena k pohybu.

5.2.2 Funkce s časovým krokem

V rámci každého simulačního kroku jsou volány dvě funkce. Funkce *AAPIManage* je volána předtím, než simulátor vykoná jakoukoliv akci, následně simulátor provádí výpočty, aktualizuje vozidla a světelnou signalizaci. Na konci časového kroku je volána funkce *AAPIPostManage.* Z toho vyplývá, že akce provedené v rámci funkce *AAPIManage* jsou zohledněny ve stávajícím

simulačním kroku, zatímco akce provedené v rámci funkce *AAPIPostManage* jsou zohledněna až v kroku následujícím.

5.2.2.1 Parametry funkcí s časovým krokem

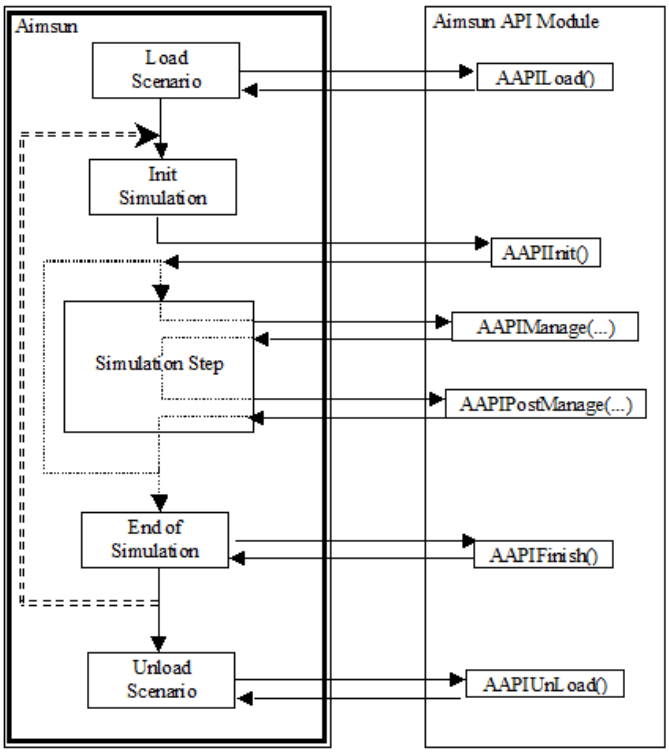
Obě krokové simulační funkce disponují stejnými čtyřmi parametry, které nesou časové informace:

- *time* – absolutní čas simulace v sekundách, na začátku simulace o hodnotě 0 (zahrnuje i čas „zahřátí“ simulace, tj. počáteční běh kdy je cílem simulaci populovat entitami aby reflektovala reálný systém.
- *timeSta* – stacionární simulační čas (tj. od začátku simulace) udáván v sekundách od půlnoci, během zahřívání simulace je jeho hodnota neplatná.
- *timeTrans* – Doba trvání zahřívaní simulace v sekundách.
- *cycle*– délka simulačního kroku v sekundách.

5.2.3 Ukončovací funkce

- **AAPIFinish –** Volána po ukončení simulace.
- **AAPIUnLoad** – Volána po uvolnění modulu rozhraní nástrojem Aimsun.

Podrobný popis interakce simulátoru a externího modulu je viditelný na obrázku 12.



Obrázek 12: Diagram popisující interakci simulátoru a modulu rozhraní, zdroj: [41]

5.2.4 Funkce událostí

- ***AAPIEnterVehicle*** – Pro vstup nového vozidla do sítě (tj. při skutečném vstupu na vozovku, ne při umístění do virtuální fronty).
- ***AAPIExitVehicle*** – Pro situaci kdy vozidlo opouští dopravní síť.
- ***AAPIEnterVehicleSection*** – Při vstupu vozidla do nové sekce dopravní sítě.
- ***AAPIExitVehicleSection –*** Vozidlo opouští konkrétní sekci vozovky.
- ***AAPIEnterPedestrian*** – Pro situace kdy do simulace vstupuje nový chodec.
- ***AAPIExitPedestrian –*** Chodec opouští dopravní síť.
- ***AAPIPreRouteChoiceCalculation*** – Voláno před samotným začátkem výpočtu volby tras.
- ***AAPIVehicleStartParking –*** Voláno při započetí parkovacího manévru vozidlem.

5.2.5 Vybrané interakční funkce

Hlavním přínosem API je možnost aktivně ovlivňovat průběh simulace na základě externí logiky. Za tímto účelem zpřístupňuje rozhraní celou řadu funkcí, od možnosti ovlivňovat chování jednotlivých vozidel, až po aktivaci (či deaktivaci) již zmiňovaných komplexních dopravních politik. Následuje krátký popis vybraných kategorií funkcí mikroskopického API.

5.2.5.1 Detektorová měření

Funkce zpřístupňují v mikroskopickém API kompletní rozhraní pro práci s virtuálními detektory dopravní sítě. Umožňuje vyhledat konkrétní detektor a přečíst jeho vlastnosti (umístění, rozsah pokrytí, podporované veličiny aj.). Vlastní měření je pak rozlišeno na instantní detekci (tj. data měřená v rámci jednoho detekčního cyklu) a na agregovaná data shromažďována na delší časové ose. [42]

5.2.5.2 Řídící akce

Podrobněji bude přiblíženo v implementační části, obecně však jde skupinu funkcí, které umožňují vyvolávat a rušit dopravní opatření, která byla představena v sekci 4.6.2.

5.2.5.3 Incidenty

Rozhraní zpřístupňuje funkce pro vyvolávání a (selektivní i plošné) rušení incidentů. V rámci API chybí možnost definice periodického incidentu, což je však možné napravit ze strany externí aplikace, která může nést pro implementaci absentující funkcionality vlastní logiku. [43]

5.2.5.4 Hromadná doprava

Umožňuje modulu detailní sledování a ovlivňování provozu všech tranzitních linek v síti. Základem je možnost dotazování na počet a identifikátory linek, úseků a zastávek, dále je možné pro jednotlivé vozy číst a měnit dynamické údaje i statické charakteristiky. Pomocí API je možné do systému vozidla vkládat, modifikovat časy zastávek či případně plně přeplánovat trasu. [44]

5.2.5.5 Dopravní poptávka

Rozhraní umožňuje čtení i modifikaci dopravní poptávky ať už je založena na maticích OD či na objektech dopravního stavu. V obou případech je umožněno vyčítat i upravovat charakteristiky zmíněných objektů. Pro dopravní stav je tedy na bázi časového řezu možné upravovat tok či poměry odbočení, zatímco pro matice OD je umožněno vyčítání i úprava počtu jízd mezi dvojicemi bodů. [45] [46]

5.2.5.6 Chodci

Sada funkcí umožňující generaci a detailní správu chodců. Do modelu je tak možné hromadně vložit libovolný počet chodců, buď po předem zadané trase, nebo automaticky po nejkratší cestě. Pro libovolného chodce lze také vyčítat a upravovat statické parametry. [47]

5.2.5.7 Sledování vozidel

Umožňuje detailní on-line kontrolu a diagnostiku individuálních vozidel, je možné například omezit či vynutit rychlost sledovaného vozidla, přepsat cílový pruh, vnutit odbočení nebo rovnou kompletně změnit zamýšlenou trajektorii vozidla. K dispozici jsou také funkce pro čtení statických charakteristik (rozměry, výkon, emisní kategorie). [48]

Mimo zmíněných kategorií obsahuje dodatečně mikroskopické API funkce pro zapisování do logovací konzole, práci se statistikami, přístup k internímu generátoru náhodných čísel, možnosti měnit rychlost probíhající simulace až po její pozastavení, vyčítání informací o dopravní síti aj.

6 Návrh řešení

Cílem praktické části práce je návrh a implementace externí řídící aplikace, která představí základní možnosti vyvolávání nehod a provedení dopravních opatření pro libovolný mikroskopický simulační model. K realizaci požadavků využívá aplikace zmiňované rozhraní nástroje Aimsun Next.

Aimsun Next API je dostupné ve dvou základních variantách: jako nativní rozhraní v jazyce C++, nebo jako rozhraní pro jazyk Python, vytvořené pomocí nástroje SWIG (Simplified Wrapper and Interface Generator). [49] Obě varianty poskytují plný přístup k funkcionalitám simulačního jádra. Výběr rozhraní tak závisí na charakteru implementované aplikace, zejména s ohledem na výkonnost, prostředí kde má být aplikace nasazená a složitost implementace.

V rámci práce byla zvolena implementace v jazyce Python, a to zejména kvůli charakteru psané aplikace, která se především zaměřuje na vyvolávání nehod a aplikaci základních dopravních opatření. Jelikož jádro aplikace tedy převážně funguje jakožto prostředník přesměrovávající volání do Aimsun API a neprovádí žádné výpočetně náročné operace, nebylo nezbytné zohledňovat výkonnost do míry, která by vyžadovala implementaci v jazyce C++.

Python pro autora představuje efektivnější variantu s ohledem na vývoj a testování – zejména díky širokému množství dostupných knihoven, možnosti snadné integrace s dalšími nástroji a také výrazně zjednodušené přenositelnosti napříč platformami.

6.1 Vstupní data pro aplikaci

Aplikace musí umožňovat základní způsob vyvolávání nehod a dopravních opatření a musí být použitelná pro libovolný mikroskopický model. Za tímto účelem se autor rozhodl pro dvojí přístup k poskytování vstupních dat.

Prvním přístupem je možnost poskytnutí množiny vstupních dat pomocí externích konfiguračních souborů, tj. plánované události, incidenty či jiné akce budou aplikaci poskytnuty v externím souboru, aplikace poté soubor validuje a dle specifikace uživatele naplánuje dané povely, které ve vhodný moment předá simulátoru.

Pro interakci se simulací v reálném čase je poté představeno odlehčené rozhraní REST (Representational State Transfer), které umožňuje předávání povelů do simulátoru způsobem ad-hoc.

6.2 Jádro aplikace

Jádro aplikace musí obsahovat již zmiňované funkce, které bude simulátor volat dle podmínek uvedených na obrázku č. 12. Ve výsledku tak jádro aplikace nepředstavuje samostatně spustitelný program, nýbrž skript, který bude v rámci Aimsun Next spouštěn vestavěným interpretem jazyka Python. [50]

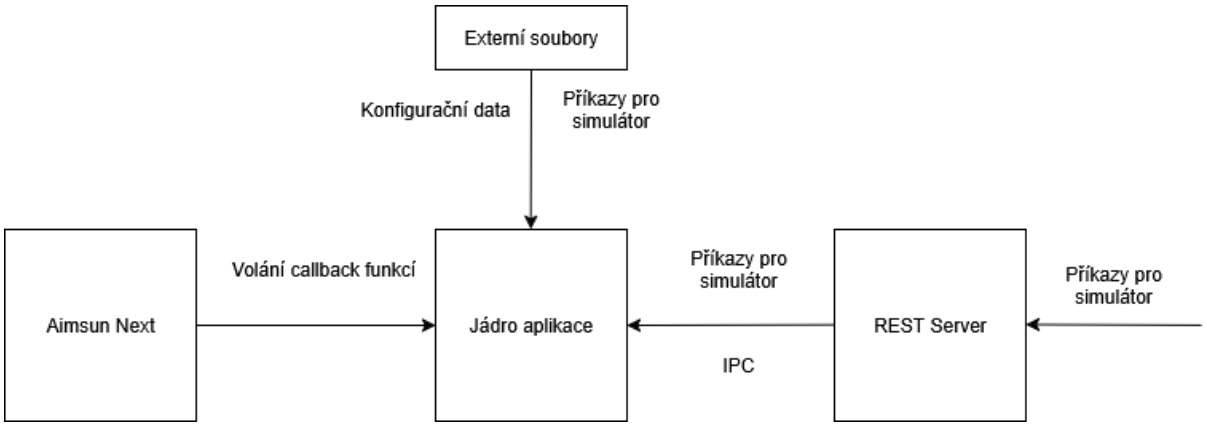
Tato skutečnost představuje jisté komplikace, které je nutné zohlednit:

- Pro implementaci rozhraní REST je nezbytný samostatný proces. Při pozastavení simulace Aimsun drží globální zámek interpretu (GIL) a server realizovaný pouze v jiném vlákně procesu by nereagoval, dokud by nedošlo k opětovnému spuštění simulace.
- Modifikace kódu externí aplikace nebude okamžitě reflektována v jejím chování (tzv. hot-reloading), importované moduly budou stále existovat v prostředí interpretu mezi běhy simulace, v důsledku by tedy bylo nutné nástroj (tj. Aimsun) restartovat.
- Využívá-li aplikace externí knihovny, je nutné, aby Aimsun měl přístup k externímu interpretu, který dané knihovny může používat. [50] V důsledku je tedy nutné jednak na stroj nainstalovat požadovanou verzi jazyka Python (v současné době jde o verzi 3.10) a dále také specifikovat proměnnou prostředí *PYTHONPATH* aby bylo možné dané knihovny využívat.

Zmíněné komplikace je nutné při implementaci brát v potaz. Hlavní odpovědností jádra bude plánování příkazů a jejich předávání simulačnímu jádru nástroje Aimsun.

6.3 Server REST

Jak bylo zmíněno, server musí být v zájmu responzivity provozován v samostatném procesu (díky tomuto přístupu je možné příkazy plánovat i v době, kdy je simulace pozastavena). To s sebou přináší nutnost řešit meziprocesovou komunikaci.



Obrázek 13: Diagram systému, zdroj: autor

Obrázek 13 zachycuje jednoduchou vizualizaci systému, je patrné, že běh navrhované aplikace je ve skutečnosti podmíněn externím simulátorem.

7 Implementace

Následující kapitola popisuje samotnou implementaci externí aplikace k řízení dopravy. V první řadě je uveden krátký popis využitých technologií, následně je popsána struktura projektu společně s podrobným popisem klíčových částí aplikace.

7.1 Použité technologie

V rámci praktické části bylo využito nástroje Aimsun Next 23.0.2 společně s Aimsun Next API, přičemž obě dvě komponenty vyžadují svoji vlastní licenci. Aplikace je implementována v jazyce Python, konkrétně ve verzi 3.10.

7.1.1 Uvicorn

Uvicorn je lehký a výkonný server ASGI (Asynchronous Server Gateway Interface) určený pro běh asynchronních aplikací v jazyce Python. V rámci práce je využit jakožto spouštěč rozhraní REST, postaveného na frameworku FastAPI. [51]

V práci je využit balíček *uvicorn[standard]*, který server doplňuje o (nejen) komponenty implementované v jazyce Cython (nadstavba jazyka Python která je kompilována do nativního kódu v jazyce C), jde zejména o balíček *uvloop* který představuje alternativní implementaci událostní smyčky (event loop) v porovnání ze standardní implementací *asyncio* a o balíček *httptools*, který představuje nízkoúrovňovou knihovnu pro parsování protokolu HTTP (Hypertext Transfer Protocol). [51]

7.1.2 FastAPI

FastAPI je moderní webový framework pro tvorbu rozhraní REST v jazyce Python, je založený na asynchronním standardu ASGI (Asynchronous Server Gateway Interface). Pozitivy jsou důraz na výkonnost, jednoduchost použití, možnosti automatické validace vstupních dat pomocí typových anotací a možnost automatické generaci dokumentace dle specifikace OpenAPI. [52]

7.1.3 Pydantic

Balíček představuje knihovnu pro práci s datovými modely, využívající typové anotace jazyka Python pro validaci a serializaci dat. Umožňuje přesně definovat strukturu očekávaných vstupů a zajišťuje konzistentní chování aplikace při zpracování různých formátů vstupních dat. [53]

7.1.4 Pytest

Balíček Pytest je robustní framework pro psaní a spouštění testů v jazyce Python. Umožňuje jednoduchou definici jednotkových i integračních testů. [54]

7.2 Struktura aplikace

Za účelem větší čitelnosti a udržitelnosti projektu je aplikace členěná do několika balíčků, které mají jasně vymezené odpovědnosti.

Kořenový adresář obsahuje vstupní bod pro simulátor (soubor *aimsun_entrypoint.py*), společně s příkladem konfiguračního souboru (*config.example)*, dále projekt obsahuje seznam externích závislostí pro danou úlohu: *requirements*.in* (test pro testovaní, doc pro tvorbu dokumentace) a konfigurační soubor pro knihovnu Pytest (*pytest.ini)*. V následujících sekcích budou popsány jednotlivé komponenty aplikace.

7.3 Balíček common

Balíček common obsahuje ty části kódu, jež jsou v rámci aplikace sdíleny a využívány na vícero místech.

7.3.1 Datové modely

Modul common.models obsahuje definice datových modelů využívaných napříč aplikací. K definici datových modelů je využito knihovny Pydantic, která umožnila v rámci vývoje relativně přímočarou a snadnou implementaci.

Modul jednak obsahuje výčtové typy *CommandType* a *MeasureType*, které definují typ příkazu či typ dopravního opatření. Následuje definice objektů DTO (Data Transfer Object) pro přenos parametrů jednotlivých příkazů (příklad viz výpis 1).

```
class IncidentRemoveDto(BaseModel):
    section_id: int = Field(...,
                            description="Identifier of the section where
the incident to remove is located")
    lane: int = Field(...,
                      description="Lane where the incident will be
generated")
    position: float = Field(...,
                            description="Position of the incident in the
section (from the beginning of the section).")
```

Výpis 1: Definice parametrů příkazu pro zrušení incidentu, zdroj: autor

Další klíčová definice v rámci modulu je model *CommandBase* (viz výpis 2), který představuje bázový model pro veškeré aplikační příkazy.

```
class CommandBase(BaseModel):
    IMMEDIATE: ClassVar[float] = -1

    command: CommandType
    time: float = Field(default=IMMEDIATE,
                        description="Sim-time in seconds from midnight,"
                        f"omit or set to {IMMEDIATE} to run as soon as
possible")
```

Výpis 2: Definice datového modelu *CommandBase*, zdroj: autor

Každý příkaz pro simulační jádro (resp. příkaz pro aplikaci, která předá parametry příkazu simulačnímu jádru) dědí, resp. rozšiřuje výše uvedený model (většina dodává parametry, některé příkazy však ne). Jak je vidět z ukázky, příkazy, u kterých je položka *time* vynechána jsou chápany jakožto příkazy, které je žádoucí provést ihned, což naznačuje implicitní časová hodnota -1.

Jednotlivé příkazy rozšiřující model *CommandBase* osazují pole *command* konkrétní hodnotou patřící danému příkazu a přidávají pole *payload*, které obsahuje parametry pro daný příkaz (či *None* nepřijímá-li příkaz dodatečné parametry). Příkladem může být příkaz pro tvorbu incidentu (viz výpis 3), který v poli *payload* očekává parametry korespondující s výše uvedeným modelem *IncidentRemoveDto*.

```
class IncidentCreateCmd(CommandBase):
    command: Literal[CommandType.INCIDENT_CREATE] = CommandType.INCIDENT_CREATE
    payload: IncidentCreateDto
```

Výpis 3: Definice příkazu pro tvorbu incidentu, zdroj: autor

Po definici jednotlivých příkazů, které dědí od zmíněného *CommandBase*, je dále využit součtový typ *Command*, který sjednocuje všechny aplikační příkazy pod jeden typ (viz výpis 4).

```
Command = Annotated[
    Union[
        IncidentCreateCmd,
        IncidentRemoveCmd,
        IncidentsClearSectionCmd,
        IncidentsResetCmd,
        MeasureCreateCmd,
        MeasureRemoveCmd,
        MeasuresClearCmd,
        PolicyActivateCmd,
        PolicyDeactivateCmd
    ],
    Field(discriminator="command")]
```

Výpis 4: Definice typu *Command*, zdroj: autor

Pro typový systém knihovny je tak dodána informace, že konkrétní forma příkazu je rozlišitelná podle pole *command*, které nese konkrétní hodnotu výčtu *CommandType*. Na obdobném principu pak fungují parametry pro příkaz tvorby dopravních opatření, kde se sjednocení rozlišuje dle hodnoty výčtu *MeasureType (*pole *type)*.

7.3.2 Konfigurace aplikace

Při inicializaci hledá aplikace ve svém kořenovém adresáři soubor *config* s koncovkami *yaml, yml* či *json*. Je-li takový soubor nalezen, je načten a interpretován jakožto soubor konfigurační. Pokud soubor nalezen není, aplikace pro nastavitelné parametry využívá výchozí hodnoty.

```yaml
api:
  host: 127.0.0.1
  port: 8000
log:
  level: INFO
  ansi: false
  modules:
    aimsun.entrypoint:
      level: DEBUG
      ansi: false
      logfile: entrypoint.log
    server.api:
      ansi: true
schedule_file: schedules/section-497-unreserve-bus-lane.yml
schedule:
  - command: incident_create
    payload:
      section_id: 492
      lane: 1
      position: 10
      length: 25
      ini_time: 60
      duration: 1200
      apply_speed_reduction: true
      max_speed_SR: 30

  - command: measure_create
    time: 240
    payload:
      type: destination_change
      duration: 600
      section_id: 492
      destination_centroid: 501
      new_destination: 502
```

Výpis 5: Vzorový obsah konfiguračního souboru, zdroj: autor

Mezi nastavitelné komponenty patří:

- REST API – umožňuje nastavit port a adresu (host), na které bude API dostupné.

- Logování – umožňuje nastavit výchozí úroveň a cílový soubor pro ukládání logů, a to globálně i samostatně pro jednotlivé komponenty.
- Cesta k interpretu – pro spouštění serverového procesu je nutno vědět, kde se nachází systémový interpret pro Python 3.10. V případě, že uživatel nespecifikuje cestu, prohledává se hodnota proměnné prostředí PATH, není-li nalezen vhodný interpret, REST API je nedostupné. Cesta je nastavitelná pomocí klíče *python_location.*
- Plány – plány příkazů je možné specifikovat buď přímo v rámci konfiguračního souboru nebo pomocí definice jednoho či více souborů které obsahují konkrétní příkazy pro vykonání. V případě definice ve vícero souborech či v kombinaci s přímou definicí v konfiguračním souboru jsou plány sloučeny do jednoho. Přístup je vhodný zejména z důvodu, že globální plán pro celou dopravní síť může být velmi nepřehledný a uživatel tak může zrcadlit princip dopravních problémů z Aimsunu, kde jeden soubor představuje konkrétní problém s definicí incidentů a případných dopravních opatření.

Výpis 5 pak představuje vzorový soubor, ze kterého je možné nahlédnout strukturu konfigurace.

7.3.3 Plánování

Balíček common také obsahuje definici třídy *Schedule,* která slouží jakožto prioritní fronta, do které jsou vkládány plánované příkazy, které jsou seřazeny podle hodnoty pole *time*. Aplikace tak každý simulační krok kontroluje, existují-li ve frontě příkazy připravené k vykonání.

7.4 Rozhraní REST

Balíček server obsahuje samotnou definici API společně s třídou *ServerProcess,* která zapouzdřuje proces spouštění a vypínání externího serverového procesu a zodpovídá za vyhodnocení cesty ke správnému interpretu (proces interpretu je spuštěn a je zkontrolována jeho verze, v případě že vyhovuje je akceptován).

Komunikace mezi jádrem aplikace a serverem je jednosměrná, tzn. server pouze validuje vstupní data. V případě že je příkaz korektní vrací daný endpoint (tj. koncový bod API) statusový kód 202 (accepted – přijato ke zpracování). Příkaz je následně předán do fronty zpráv, ze které bude eventuálně vyzvednut jádrem aplikace. Pokud příkaz obsahuje chybu, endpoint vrací status 422, tj. unprocessable entity (nezpracovatelná entita).

Struktura endpointů v podstatě zrcadlí strukturu mikroskopického API, podporované funkce tedy mají vlastní endpoint, na který je možné zasílat příkazy. Oproti rozhraní nástroje Aimsun

je však jakýkoliv příkaz možné naplánovat na konkrétní simulační čas (stejné platí pro definice v souborech).

7.5 Vstupní bod simulátoru

Představuje jej jediný soubor – *aimsun_entrypoint.py*, tento soubor slouží jakožto vstupní bod simulátoru, resp. obsahuje funkce, které jsou v daných fází běhu simulace volány programem Aimsun.

7.5.1.1 Životní cyklus aplikace

Životní cyklus aplikace se odvíjí od inicializačních funkcí, které byly popsány v sekci 5.2.1. V momentě, kdy je zavolána patřičná funkce, provádí aplikace úkony náležící dané fázi. Jelikož aplikace pro své fungování vyžaduje jen některé z funkcí, následuje jejich zmínka a popis úkonů, které musí aplikace vykonávat.

AAPILoad

Tato funkce je volána v momentě, kdy je externí modul načten do prostředí Aimsun Next, v tomto kroku ještě nejsou přes API k dispozici data o simulaci, jelikož ještě nedošlo k její inicializaci systémem.

Tato fáze je tedy vhodná zejména pro inicializační akce nevyžadující interakci se simulačním jádrem a s danou simulací. Inicializace je v modulu obstarána funkcí *_load*, která zodpovídá za následující:

1. Je-li to nutné, je proveden (re)import daných modulů.

   V práci je využit mechanismus pro importování ostatních komponent do jádra aplikace (za využití balíčku *importlib* ze standardní knihovny jazyka*)*, autor tento způsob volil z důvodu, že změna kódu jiné části aplikace nebývala reflektována, dokud nedošlo k restartování nástroje Aimsun.

   Příčinou je perzistentní prostředí intepretu v rámci běhu aplikace Aimsun Next. Jednou importované moduly přetrvávají v *sys.modules*. Bez dodatečného zásahu se změny projeví až po restartování interpretu (tj. opětovné spuštění Aimsun Next). [55]

   Jako řešení definuje implementovaná aplikace v globální tabulce symbolů strukturu, která k danému souboru se zdrojovým kódem přiřadí položku nesoucí čas modifikace. Pokud je detekováno, že daný modul je starší, nežli nejnovější úprava, dochází k jeho

opětovnému importování. Tímto způsobem je možné do jisté míry reflektovat změny zdrojového kódu bez nutnosti restartování aplikace Aimsun Next (viz výpis 6).

```
Not reimporting module `common.config`, cached
mtime='1754600136.3613095', current='1754600136.3613095' => NO CHANGE

Importing module `server.ipc`, cached mtime='1754697393.11918',
current='1754698358.4610488' => CHANGED
```

Výpis 6: Ukázka z logu pro importování modulů, zdroj: autor

2. Načítání a validace konfiguračních souborů.
3. Inicializace potřených komponent (logger, čítač pro generaci identifikátorů, spuštění serverového procesu)

AAPISimulationReady

```
def AAPISimulationReady() -> int:
    _process_schedule(up_to=0.0)
    return 0
```

Výpis 7: Obsah funkce AAPISimulationReady, zdroj: autor

V této fázi je již simulace inicializována a aplikace tak může zasahovat do jejího průběhu. Jak je vidět z výpisu 7 dochází ke zpracování připravených příkazů, ty mohou být naplánovány buď na začátek simulace (tj. stacionární simulační roven nule) nebo mohou být míněné k okamžitému zpracování, což indikuje hodnota pole *time* rovná -1.

AAPIManage

```
def AAPIManage(time: float, timeSta: float, timTrans: float, acicle:
float) -> int:
    _process_ipc(current_time=timeSta)
    _process_schedule(up_to=timeSta)
    return 0
```

Výpis 8: Obsah funkce AAPIManage, zdroj: autor

Jak je vidět ve výpisu 8, v rámci funkce dochází nejprve ke zpracování povelů předaných serverovým procesem skrze frontu zpráv. Dané zprávy jsou interpretovány (a validovány) jakožto příkazy pro simulátor. Každý příkaz s sebou nese časové razítko, které značí na jaký čas je naplánováno provedení příkazu.

Je třeba zdůraznit, že uvedený čas určuje okamžik, kdy má být aplikací vykonán příslušný příkaz, například předání požadavku na vytvoření dopravního incidentu simulačnímu jádru, přičemž samotný incident může být naplánován na jiný čas. Tento postup byl zvolen, jelikož

většina funkcí API nedovoluje odložit vykonání na konkrétní simulační čas, nýbrž bývá zpracována okamžitě.

Při zpracování příkazů z fronty zpráv mohou nastat dvě varianty:

1. Simulační čas je větší nebo roven časovému razítku příkazu – v tomto momentě dochází k okamžitému vykonání daného příkazu (důsledkem příkazu může mj. být také naplánování dalšího příkazu).
2. Simulační čas je menší než časové razítko příkazu – pro tuto variantu dochází k vložení příkazu do prioritní fronty a jeho obsluha je tedy aplikací odložena.

Po zpracování fronty zpráv dochází ke zpracování plánu (funkce *_process_schedule*), v této fázi dochází k vykonávání plánovaných příkazů, jejichž časové razítko je menší nebo rovno stacionárnímu simulačnímu času (předáno parametrem *up_to*).

```
@register_handler(CommandType.MEASURE_REMOVE)
def _measure_remove(measure: MeasureRemoveDto) -> Result[int]:
    code = AimsunStatus.OK
    try:
        AKIActionRemoveActionByID(measure.id_action)
    except Exception as exc:
        log.exception(exc)
        code = AimsunStatus.API_FAILURE
    return Result.from_aimsun(code,
                              msg_ok=f"Removed measure {measure.id_action}",
                              msg_err=f"Failed to remove measure {measure.id_action}")
```

Výpis 9: Příklad registrace obslužné funkce pro daný příkaz, zdroj: autor

Za vykonání daného příkazu zodpovídá funkce *_execute*, která pro daný typ příkazu dohledá obslužnou rutinu (viz výpis 9), spustí ji, vyzvedne výsledek a informuje uživatele (prostřednictvím výpisu v nástroji Aimsun). Obslužné funkce jsou registrovány pomocí dekorátoru *@register_handler*, konkrétní funkce je dohledána dle typu příkazu.

7.6 Přehled podporovaných funkcí mikroskopického API

V této sekci je uveden přehled aplikací podporovaných funkcí mikroskopického API. Parametry funkcí simulačního jádra a parametry příkazů aplikaci na sebe nepasují 1:1, k aplikaci je však zpřístupněna dokumentace, díky které uživatel ví, jaké příkazy jsou v aplikaci podporované.

7.6.1 Incidenty

Pro tvorbu a rušení incidentů je v rámci aplikace implementováno pět funkcí mikroskopického API: *AKIGenerateIncident, AKIGenerateIncidentDistancePerVehType, AKIRemoveIncident, AKIRemoveAllIncidentsInSection, AKIResetAllIncidents.* Korespondující aplikační příkazy společně s popisem chování jsou uvedené v tabulce 3.

Tabulka 3: Popis aplikačních příkazů pro správu incidentů, zdroj: autor

Aplikační příkaz	**Chování**
incident_create	Tvorba dopravního incidentu ve specifickém pruhu konkrétní sekce vozovky. Umožňuje nastavit rychlostní omezení a viditelnost incidentu pro vozidla (tj. vzdálenost kdy jej začnou brát v potaz).
incident_remove	Odebrání specifického incidentu.
incidents_clear_section	Odebrání všech aktivních incidentů v sekci vozovky.
incidents_reset	Odebrání všech incidentů vytvořených pomocí API. Načte výchozí dopravní incidenty (jsou-li definovány).

7.6.1.1 Tvorba incidentu

Umožňuje jí aplikační příkaz *incident_create,* podle specifikovaných parametrů dochází k volání jedné z funkcí *AKIGenerateIncident* nebo *AKIGenerateIncidentDistancePerVehType.* Jednotlivé parametry jsou viditelné v tabulce 4.

Tabulka 4: Popis parametrů příkazu *incident_create*, zdroj: autor

Parametr	**Popis**
section_id	Identifikátor sekce vozovky
lane	Dopravní pruh dané sekce
position	Pozice incidentu od začátku sekce vozovky
length	Prostorová délka incidentu
ini_time	Stacionární simulační čas počátku incidentu (tj. čas od půlnoci)
duration	Doba trvání incidentu
visibility_distance	Výchozí viditelnost incidentu (tj. z jaké vzdálenosti jej vozidla začnou zohledňovat ve svém chování)
per_veh_visibility	Umožňuje asociovat viditelnost incidentu s konkrétním typem vozidla
update_id_group	Jde-li o nový incident, nebo je-li rozšiřován naposled generovaný incident.
apply_speed_reduction	Má-li se aplikovat rychlostní omezení.

upstream_distance_SR	Vzdálenost účinku omezení směrem k incidentu
downstream_distance_SR	Vzdálenost účinku omezení směrem od incidentu
max_speed_SR	Cílová rychlost pro omezení

Podle toho, zdali je definováno pole *per_veh_visibility* pak dochází k volání korespondující funkce API.

7.6.1.2 Zrušení incidentu

Tuto funkcionalitu obstarávají tři aplikační příkazy. Příkaz *incidents_reset* nepřijímá žádné dodatečné parametry a není nutné jej blíže popisovat. Příkaz *incidents_clear_section* přijímá pouze parametr identifikátoru sekce vozovky na které má dojít k odebrání všech aktivních incidentů. Tabulka 5 pak popisuje parametry příkazu *incident_remove*, který slouží k odstranění konkrétního dopravního incidentu.

Tabulka 5: Parametry příkazu *incident_remove*, zdroj: autor

Parametr	**Popis**
section_id	Identifikátor sekce vozovky
lane	Dopravní pruh sekce vozovky
position	Pozice incidentu od začátku sekce

7.6.2 Dopravní opatření

Na základní úrovni definuje aplikace pro interakci s dopravními opatřeními tři příkazy: *measure_create, measure_remove a measures_reset*. Příkaz *measure_create* slouží pro tvorbu dopravních opatření, přičemž parametry (resp. *payload)* definují o jaký typ dopravního opatření se jedná a specifikují jeho charakteristiky. Příkaz *measure_remove* slouží pro odstranění konkrétního dopravního opatření a příkaz *measures_reset* slouží pro deaktivaci všech aktivních opatření.

V rámci interakce s mikroskopickým API v jazyce Python je nutné pro dopravní opatření předem specifikovat identifikátor, se kterým je dané opatření vytvořeno. Identifikátor opatření (resp. identifikátor dopravní akce dle terminologie API) je poté nutné předat funkci ke zrušení příkazu.

Aplikačně je tato skutečnost řešena dvojím způsobem: uživatel buď může zadat předem definovaný identifikátor, nebo je identifikátor automaticky generován aplikací a uživatel je informován o výsledném identifikátoru.

```python
@register_handler(CommandType.MEASURE_CREATE)
def _measure_create(payload: MeasureCreateDto, starts_at: float) -> Result[int]:
    m = payload.root
    result = _apply_measure(m)
    if (result.is_ok() and m.duration and _SCHEDULE is not None):
        action_id = result.unwrap()
        ends_at = starts_at + m.duration
        _SCHEDULE.push(
            MeasureRemoveCmd(
                time=ends_at,
                payload=MeasureRemoveDto(id_action=action_id)))
        log.debug("Auto-scheduled MEASURE_REMOVE id=%s at t=%.1f s", action_id, ends_at)

    return result
```

Výpis 10: Obsluha tvorby dopravního opatření, zdroj: autor

Jak je vidět z výpisu 10, pro dopravní opatření je možné definovat hodnotu pole *duration,* které označuje dobu trvání daného opatření. V případě, kdy je hodnota pole specifikována, pak aplikace automaticky plánuje příkaz *measure_remove* s identifikátorem daného dopravního opatření na daný simulační čas. Dopravní opatření je tedy aplikací automaticky zrušeno po uplynuté době trvání.

Rozlišení typu dopravního opatření je uvedeno v poli *type*, které nabývá jedné z hodnot výčtového typu *MeasureType*. Typ pole *payload* je poté využit v rámci mechanismu *singledispatch* z balíčku *functools* (součást standardní knihovny jazyka), díky kterému je možné snadno vyvolat korektní obslužnou funkci pro dané opatření.

Konkrétní popis parametrů pro jednotlivá dopravní opatření je obsažen v dokumentaci, nicméně je žádoucí uvést výčet podporovaných dopravních akcí, které aplikace umožňuje vykonávat (viz tabulka 6).

Tabulka 6: Popis dopravních akcí podporovaných aplikací, zdroj: autor

Identifikátor opatření	**Korespondující dopravní akce**
speed_section	Změna maximální rychlosti v daných sekcích vozovky
speed_detailed	Změna maximální rychlosti v daných segmentech sekcí vozovky
lane_closure	Uzavření jízdního pruhu
lane_closure_detailed	Uzavření jízdního pruhu (nastavitelná vzdálenost a nastavitelné aplikování two-lane car-following modelu)
turn_close	Uzavření dané odbočky

turn_force_od	Vynucení odbočení (pro poptávku na bázi matic OD)
turn_force_result	Vynucení odbočení (pro poptávku na bázi dopravních stavů)
destination_change	Změna cíle vozidel

7.6.3 Dopravní politika

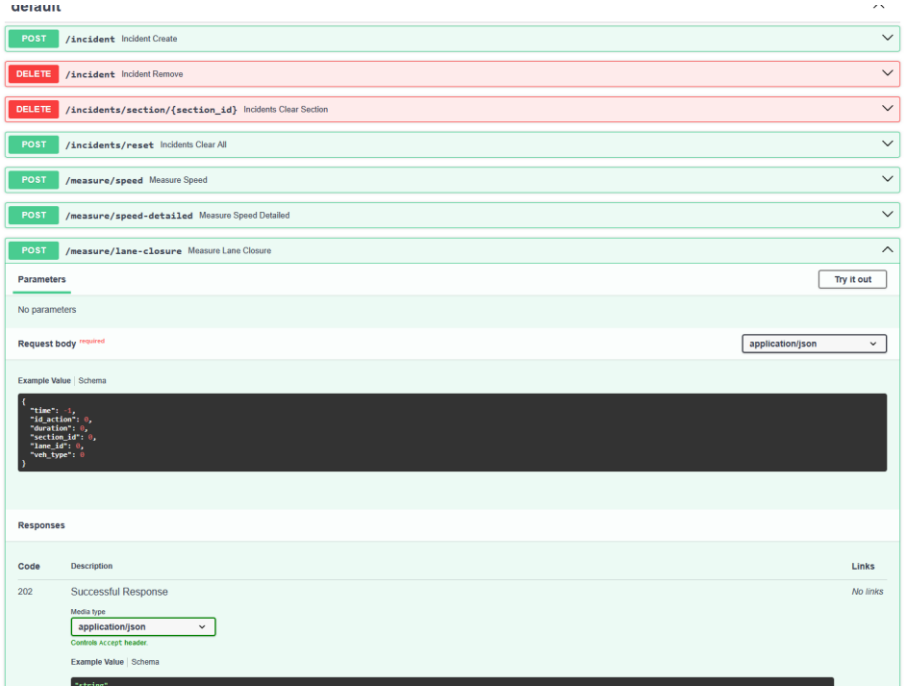
```
@register_handler(CommandType.POLICY_ACTIVATE)
def _policy_activate(payload: PolicyTargetDto):
    try:
        ANGConnActivatePolicy(payload.policy_id)
    except Exception as exc:
        log.exception("Policy activate API failed: %s", exc)
        return Result.err("Policy activate action failed")
    msg = f"Activated policy '{payload.policy_id}'."
    return Result.ok(payload.policy_id, msg)


@register_handler(CommandType.POLICY_DEACTIVATE)
def _policy_deactivate(payload: PolicyTargetDto):
    try:
        ANGConnDeactivatePolicy(payload.policy_id)
    except Exception as exc:
        log.exception("Policy deactivate API failed: %s", exc)
        return Result.err("Policy deactivate action failed")
    msg = f"Deactivated policy '{payload.policy_id}'."
    return Result.ok(payload.policy_id, msg)
```

Výpis 11: Obsluha dopravní politiky, zdroj: autor

Jak je vidět ve výpisu 11, interakce s dopravní politikou, je v rámci mikroskopického API velmi jednoduchá, podle identifikátoru dané dopravní politiky je možné politiku buď aktivovat nebo deaktivovat.

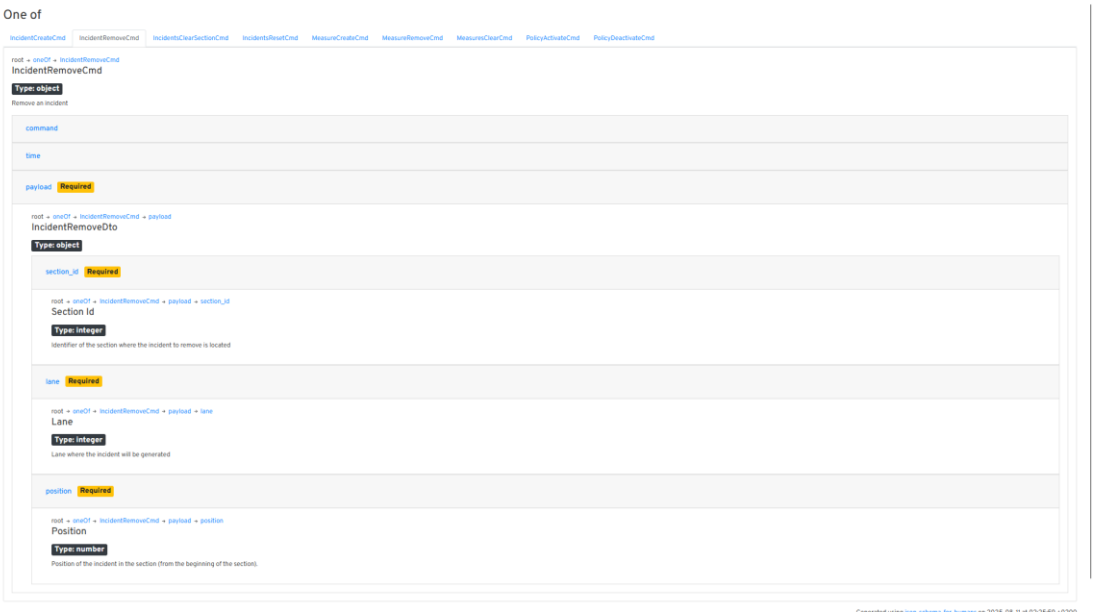
7.7 Dokumentace

Jak bylo zmíněno, aplikace obsahuje automaticky generovanou dokumentaci, a to jak k rozhraní REST, tak k příkazům, které je aplikace schopná zpracovat.



Obrázek 14: Ukázka dokumentace k REST API, zdroj: autor

Dokumentace k REST API je zprostředkována přímo pomocí balíčku FastAPI, který je v kombinaci s knihovnou Pydantic schopen vygenerovat dokumentaci dle specifikace OpenAPI (viz obrázek 14).



Obrázek 15: Ukázka dokumentace aplikačních příkazů, zdroj: autor

Dokumentace k příkazům, které aplikace akceptuje formou konfiguračních souborů je generována pomocí skriptu *doc.py* v adresáři *tools*. Skript nejprve prochází definici typu *Command* a dohledává konkrétní modely specifických příkazů (díky tomu je přidání nového příkazu reflektováno v dokumentaci bez nutnosti upravovat skript). Pro modely konkrétních

aplikačních příkazů je poté generováno schéma ve formátu JSON (Javascript Object Notation) a pomocí knihovny *json-schema-for-humans* jsou schémata převedena na soubory HTML (Hypertext Markup Language), které jsou pro koncového uživatele čitelnější. Příklad generované dokumentace je viditelný na obrázku 15.

Generovaná dokumentace pro aplikační příkazy se ve výchozím nastavení nachází v adresáři *docs*, přičemž skript je možno parametrizovat cestami k výstupním schématům a cestou k výstupní dokumentaci ve formátu HTML.

7.8 Testování

Nezbytnou složku vývoje představuje bezpochyby také testování dané aplikace. Pro testování aplikace bylo využito kombinace automatizovaných testů společně s manuálním testováním. Automatizované testy byly voleny pro jednotky, které je jednoduché (a dle názoru autora smysluplné) testovat v izolaci, konkrétně jde o konfigurační komponenty a o testování korektního fungování REST API.

```
def test_destination_change_bad_percent_sum(self):
        """POST /measure/destination-change with invalid percentage
sums (> 100) returns an HTTPStatus.UNPROCESSABLE_ENTITY CODE."""
        payload = {
            "section_id": 502,
            "new_destinations": [
                {"dest_id": 501, "percentage": 70},
                {"dest_id": 502, "percentage": 50}
            ]
        }
        res = self.client.post("/measure/destination-change",
json=payload)
        self.assertEqual(res.status_code,
HTTPStatus.UNPROCESSABLE_ENTITY)
        self.assertEqual(self._drain_queue(), [])
```

Výpis 12: Příklad jednotkového testu pro REST API, zdroj: autor

Výpis 12 ukazuje příklad jednotkového testu, který konkrétně testuje funkční validaci proporcí nových destinací pro přesměrovaná vozidla. Suma podílů vozidel nesmí překračovat hodnotu 100 %.

Manuální testování bylo nezbytné pro ověření funkčnosti vstupního bodu simulátoru, autor tedy musel každou implementovanou funkci, která volá simulační jádro, ověřit kontrolou přímo v rámci Aimsun Next.

8 Použití aplikace

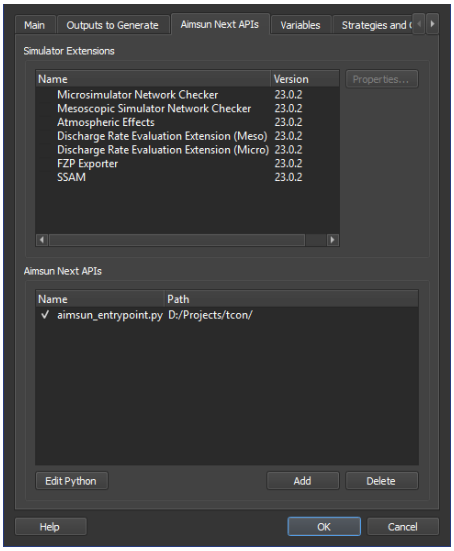
8.1 Prerekvizity

Pro korektní spuštění aplikace je nutné splnit několik prerekvizit:

1. Uživatel musí mít nainstalovanou verzi jazyka Python 3.10 (interpret ideálně dohledatelný v PATH, případně – jak bylo zmíněno – může uživatel poskytnout cestu k interpretu v konfiguračním souboru).
2. Uživatel musí nainstalovat závislosti aplikace. Je umožněno pomocí (generovaného) souboru *requirements.txt*, kde jsou jednotlivé závislosti (včetně jejich verzí) uvedeny. Lze nainstalovat pomocí příkazu: *python -m pip install -r requirements.txt*
3. Aby byly využité závislosti dohledatelné v interpretu prostředí Aimsun Next je nutné nastavit proměnnou prostředí *PYTHONPATH*, která specifikuje dodatečné cesty, kde má interpret hledat moduly pro import.
4. Pro využitelnost aplikace logicky představuje prerekvizitu také vlastnictví platných licencí pro Aimsun Next a Aimsun Next API.

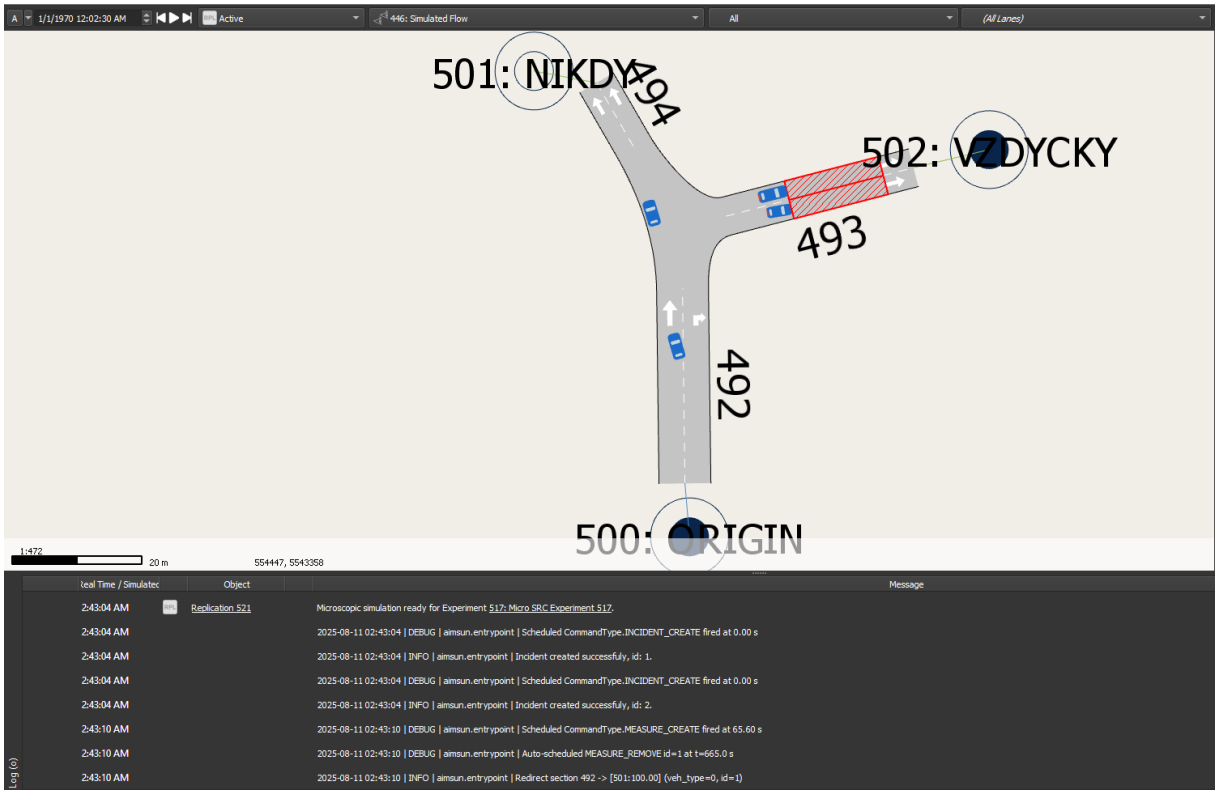
8.2 Integrace s Aimsun Next

Po seznámení s dokumentací je již uživatel připraven aplikaci používat, posledním nutným krokem je tedy napojení aplikace na konkrétní simulační experiment. K tomu stačí v rámci nástroje Aimsun Next zvolit pod položkou *Scenarios* možnost *Properties*, ve vyvolaném dialogovém okně pak zvolit záložku *Aimsun Next APIs* a pod sekcí *Aimsun Next APIs* přidat skript *aimsun_entrypoint.py*.



Obrázek 16: Propojení aplikace a simulačního scénáře, zdroj: autor

Po učinění výše zmíněného (viz obrázek 16) již bude vstupní bod korektně volán simulátorem a pomocí implementované aplikace tak bude pro libovolný mikrosimulační model umožněno řízení dopravního provozu skrz tvorbu incidentů a správu dopravních opatření a politik.



Obrázek 17: Ukázka přesměrování vozidel kvůli dopravnímu incidentu, zdroj: autor

Jak je vidět na obrázku č. 17, aplikace umožňuje vyvolat dopravní incident a v reakci zavést korespondující dopravní opatření, v této situaci jde konkrétně o odklon dopravy, kdy vozidla směřující do destinace 502 byla přesměrována do cíle 501.

ZÁVĚR

Cílem diplomové práce bylo navrhnout a implementovat externí řídící aplikaci, která prostřednictvím Aimsun Next API představí základní způsoby vyvolávání nehod, omezení a zavádění dopravních opatření.

Teoretická část uvedla čtenáře do problematiky simulace se specifickým zaměřením na dopravní simulace, čtenář byl postupně seznámen s různými způsoby, jak strukturovat dopravní simulace dle zkoumaných skutečností. Dále se kapitola detailně zaměřila na oblast mikroskopických dopravních simulací, kde byly podrobně představeny mikrosimulační metodiky, včetně kalibrace a validace dopravního modelu. Autor popsal požadavky na vstupní data pro tvorbu mikrosimulace a dále představil detaily mikrosimulačního jádra – zejména jednotlivé komponenty pohybu vozidel s popisem konkrétních modelů.

Teoretická část pokračovala představením simulačního nástroje Aimsun Next, kde autor nejprve uvedl obecné informace o nástroji a posléze se zaměřil specificky na aspekty řízení dopravy, tj. tvorba dopravních incidentů a nasazení dopravních opatření či politik.

Praktická část obsahuje stručný popis Aimsun Next API pro kontext potřebný v rámci návrhu aplikace. Návrhová část vzpomíná jistá omezení, která s sebou integrace s externím nástrojem přináší a řeší vlastní návrh řídící aplikace. Implementační sekce praktické části pak popisuje samotnou aplikaci z hlediska struktury kódu a zodpovědností jednotlivých modulů. Praktickou část diplomové práce zakončuje kapitola o vlastním použití aplikace uživatelem, popisuje dohledatelnost dokumentace a integraci s nástrojem Aimsun Next.

Vyvinutá aplikace umožňuje základní formu řízení dopravy prostřednictvím vyvolávání dopravních incidentů a nasazování dopravních opatření a politik. Aplikaci by bylo možné rozvíjet několika směry, prvním z nich je určitě tvorba grafické nadstavby, která by uživateli umožnila editovat plánované události pomocí formulářů, předešlo by se tak nutnosti editovat surové konfigurační soubory. Další možností budoucího vývoje je rozšíření škály podporovaných funkcí v rámci aplikace, případně i podpora mezoskopického rozhraní.

POUŽITÁ LITERATURA

[1] KŘIVÝ, Ivan a Evžen KINDLER. Simulace a modelování. Ostrava: Ostravská univerzita, 2001. Učební texty Ostravské univerzity. ISBN 80-7042-809-0.

[2] PURSULA, Matti. Simulation of Traffic Systems - An Overview. *Journal of Geographic Information and Decision Analysis* [online]. 1999, (3), 1-8 [cit. 2025-08-13]. Dostupné z: https://publish.uwo.ca/~jmalczew/gida_5/Pursula/Pursula.html

[3] TREIBER, Martin a Arne KESTING. Traffic Flow Dynamics [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013 [cit. 2025-06-18]. ISBN 978-3-642-32459-8. Dostupné z: doi:10.1007/978-3-642-32460-4

[4] AIMSUN. Simulation Process. In: *Aimsun Next Users Manual* [online]. Barcelona, Spain: Aimsun SLU [cit. 2025-08-12]. Dostupné z: https://docs.aimsun.com/next/23.0.2/UsersManual/MesoDiscreteSimulation.html

[5] WUNDERLICH, Karl, Meenakshy VASUDEVAN, Peiwei WANG (NOBLIS), Richard DOWLING a Vassili ALEXIADIS. Traffic Analysis Toolbox Volume III: Guidelines for Applying Traffic Microsimulation Modeling Software 2019 Update to the 2004 Version: Microsimulation Analysis Planning. In: *U.S. Department of Transportation Federal Highway Administration* [online]. 2019 [cit. 2025-08-13]. Dostupné z: https://ops.fhwa.dot.gov/publications/fhwahop18036/chapter1.htm

[6] Munzilah Md Rohani a Nurul Nasuha Nor Azlan. Overview Of Application Of Traffic Simulation Model. *MATEC Web of Conferences* [online]. 2018, (150) [cit. 2025-08-13]. Dostupné z: doi:10.1051/matecconf/201815003006

[7] ADEBISI, Adekundele. A REVIEW OF THE DIFFERENCE AMONG MACROSCOPIC, MICROSCOPIC AND MESOSCOPIC TRAFFIC MODELS. In: *ResearchGate: Find and share research* [online]. Germany, Berlin: ResearchGate, December 2017 [cit. 2025-08-13]. Dostupné z: doi:10.13140/RG.2.2.11508.65929

[8] ZHOU, Xuesong a Jeffrey TAYLOR. DTALite: A queue-based mesoscopic traffic simulator for fast model evaluation and calibration. *Cogent Engineering* [online]. 2014, 2014-10-01, **1**(1), 961345-961345 [cit. 2025-08-13]. ISSN 2331-1916. Dostupné z: doi:10.1080/23311916.2014.961345

[9] OR, Sagi. Mesoscopic and Hybrid Simulations in PTV Vissim. In: PTV GROUP. *PTV Blog* [online]. 2025, June 25, 2025 [cit. 2025-08-13]. Dostupné z: https://blog.ptvgroup.com/en/technologyplus/mesoscopic-and-hybrid-simulations-in-ptv-vissim/

[10] WUNDERLICH, Karl, Meenakshy VASUDEVAN, Peiwei WANG (NOBLIS), Richard DOWLING a Vassili ALEXIADIS. Traffic Analysis Toolbox Volume III: Guidelines for Applying Traffic Microsimulation Modeling Software 2019 Update to the 2004 Version: Data Collection and Analysis. In: *U.S. Department of Transportation Federal Highway Administration* [online]. 2019 [cit. 2025-08-13]. Dostupné z: https://ops.fhwa.dot.gov/publications/fhwahop18036/chapter2.htm

[11] AIMSUN. Modeling Vehicle Movement. In: *Aimsun Next Users Manual* [online]. Barcelona, Spain: Aimsun SLU [cit. 2025-08-12]. Dostupné z: https://docs.aimsun.com/next/23.0.2/UsersManual/MicrosimulationModellingVehicleMovement.html

[12] OLSTAM, Johan a Andreas TAPANI. Comparison of Car-following models. *VTI meddelande* [online]. Linköping Sweden: Swedish National Road and Transport Research Institute, 2004 [cit. 2025-08-13]. ISSN 0347-6049. Dostupné z: https://www.researchgate.net/publication/265198439_Comparison_of_Car-following_models

[13] LOPEZ, Pablo Alvarez, Evamarie WIESSNER, Michael BEHRISCH, et al. Microscopic Traffic Simulation using SUMO. In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)* [online]. IEEE, 2018, s. 2575-2582 [cit. 2025-08-13]. Dostupné z: doi:10.1109/itsc.2018.8569938

[14] QAYYUM, Rida a Hina EJAZ. A Comparative Study of Location Based Services Simulators. *International Journal of Computer Engineering in Research Trends* [online]. IJCERT Publication House, 2020, 2020-11-18, **7**(11), 1 [cit. 2025-08-13]. ISSN 2349-7084. Dostupné z: doi:10.22362/ijcert/2020/v7/i11/v7i1101

[15] PTV GROUP. *PTV Vissim – Product Description* [online]. PTV GROUP [cit. 2025-08-12]. Dostupné z: www.ptvgroup.com/en-us/product_description_ptv_vissim_en.pdf

[16] OR, Sagi. Realistic Traffic Simulation: Driving Behavior is Key. In: PTV GROUP. *PTV Blog* [online]. 2025, June 24, 2025 [cit. 2025-08-13]. Dostupné z: https://blog.ptvgroup.com/en/technologyplus/realistic-traffic-simulation-driving-behavior-is-key/

[17] SAROJ, Abhilasha, Guanhao XU, Yunli SHAO a Chieh Ross WANG. A Systematic Comparison for Consistent Scenario Development Using Microscopic Simulation Software. In: *2024 Winter Simulation Conference (WSC)* [online]. IEEE, 2024, 2024-12-15, s. 194-205 [cit. 2025-08-13]. Dostupné z: doi:10.1109/wsc63780.2024.10838810

[18] AIMSUN. Aimsun Next Architecture. In: *Aimsun Next Users Manual* [online]. Barcelona, Spain: Aimsun SLU [cit. 2025-08-12]. Dostupné z: https://docs.aimsun.com/next/22.0.4/UsersManual/ScriptArchitecture.html

[19] AIMSUN. Command Line Options. In: *Aimsun Next Users Manual* [online]. Barcelona, Spain: Aimsun SLU [cit. 2025-08-12]. Dostupné z: https://docs.aimsun.com/next/23.0.2/UsersManual/CommandLine.html

[20] AIMSUN. Interfaces. In: *Aimsun Next Users Manual* [online]. Barcelona, Spain: Aimsun SLU [cit. 2025-08-12]. Dostupné z: https://docs.aimsun.com/next/23.0.2/UsersManual/AimsunNextInterfaces.html

[21] AIMSUN. Project Structure. In: *Aimsun Next Users Manual* [online]. Barcelona, Spain: Aimsun SLU [cit. 2025-08-12]. Dostupné z: https://docs.aimsun.com/next/23.0.2/UsersManual/ProjectStructure.html

[22] AIMSUN. Aimsun Next Files. In: *Aimsun Next Users Manual* [online]. Barcelona, Spain: Aimsun SLU [cit. 2025-08-12]. Dostupné z: https://docs.aimsun.com/next/22.0.1/UsersManual/AimsunFileFormats.html

[23] AIMSUN. Designing a Traffic Model. In: *Aimsun Next Users Manual* [online]. Barcelona, Spain: Aimsun SLU [cit. 2025-08-12]. Dostupné z: https://docs.aimsun.com/next/23.0.2/UsersManual/PreparingTrafficNetwork.html

[24] AIMSUN. Road Sections. In: *Aimsun Next Users Manual* [online]. Barcelona, Spain: Aimsun SLU [cit. 2025-08-12]. Dostupné z: https://docs.aimsun.com/next/23.0.2/UsersManual/SectionEditing.html

[25] AIMSUN. Nodes. In: *Aimsun Next Users Manual* [online]. Barcelona, Spain: Aimsun SLU [cit. 2025-08-12]. Dostupné z: https://docs.aimsun.com/next/23.0.2/UsersManual/NodeEditing.html

[26] AIMSUN. Centroids. In: *Aimsun Next Users Manual* [online]. Barcelona, Spain: Aimsun SLU [cit. 2025-08-12]. Dostupné z: https://docs.aimsun.com/next/23.0.2/UsersManual/CentroidEditing.html

[27] AIMSUN. OD Matrices. In: *Aimsun Next Users Manual* [online]. Barcelona, Spain: Aimsun SLU [cit. 2025-08-12]. Dostupné z: https://docs.aimsun.com/next/23.0.2/UsersManual/ODMatrixEditing.html

[28] AIMSUN. Traffic States. In: *Aimsun Next Users Manual* [online]. Barcelona, Spain: Aimsun SLU [cit. 2025-08-12]. Dostupné z: https://docs.aimsun.com/next/23.0.2/UsersManual/TrafficStateEditing.html

[29] AIMSUN. Demand. In: *Aimsun Next Users Manual* [online]. Barcelona, Spain: Aimsun SLU [cit. 2025-08-12]. Dostupné z: https://docs.aimsun.com/next/23.0.2/UsersManual/DemandOverview.html

[30] AIMSUN. Arrivals. In: *Aimsun Next Users Manual* [online]. Barcelona, Spain: Aimsun SLU [cit. 2025-08-12]. Dostupné z: https://docs.aimsun.com/next/23.0.2/UsersManual/ArrivalsAlgorithms.html

[31] AIMSUN. Modeling Vehicle Movement. In: *Aimsun Next Users Manual* [online]. Barcelona, Spain: Aimsun SLU [cit. 2025-08-12]. Dostupné z: https://docs.aimsun.com/next/23.0.2/UsersManual/MicrosimulationModellingVehicleMovement.html

[32] AIMSUN. Managing Traffic. In: *Aimsun Next Users Manual* [online]. Barcelona, Spain: Aimsun SLU [cit. 2025-08-12]. Dostupné z: https://docs.aimsun.com/next/23.0.2/UsersManual/TrafficManagement.html

[33] AIMSUN. Aimsun Next software. In: *Aimsun Next Users Manual* [online]. Barcelona, Spain: Aimsun SLU [cit. 2025-08-12]. Dostupné z: https://docs.aimsun.com/next/23.0.2/UsersManual/AimsunNextSoftware.html

[34] AIMSUN. Vehicle based Simulators. In: *Aimsun Next Users Manual* [online]. Barcelona, Spain: Aimsun SLU [cit. 2025-08-12]. Dostupné z: https://docs.aimsun.com/next/23.0.2/UsersManual/VehicleBasedSimulatorsIntro.html

[35] AIMSUN. Running a Simulation. In: *Aimsun Next Users Manual* [online]. Barcelona, Spain: Aimsun SLU [cit. 2025-08-12]. Dostupné z: https://docs.aimsun.com/next/23.0.2/UsersManual/RunningSimulation.html

[36] AIMSUN. Scenarios, Experiments, Results, and Replications. In: *Aimsun Next Users Manual* [online]. Barcelona, Spain: Aimsun SLU [cit. 2025-08-12]. Dostupné z: https://docs.aimsun.com/next/23.0.2/UsersManual/ScenariosExperimentsResultsReplications.html

[37] AIMSUN. Dynamic Traffic Assignment. In: *Aimsun Next Users Manual* [online]. Barcelona, Spain: Aimsun SLU [cit. 2025-08-12]. Dostupné z: https://docs.aimsun.com/next/23.0.2/UsersManual/DynamicTrafficAssignment.html

[38] AIMSUN. Dynamic User Equilibrium (DUE). In: *Aimsun Next Users Manual* [online]. Barcelona, Spain: Aimsun SLU [cit. 2025-08-12]. Dostupné z: https://docs.aimsun.com/next/23.0.2/UsersManual/DynamicUserEquilibrium.html

[39] AIMSUN. Microsimulation Process. In: *Aimsun Next Users Manual* [online]. Barcelona, Spain: Aimsun SLU [cit. 2025-08-12]. Dostupné z: https://docs.aimsun.com/next/23.0.2/UsersManual/MicrosimulationProcess.html

[40] AIMSUN. Aimsun Next API. In: *Aimsun Next Users Manual* [online]. Barcelona, Spain: Aimsun SLU [cit. 2025-08-12]. Dostupné z: https://docs.aimsun.com/next/23.0.2/UsersManual/Api.html

[41] AIMSUN. API Architecture. In: *Aimsun Next Users Manual* [online]. Barcelona, Spain: Aimsun SLU [cit. 2025-08-12]. Dostupné z: https://docs.aimsun.com/next/23.0.2/UsersManual/ApiArchitecture.html

[42] AIMSUN. Aimsun Next API Detector Measures. In: *Aimsun Next Users Manual* [online]. Barcelona, Spain: Aimsun SLU [cit. 2025-08-12]. Dostupné z: https://docs.aimsun.com/next/22.0.2/UsersManual/ApiDetectorMeasures.html

[43] AIMSUN. Aimsun Next API Incidents. In: *Aimsun Next Users Manual* [online]. Barcelona, Spain: Aimsun SLU [cit. 2025-08-12]. Dostupné z: https://docs.aimsun.com/next/23.0.2/UsersManual/ApiIncidents.html

[44] AIMSUN. Aimsun Next API Managing Transit. In: *Aimsun Next Users Manual* [online]. Barcelona, Spain: Aimsun SLU [cit. 2025-08-12]. Dostupné z: https://docs.aimsun.com/next/23.0.2/UsersManual/ApiManagePublicTransport.html

[45] AIMSUN. Aimsun Next API OD Demand. In: *Aimsun Next Users Manual* [online]. Barcelona, Spain: Aimsun SLU [cit. 2025-08-12]. Dostupné z: https://docs.aimsun.com/next/23.0.2/UsersManual/ApiODDemand.html

[46] AIMSUN. Aimsun Next API Traffic States Demand. In: *Aimsun Next Users Manual* [online]. Barcelona, Spain: Aimsun SLU [cit. 2025-08-12]. Dostupné z: https://docs.aimsun.com/next/23.0.2/UsersManual/ApiTrafficStates.html

[47] AIMSUN. Aimsun Next API Pedestrians. In: *Aimsun Next Users Manual* [online]. Barcelona, Spain: Aimsun SLU [cit. 2025-08-12]. Dostupné z: https://docs.aimsun.com/next/23.0.2/UsersManual/ApiPedestrians.html

[48] AIMSUN. Aimsun Next API Vehicle Tracking. In: *Aimsun Next Users Manual* [online]. Barcelona, Spain: Aimsun SLU [cit. 2025-08-12]. Dostupné z: https://docs.aimsun.com/next/23.0.2/UsersManual/ApiVehicleTracking.html

[49] AIMSUN. Building a Microscopic Aimsun Next API. In: *Aimsun Next Users Manual* [online]. Barcelona, Spain: Aimsun SLU [cit. 2025-08-12]. Dostupné z: https://docs.aimsun.com/next/23.0.2/UsersManual/ApiBuildAimsunApi.html

[50] AIMSUN. Aimsun Next Scripting. In: *Aimsun Next Users Manual* [online]. Barcelona, Spain: Aimsun SLU [cit. 2025-08-12]. Dostupné z: https://docs.aimsun.com/next/23.0.2/UsersManual/ScriptIntro.html

[51] Introduction. In: ENCODE. *Uvicorn* [online]. [cit. 2025-08-13]. Dostupné z: https://www.uvicorn.org/

[52] FastAPI features. In: *FastAPI* [online]. [cit. 2025-08-13]. Dostupné z: https://fastapi.tiangolo.com/

[53] Pydantic: Welcome to Pydantic. In: *Pydantic* [online]. [cit. 2025-08-13]. Dostupné z: https://docs.pydantic.dev/latest/

[54] Pytest: helps you write better programs. In: *Pytest documentation* [online]. © 2015 [cit. 2025-08-13]. Dostupné z: https://docs.pytest.org/en/stable/

[55] The import system. In: PYTHON. *Welcome to Python.org* [online]. © 2001-2025 [cit. 2025-08-13]. Dostupné z: https://docs.python.org/3.10/reference/import.html