ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA DOPRAVNÍ

Bc. Jakub Krejčí

Návrh výukové databáze dopravních dat

Design of educational traffic data database

Diplomová práce

2019

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta dopravní
děkan
Konviktská 20, 110 00 Praha 1

K620..... **Ústav dopravní telematiky**

ZADÁNÍ DIPLOMOVÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení studenta (včetně titulů):

Bc. Jakub Krejčí

Kód studijního programu a studijní obor studenta:

N 3710 – IS – Inteligentní dopravní systémy

Název tématu (česky): **Návrh výukové databáze dopravních dat**

Název tématu (anglicky): Design of Educational Traffic Data Database

Zásady pro vypracování

Při zpracování diplomové práce se řiďte následujícími pokyny:

- Analyzujte dostupná dopravní data o provozu v Praze.
- Analyzujte pedagogické potřeby v oblasti dostupnosti dopravních dat.
- Navrhněte databázi dopravních dat pro výukové potřeby odborných předmětů.
- Navrhněte pravidla pro pravidelnou aktualizaci uložených dat z dostupných zdrojů.
- Při návrhu databáze zohledněte potřeby a požadavky uživatelů a správců databáze.

ČVUT
FD

Rozsah grafických prací: dle pokynů vedoucího

Rozsah průvodní zprávy: minimálně 55 stran textu (včetně obrázků, grafů a tabulek, které jsou součástí průvodní zprávy)

Seznam odborné literatury: Hernandes Michael J.: Návrh databází, Praha, Grada 2006

Riordan Rebecca M.: Vytváříme relační databázové aplikace, Brno, Computer Press 2000

Transmitter 3 - komunikační protokol

Vedoucí diplomové práce: **Ing. Martin Langr, Ph.D.**

Datum zadání diplomové práce: **27. července 2018**
(datum prvního zadání této práce, které musí být nejpozději 10 měsíců před datem prvního předpokládaného odevzdání této práce vyplývajícího ze standardní doby studia)

Datum odevzdání diplomové práce: **2. prosince 2019**
a) datum prvního předpokládaného odevzdání práce vyplývající ze standardní doby studia a z doporučeného časového plánu studia
b) v případě odkladu odevzdání práce následující datum odevzdání práce vyplývající z doporučeného časového plánu studia

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE Fakulta dopravní L. S.

Ing. Zuzana Bělinová, Ph.D.
vedoucí
Ústavu dopravní telematiky

doc. Ing. Pavel Hrubeš, Ph.D.
děkan fakulty

Potvrzuji převzetí zadání diplomové práce.

Bc. Jakub Krejčí
jméno a podpis studenta

V Praze dne ..... 28. května 2019

Poděkování

Na tomto místě bych rád poděkoval všem, kteří mi poskytli podklady pro vypracování této práce. Zvláště pak děkuji vedoucímu mé diplomové, a to Ing. Martinu Langrovi, Ph.D. a také kolegovi Michalu Kovaljovovi, za odborné konzultování mé práce a za rady, které mi poskytovali.

Prohlášení

Předkládám tímto k posouzení a obhajobě diplomovou práci, zpracovanou za závěr studia na ČVUT v Praze Fakultě dopravní.

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu § 60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 2. prosince 2019 Jakub Krejčí

……

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta dopravní

Ústav dopravní telematiky

Návrh výukové databáze dopravních dat

Diplomová práce

prosinec 2019

Bc. Jakub Krejčí

Abstrakt

Předmětem diplomové práce „Návrh výukové databáze dopravních dat“ je shrnutí uživatelských požadavků na systém databáze a navazujících uživatelských aplikací, určeného k výuce i výzkumu na Ústavu dopravní telematiky a návrh tohoto nástroje.

Klíčová slova: dopravní data, dopravní databáze, zpracování dat, systémový popis, zdroje dopravních dat, výuka

CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Transportation Sciences

Department of Transport Telematics

Design of Educational Traffic Data Database

Diploma thesis

December 2019

Bc. Jakub Krejčí

Abstract

The subject of the diploma thesis „Design of Educational Traffic Data Database“ is summary of user requirements of database system with user applications which is designed for teaching and research by Department of Transport Telematics and making design of this tool.

Key words: transportation data, traffic data, transportation database, traffic database, data processing, system description, sources of transportation data, sources of traffic data, education

Obsah

Seznam zkratek

GIS – Geografický informační systém

SEČ – Středoevropský čas

SELČ – Středoevropský letní čas

SDDŘ – Strategický dopravní detektor řezový

TSK – Technická správa komunikací (Praha)

UITS – Úvod do inteligentních dopravních systémů; předmět vyučovaný katedrou dopravní telematiky; kód předmětu: 20UITS

UTC – Coordinated Universal Time

– Koordinovaný světový čas

XML – Extensible Markup Language

– Rozšiřitelný značkovací jazyk

Úvod

Hlavním motivem tvorby výukové databáze dopravních dat je pedagogická potřeba seznámení studentů s reálnými způsoby získávání dopravních dat a jejich zpracováním a porozuměním. V době před odevzdáním této diplomové práce jsou takto používána zejména data ze strategických detektorů pro výukové účely předmětu 20UITS, ve kterém studenti dopravní data stahují a zpracovávají. Autor má zkušenosti s dosavadním procesem přípravy dat pro studentská zadání a cílem navrhovaného systému databází a aplikací je usnadnit a zefektivnit tento zdlouhavý a nespolehlivě fungující proces.

Práce čtenáře zasvětí do návrhu databázového systému dopravních dat. Tím se rozumí relační databáze a navazující aplikace, které jsou nezbytné pro správný chod tohoto systému. Práce je založena na předchozí činnosti autora a částečně také na jeho bakalářské práci s názvem "Analýza požadavků pro tvorbu výukové databáze dopravních dat" z roku 2017. Zmiňovaná předchozí závěrečná práce byla orientována především na uživatelské požadavky a jejich základní definice. Tato diplomová práce poznatky prohlubuje a přichází s výsledky v podobě návrhu relační databáze a navazujících aplikací. Je také spjata s předchozí prací autora v oblasti dopravních dat a několikaletými zkušenostmi jejich zpracování při spolupráci na projektech Ústavu dopravní telematiky.

Aby taková databáze byla funkční a používaná, musí vyhovovat všem jejím uživatelům a její funkce musí splňovat co nejvíce uživatelských požadavků. Dále je také nutné podrobně znát obsah dat a jejich původ, se kterým se v práci lze také seznámit.

Cílem práce je tedy vytvořit podpůrnou publikaci pro tvorbu databázového systému a systému navazujících aplikací s jednotlivými částmi návrhu založených na znalostech o zdroji dat a o požadavcích uživatelů.

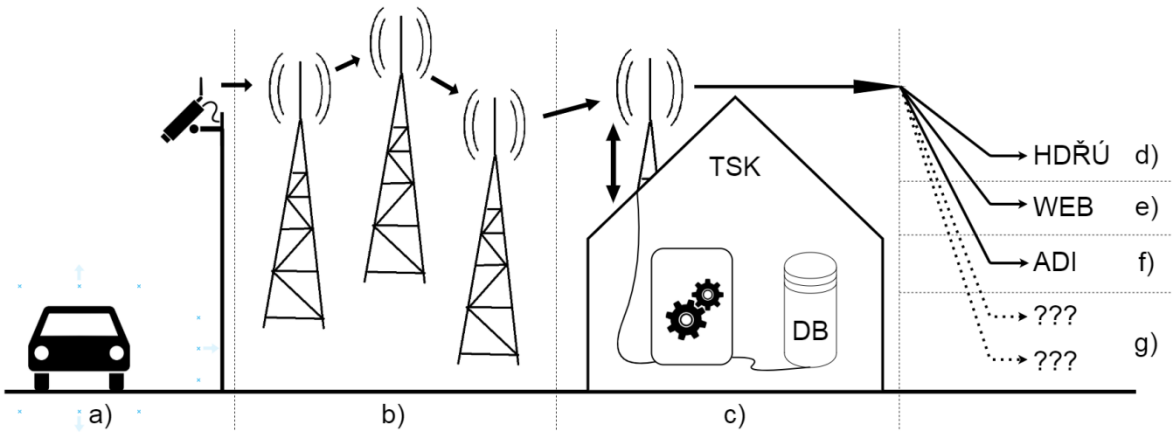
1 Dostupný zdroj dopravních dat

V této kapitole se čtenář blíže seznámí s dostupným zdrojem dopravních dat uvažovaným coby vhodným pro výukové i výzkumné účely Ústavu dopravní telematiky FD ČVUT. Bude zde stručně popsán systém od vzniku dat až po jejich distribuci a bude též popsána dosavadní klientská aplikace, která slouží k příjmu těchto dat.

1.1 Dostupný zdroj jako systém

Jednotlivé části systému budou popsány i z hlediska vlastníka a odpovědných osob. Podrobným odborným zdrojem je veřejná smlouva [1] na provádění údržby strategických dopravních detektorů uzavřená mezi firmami TSK a.s. (dále jen TSK) a Eltodo a.s. (dále jen Eltodo) – zejména zadávací dokumentace.

Pro lepší pochopení systému, ve kterém dopravní data vznikají bylo nakresleno schéma na obrázku č. 1, které obsahuje jednotlivé části systému **a** až **g**, které budou v této kapitole popsány.



Obrázek 1- Schéma systému sběru a zpracování dopravních dat TSK; zdroj: archiv autora

1.1.1 Detektory

Do skupiny detektorů, které posílají dopravní data, patří několik druhů detektorů mající různé vlastnosti a měřící různé dopravní veličiny popsané v kapitole 1.1.2 . Všechny detektory vlastní TSK.

Prvním a v rámci výukové databáze nejdůležitějším, typem detektorů je CollectR, často též známý jako SDDŘ. Jedná se o kamerový detektor druhé generace řady Traficam od výrobce Flir [2], který je schopen měřit dopravní veličiny až na čtyřech jízdních pruzích. Zpravidla je detektor umístěn na sloupech veřejného osvětlení, které je spravované organizací Technologie Hlavního města Prahy, a. s.. Detektor posílá data pro každý pruh zvlášť, a je tedy z hlediska dat rozdělen na více virtuálních detektorů. Příkladem může být detektor na ulici V Holešovičkách na sloupu veřejného osvětlení č. R802586, který měří dopravní veličiny ve čtyřech pruzích pod různými ID - například virtuální detektor s ID: R802586_S1 udává data z pomalého pruhu ve směru do Holešovic. Podrobný popis lokalit detektorů je k nalezení v příloze servisní smlouvy [zdroj č.1, příloha č.1, kapitola 3.2]. Celkem je známo 107 fyzických detektorů měřící dopravní veličiny na 348 jízdních pruzích. Detektory mají mít záložní baterie na minimálně 20 hodin provozu po výpadku napájení. Jejich správcem je firma Eltodo. První generace těchto detektorů čítá v Praze asi 20 ks což odpovídá 78 jízdním pruhům. Je však značně zastaralá, měří pouze intenzitu dopravy a lokace detektorů se nepodařilo zjistit.

Dalším typem jsou strategické dopravní detektory úsekové (dále jen SDDU), které měří dopravní veličiny nikoliv v místě, ale v úseku ohraničeném kamerami rozpoznávající poznávací značky vozidel. Tento systém sám na základě časů vjezdu a výjezdu konkrétních vozidel z/do úseku vypočítá dopravní veličiny a posílá data v 5min intervalech. Popis lokalit detektorů je k nalezení v příloze A, nicméně se nepodařilo zjistit přiřazení známých označení detektorů k lokalitám.

Nejpočetnější skupinou detektorů jsou křižovatkové indukční smyčky. Tyto detektory jsou napojeny ve většině světelně řízených křižovatek, konkrétně ve 259 křižovatkách a skupina obsahuje 4565 detektorů. V popisovaném systému se jedná o typ detektoru SCALA. Tyto detektory údajně nejsou nikde souhrnně popsány. Přiřazení detektorů k jízdním pruhům je

však dostupné v dokumentu zvaném Dopravní Řešení. Tento dokument pro každou křižovatku zvlášť je možno po vyžádání získat u provozovatele křižovatek.

Posledním typem detektorů je WIM (Weight in motion). Ty využívají tlakové senzory zabudované ve vozovce ve spolupráci s indukčními smyčkami a s videodetekcí. Na základě hmotnosti, rozvoru a počtu náprav či šířky kol dokáže detektor klasifikovat vozidla do 13 kategorií. Dle ročenky TSK bylo na konci roku 2015 v Praze 7 těchto detektorů. Lokality těchto detektorů se nepodařilo zjistit. Obdobně jsou na tom detektory typu MP, MUR.

Přes dlouhodobou snahu autora zjistit více informací o detektorech je řada těchto informací (jako například lokace detektorů) z pohledu Ústavu dopravní telematiky neznámá. Důvodem je nepřehledný registr veřejných smluv, nedostatečně odborně vypracované smlouvy o údržbě jednotlivých telematických subsystémů, a ochrana know-how soukromých firem.

1.1.2 Komunikační síť

Komunikace mezi detektory a výpočetním centrem je zajišťována bezdrátovou sítí TETRA provozovanou Hlavním městem Praha. Síť TETRA slouží pro komunikaci bezpečnostním a záchranným složkám, přenosům dat z dopravních detektorů i například přenosům dat informačním cedulím. Přestože standard [3] obsahuje a doporučuje zabezpečenou komunikaci, má síť TETRA v Praze bezpečnostní mezery v podobě komunikace nezabezpečené [4]. Tyto bezpečnostní mezery však autor nepovažuje pro přenos dat z dopravních detektorů za bezpečnostní hrozbu.

1.1.3 Uložení a zpracování dat

Analyzovaný systém toku dopravních dat výše uvedených detektorů, respektive jejich zpracovávající servery ve vlastnictví TSK v současnosti spravuje firma Eltodo. Data ukládá do databáze pro další zpracování a distribuci. Některá chybějící data je možno na základě požadavků TSK [zdroj č. 5, příloha č.1] dopočítat a agregovat viz kapitola 1.3.4 . Jsou známy minimálně tři větve výstupů dat. Je možné že existují i další, například

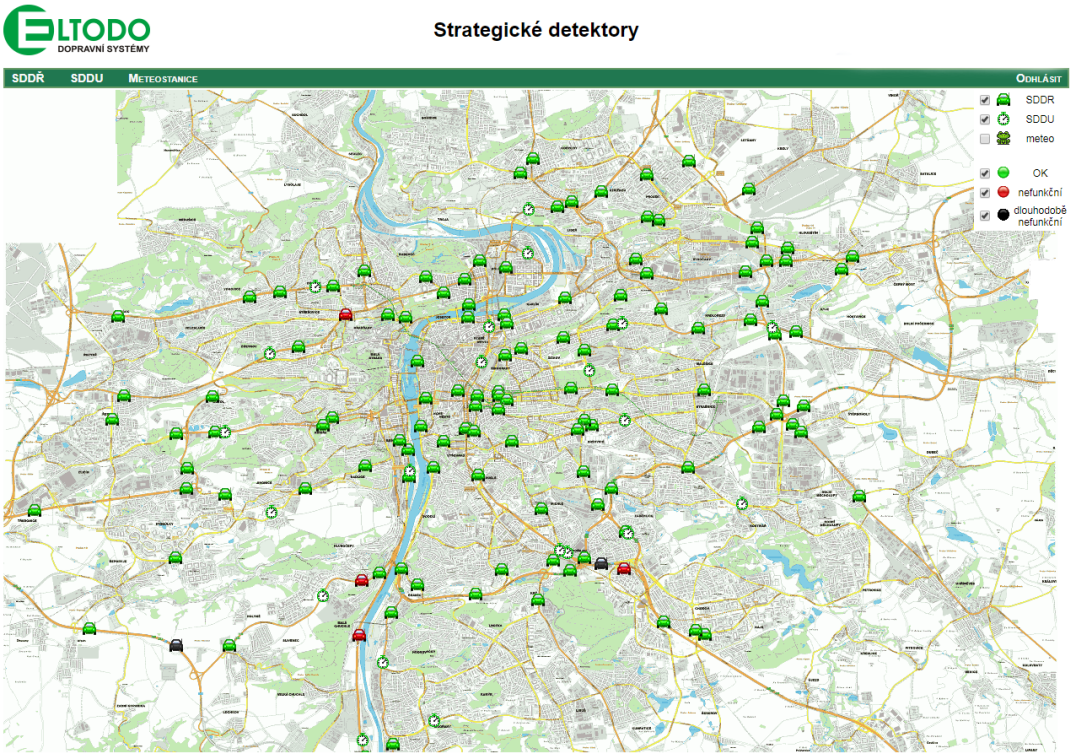
servisní, či jiné výstupy dalším nepopsaným subjektům. Následující tři výstupy budou samostatně popsány v jednotlivých podřazených podkapitolách 1.1.4 až 1.1.6 .

1.1.4 Výstup do HDŘÚ

TSK využívá data z detektorů pro tvorbu dopravního modelu Prahy. To je důležitý nástroj pro dopravní inženýrství a rozvoj města [6], zejména pro řešení dopravně-inženýrských problematik v současném a střednědobém horizontu.

1.1.5 Webová aplikace

Druhý výstup je webová aplikace dostupná na webu stratdet.eltodo.cz [11]. Tato aplikace slouží především objednateli údržby systému detektorů – firmě TSK pro kontrolu a přehled, zdali všechny detektory fungují, jak mají. Pro přístupové údaje je nutno požádat jednu ze zúčastněných stran. Náhled funkce vyhodnocování stavu v podobě mapy je na obrázku č. 2. a v podobě tabulky na obrázku č. 3. Následující náhledy jsou veřejně nepřístupné, aplikace je spravována firmou Eltodo.



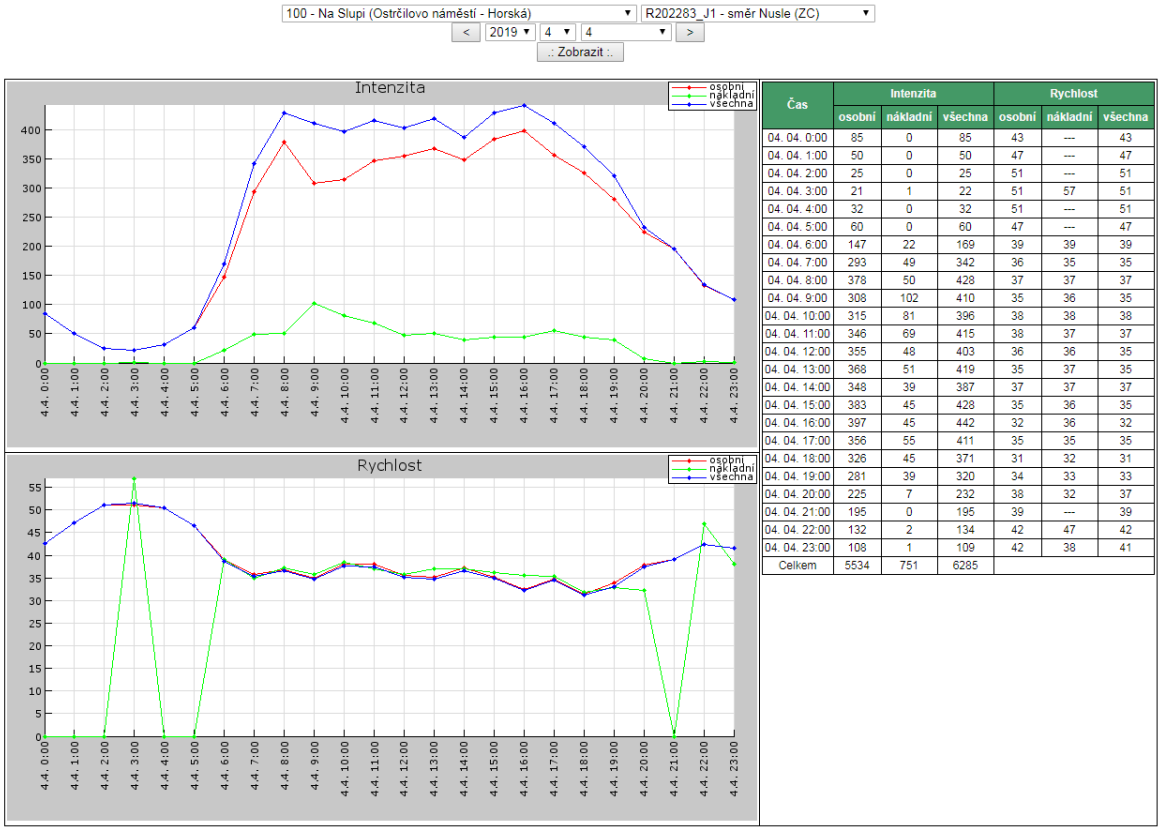
Obrázek 2 - Mapa strategických detektorů; zdroj: stratdet.eltodo.cz [11]

2. generace

	Adresa	Popis	Stav	Sloup VO	ID dle TSK	Souřadnice
	1	Modřanská (Branická - rampa Barrandovský most)		R401552	00011 00013	E14°24'30,8", N50°2'17,9"
	2	Modřanská (Železniční most [Inteligence] - Údolní)		R401670	00023	E14°24'18,4", N50°1'43,5"
				R401671	00021	
	3	Černokostelecká (Ústřední - rampy Štěrboholské radiály)		R005590	00032	E14°32'27,6", N50°4'13,6"
				R005589	00034	

Obrázek 3 - Ukázka seznamu strategických detektorů; zdroj: stratdet.eltodo.cz [11]

Webová aplikace také umožňuje zobrazit jednodenní přehled průběhu intenzity a rychlosti dopravního proudu. Tato funkce je zobrazena na obrázku č. 4 . Zobrazené veličiny jsou pouze agregované do hodiny (více o agregaci v kapitole 1.3.4). Z hlediska historie je možnost si přes web prohlížet agregovaná data zpětně až do listopadu 2011. Grafy je možno uložit ve formátu JPEG a hodnoty v tabulce lze zkopírovat, ne však uložit jako soubor.



Čas	Intenzita			Rychlost		
	osobní	nákladní	všechna	osobní	nákladní	všechna
04. 04. 0:00	85	0	85	43	---	43
04. 04. 1:00	50	0	50	47	---	47
04. 04. 2:00	25	0	25	51	---	51
04. 04. 3:00	21	1	22	51	57	51
04. 04. 4:00	32	0	32	51	---	51
04. 04. 5:00	60	0	60	47	---	47
04. 04. 6:00	147	22	169	39	39	39
04. 04. 7:00	293	49	342	36	35	35
04. 04. 8:00	378	50	428	37	37	37
04. 04. 9:00	308	102	410	35	36	35
04. 04. 10:00	315	81	396	38	38	38
04. 04. 11:00	346	69	415	38	37	37
04. 04. 12:00	355	48	403	36	36	35
04. 04. 13:00	368	51	419	35	37	35
04. 04. 14:00	348	39	387	37	37	37
04. 04. 15:00	383	45	428	35	36	35
04. 04. 16:00	397	45	442	32	36	32
04. 04. 17:00	356	55	411	35	35	35
04. 04. 18:00	326	45	371	31	32	31
04. 04. 19:00	281	39	320	34	33	33
04. 04. 20:00	225	7	232	38	32	37
04. 04. 21:00	195	0	195	39	---	39
04. 04. 22:00	132	2	134	42	47	42
04. 04. 23:00	108	1	109	42	38	41
Celkem	5534	751	6285			

Obrázek 4 - Ukázka grafického rozhraní dat strategických detektorů; zdroj: stratdet.eltodo.cz [11]

Aplikace se z dopravních detektorů věnuje pouze typům SDDU a SDDŘ (SDDŘ, či jinak CollectR). Je také rozšířená o klimatické detektory, které zobrazují ve stejném hodinovém měřítku denní průběhy teploty, denní průběhy vlhkosti vzduchu, teplotu rosného bodu a intenzitu srážek. Klimatické detektory jsou zmíněny v rámci stejné servisní smlouvy, nicméně jimi měřená data nemají vliv na dopravní veličiny.

1.1.6 Aplikace pro dopravní inženýry TSK

Pro účel této diplomové práce je nejpodstatnější výstup dat do tzv. Aplikace pro dopravní inženýry (zkr. ADI). Jedná se o formu přenosu dat v XML formátu pomocí protokolu T3 což je standart popisující komunikaci mezi klientem a serverem, který není nutný podrobněji popisovat vzhledem k záměru diplomové práce.

Pro stahování dat odkudkoliv pomocí internetového připojení slouží aplikace – klient s názvem T3 Remote Client, který v jednoduché formě vyvinula firma Eltodo. Ústav dopravní telematiky má tuto aplikaci plně k dispozici včetně zdrojového kódu. Konkrétnějšímu popisu této aplikace se věnuje následující podkapitola č. 1.3.

Dne 3. dubna 2019 došlo k náhlému výpadku této služby v podobě odmlčení komunikace ze strany serveru. Výpadek byl způsoben skončením platnosti servisní smlouvy mezi firmami TSK a Eltodo. Provoz byl obnoven 12. září 2019. Přesto byla služba nadále nestabilní a docházelo k častým disfunkcionalitám během listopadu 2019.

1.2 Dostupná data

Jedním z hlavních cílů této práce byl návrh podrobného řešení databáze dat. Aplikační moduly, které budou se budou starat o stahování dat z databáze spravované firmou Eltodo do plánované školní databáze budou vycházet ze současného stavu aplikace – klienta T3. Jak bude nově navrhovaná aplikace fungovat, a které funkce bude mít se lze dočíst v kapitole 3.2.

1.3 Aplikace T3 Remote Client v dosavadní verzi

Podkapitola vychází z návodu k aplikaci T3 Remote Client [5].

Jedná se o klientskou aplikaci, která se spojí se serverovou aplikací na straně TSK, předá jí požadavky a specifikace požadovaných dat. Po obdržení dat od serverové aplikace je uloží do souboru na straně klienta. Data jsou ve formátu XML.

Aplikace je napsána programovacím jazykem C# a její členění je poměrně jednoduché. Pro její stručný popis v současnosti je potřeba nahlédnout na její zdrojový kód, který se rozděluje do několika částí. Samotný zdrojový kód nelze zveřejnit.

V úvodní části se načtou deklarace systémových funkcí, které nejsou nijak významné pro popis chování aplikace. Nadále se vytvoří textový soubor, jehož název obsahuje datum a čas vytvoření souboru. Tento soubor slouží jako výstup pro uživatele aplikace. Poté se načtou parametry od uživatele, tedy požadavky a specifikace dat, které uživatel žádá. Po načtení parametrů se otevře spojení klientské aplikace a serverové aplikace na straně firmy Eltodo a specifikují se přístupové údaje. V tomto bodě se aplikace rozděluje na čtyři různé části dle požadavku uživatele.

Tyto čtyři části programu, tzv. metody mají každá své specifické parametry, a rozhodují o tom, jaká data budou uživatelovi zaslána – zda historická surová, historická agregovaná či online. V této podkapitole se jim bude nadále podrobněji věnováno.

Systém klientské a serverové aplikace není zcela konzistentní, protože při některých specifických kombinacích jednotlivých metod a typů detektorů se vyskytují anomálie – například nesoulad žádaných a obdržených dat. Neznalost těchto anomálií může vést i k mylnému významu dat při dalším zpracování. Proto je důležité systémové anomálie odhalit, podrobně popsat a v návrhu zohlednit a předejít chybám.

1.3.1 Metoda GetOnlineData

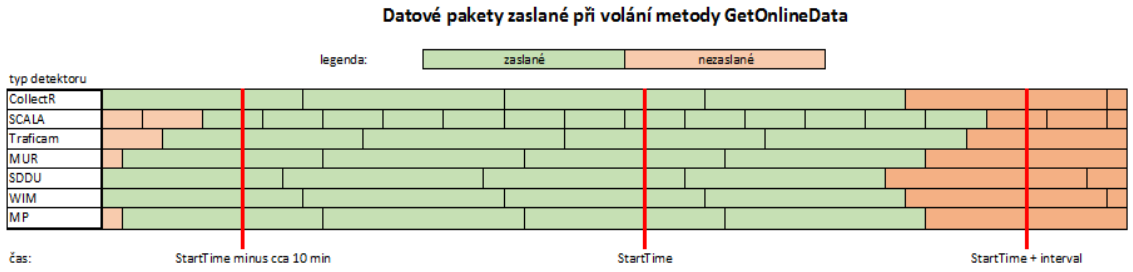
Metoda vrací zprávu obsahující surová aktuální data. Na jedno zavolání metody přijde jedna náhodná zpráva s daty. Pro řádné použití je nutné ji specifikovat (kromě přístupových údajů) také dvěma dalšími parametry.

Parametr „interval“ je parametr klientské aplikace, tedy na rozhraní klientská aplikace – uživatel. Tento parametr určuje, po jakou dobu je metoda opakovaně volána pro obdržení více než jedné zpráv. Parametr má jednotku vteřina.

Dalším důležitým parametrem je „token“. Tentokrát se jedná o parametr metody, tedy na rozhraní klientská aplikace – serverová aplikace. Je to identifikátor každé zprávy přijaté touto metodou. Pro docílení přijímání navazujících zpráv se s každým voláním metody musí uvádět parametr „token“ jenž přišel v předchozí zprávě. Parametr je bezrozměrný, a pro první volání metody se používá nulová hodnota.

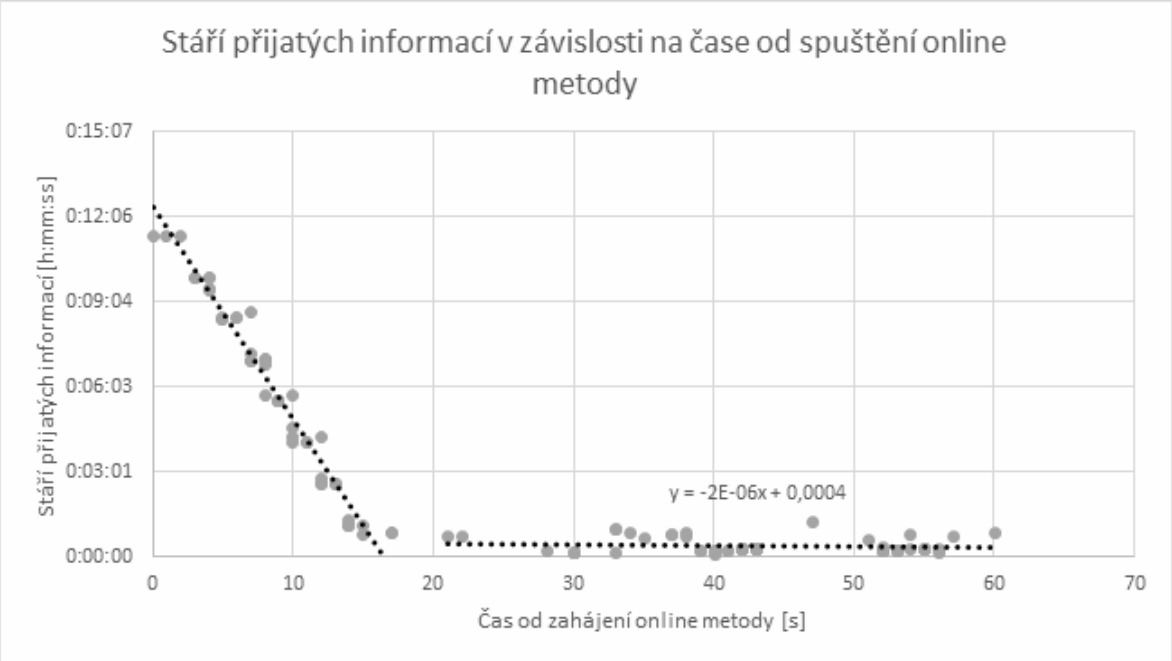
Stáří online zpráv

Zasílaná data jsou údaje o veličinách vztažená k určitému časovému období – specifickému měřícího intervalu daného typu detektoru. Data vztažená k tomuto období jsou zasílána vždy na konci tohoto měřícího intervalu. Začátek a konec období má každý detektor jinak a náhodně posunutý pro lepší rozložení zasílaného datového objemu v čase. Obrázek č. 5 vyjadřuje závislost přijatých dat na základě času spuštění opakovaného volání metody.

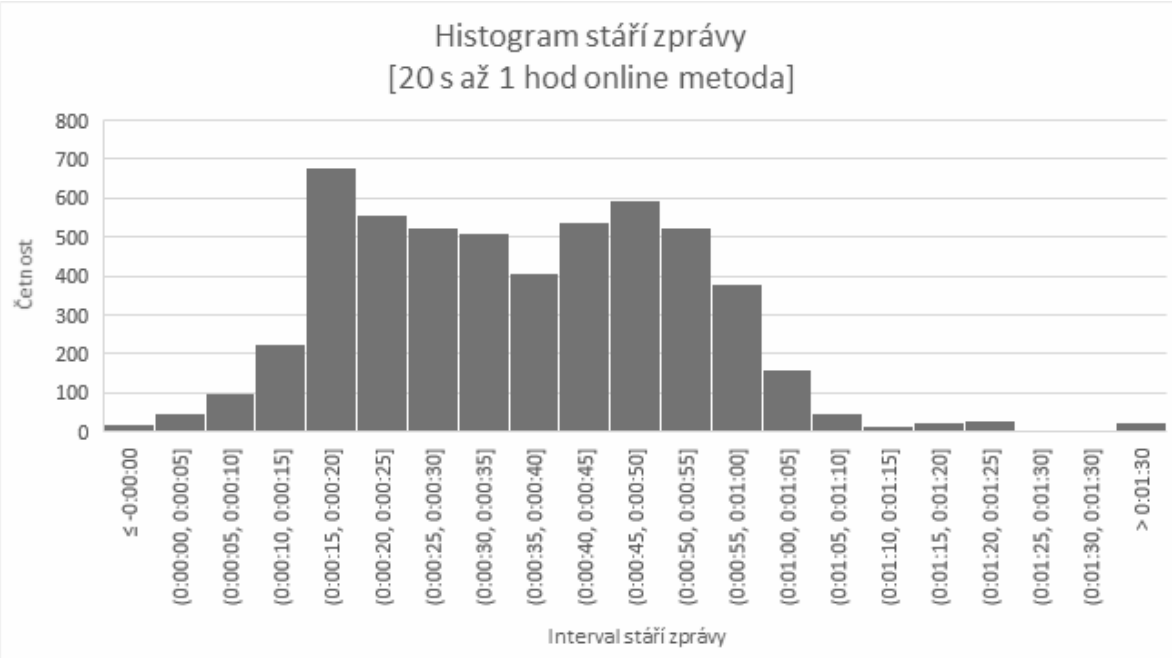


Obrázek 5 - Datové pakety zaslané při volání metody GetOnlineData

Po spuštění opakovaného volání metody GetOnlineData se však nejprve posílají data přibližně 10 minut stará, a až zhruba po 15 vteřinách začínají přicházet data aktuální. Tato skutečnost je vyjádřena grafem na obrázku č. 6. Lze také pozorovat že po zmíněném čase 15 sekund od spuštění opakovaného dotazování metody chodí převážně zprávy s maximálním stářím 1 minuta, což vyjadřuje histogram na obrázku č. 7.



Obrázek 6 - Graf stáří přijatých informací v závislosti na čase od spuštění metody GetOnlineData; zdroj: archiv autora

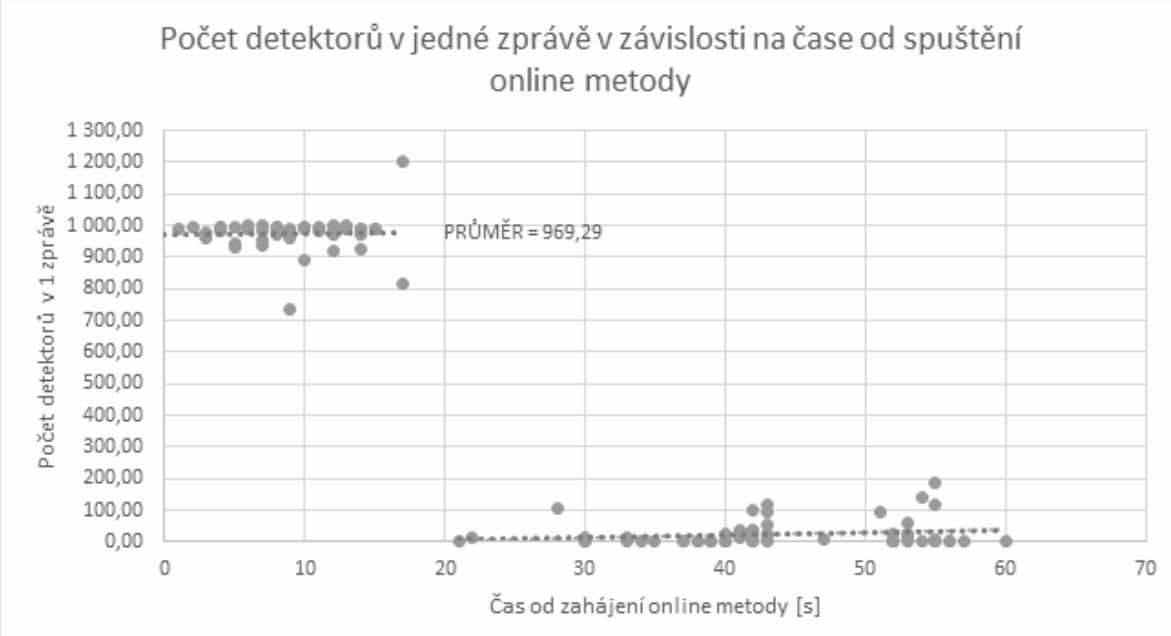


Obrázek 7 - Histogram stáří zpráv získaných metodou GetOnlineData; zdroj: archiv autora

Zvláštní anomálií u této metody (pouze u metody GetOnlineData) jsou však data detektorů typu CollectR. Ty mají časové razítko s nestandardní koncovkou určující dané časové pásmo. To je ve formátu jednoho písmene na rozdíl od standardní koncovky. Navíc data přichází s o hodinu nižším časem, tedy ve formátu 11:00:00Z namísto 12:00:00+01:00 (tento čas vyjadřuje poledne dne v zimním čase). Nicméně tato metoda nebude využita pro návrh stahovací aplikaci výukové databáze, tudíž tato anomálie nijak neohrozí její chod.

Analýza datového toku

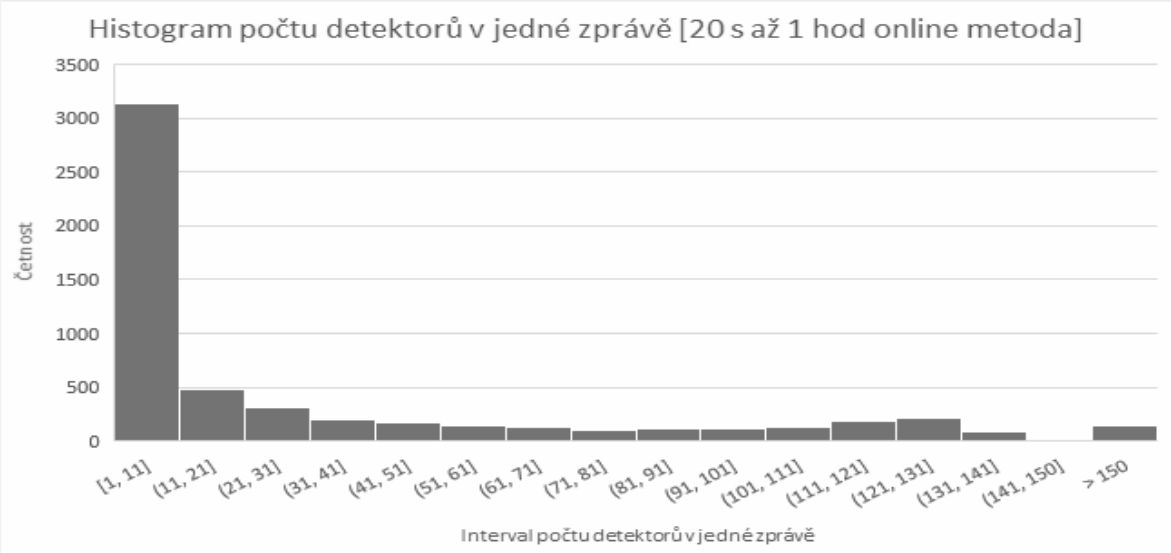
Pomocí metody GetOnlineData lze pozorovat zajímavý systémový parametr, a to sice maximální datový tok tohoto systému. Tedy objem dat přenesený za čas. Maximální datový tok nastává asi 15 vteřin od spuštění opakovaného volání metody, kdy přichází zprávy z posledních cca 10 minut. Obrázek č. 8 vyjadřuje z kolika měřících intervalů (od libovolného detektoru) přichází dat v jediné zprávě.



Obrázek 8 - Počet detektorů v jedné zprávě v závislosti na čase od spuštění metody GetOnlineData; zdroj: archiv autora

V prvních 15 vteřinách se jedná průměrně o 970 měřících intervalů v jedné zprávě, přičemž informace z jednoho časového měřícího intervalu od libovolného detektoru má v XML podobě průměrnou velikost 480 bajtů. Při průměrném počtu 2,75 zpráv za vteřinu

(v prvních 15 vteřinách) se dostáváme k maximálnímu toku 19,3 kB/s. Počet měřících intervalů v druhé části, kdy přicházejí data skutečně online, lépe vystihuje histogram na obrázku č. 9, z něhož je patrné že nejčastěji přijde malý počet zpráv.



Obrázek 9 - Histogram počtu detektorů v jedné zprávě získané metodou GetOnlineData; zdroj: archiv autora

1.3.2 Metoda GetOnlineData2

Metoda vrací zprávu obsahující surová aktuální data. Na rozdíl od metody GetOnlineData je zde navíc parametr metody „source“. Tento parametr omezuje online data na vybraný typ detektorů. Je-li tato hodnota tohoto parametru rovna například „CollectR“, budou přicházet pouze data od těchto detektorů.

Tato metoda se zdá neužitečná pro návrh plánované aplikace, proto nebyla blíže zkoumána.

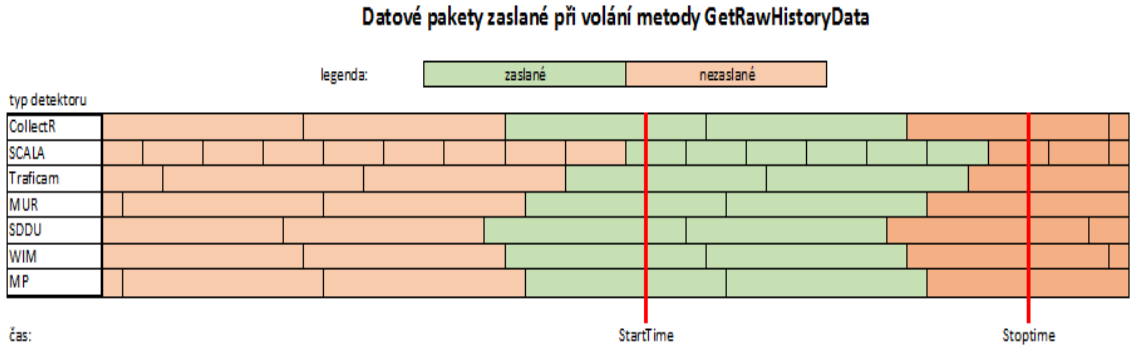
1.3.3 Metoda GetRawHistoryData

Metoda vrátí zprávu obsahující surová historická data. Při použití této metody se musí uvést (kromě přístupových údajů) také další čtyři parametry.

Pro specifikaci konkrétního detektoru se používá parametr „deviceId“, který určuje jeden konkrétní detektor. Při jednom volání metody je možno obdržet pouze data od jednoho

konkrétního detektoru. Navíc se musí specifikovat typ detektoru parametrem „deviceType“.

Pro specifikaci časového období přijatých dat slouží parametry „startTime“ a „stopTime“, které určují začátek a konec období. I zde platí, že se data odesílají na konci měřícího intervalu detektoru a lépe graficky vyjádřeno je to na obrázku č. 10.



Obrázek 10 - Datové pakety zaslané při volání metody GetRawHistoryData; zdroj: archiv autora

I zde je však jedna anomálie, a to sice u křižovatkových indukčních detektorů SCALA. U tohoto typu se mohou vracet zcela nelogicky hodnoty z jiných časových období. Byla tedy provedena reverzní inženýrská analýza pro zpracování požadovaných časů, které od klientské aplikace dostane serverová aplikace. Po sérii pokusů bylo zjištěno, jak se odpověď serverové aplikace chová. Vypovídá o tom obrázek č. 11 . Hranaté závorky rozdělují vrácené časové razítko na čas a časové pásmo. Kulaté závorky symbolizují čas UTC. Zadaným časem a zadaným pásmem je rozuměno čas a pásmo, které se posílají serverové aplikaci. Pásmo zadaného data je určeno podle toho, zda v zadaném datu a času platí letní či zimní čas.

CollectR a ostatní: $\left[\begin{pmatrix}\text{zadaný}\\ \text{čas}\end{pmatrix} - \begin{matrix}\text{zadané}\\ \text{pásmo}\end{matrix}\right) + \begin{matrix}\text{pásmo}\\ \text{zadaného}\\ \text{data}\end{matrix}\Big][\begin{matrix}\text{pásmo}\\ \text{zadaného}\\ \text{data}\end{matrix}]$

SCALA StartTime: $\left[\begin{pmatrix}\text{zadaný}\\ \text{čas}\end{pmatrix} - \begin{matrix}\text{zadané}\\ \text{pásmo}\end{matrix}\right) + \begin{matrix}\text{pásmo}\\ \text{zadaného}\\ \text{data}\end{matrix} + \begin{matrix}\text{pásmo}\\ \text{zadaného}\\ \text{data}\end{matrix}\Big][\begin{matrix}\text{pásmo}\\ \text{zadaného}\\ \text{data}\end{matrix}]$

SCALA StopTime: $[\begin{matrix}\text{zadaný}\\ \text{čas}\end{matrix}][\begin{matrix}\text{pásmo}\\ \text{zadaného data}\end{matrix}]$

Obrázek 11 - Chybové chování serveru TSK zjištěné reverzním inženýrstvím; zdroj: archiv autora

Podrobnější popis, jak datum zadat, tak aby byla navrácena hodnota, kterou uživatel skutečně očekává, je k naleznutí v obrázku č.12.

parametry	typ detektoru	CollectR a ostatní		SCALA			
	časový parametr	startTime i stopTime		startTime		stopTime	
požadovaný příklad (to co uživatel chce)	časové pásmo požadovaného dne	SEČ	SELČ	SEČ	SELČ	SEČ	SELČ
	příklad požadovaného dne	1. ledna	1. června	1. ledna	1. června	1. ledna	1. června
	požadovaná hodina	12:00:00 (poledne)		12:00:00 (poledne)		12:00:00 (poledne)	
očekávaný vstup (to co se očekává, že uživatel zadá)	zadaný čas a pásmo	12:00:00+01:00	12:00:00+02:00	12:00:00+01:00	12:00:00+02:00	12:00:00+01:00	12:00:00+02:00
	zadaný čas v UTC	11:00:00	10:00:00	11:00:00	10:00:00	11:00:00	10:00:00
odpověď serveru	vrácený čas a pásmo	12:00:00+01:00	12:00:00+02:00	13:00:00+01:00	14:00:00+02:00	12:00:00+01:00	12:00:00+02:00
	vrácený čas v UTC	11:00:00	10:00:00	12:00:00	12:00:00	11:00:00	10:00:00
nutno zadat	čas a pásmo	-	-	12:00:00+02:00	10:00:00+02:00	-	-
	slovně čas	požadovaný čas	požadovaný čas	požadovaný čas	požadovaný čas snížený o 2 hod	požadovaný čas	požadovaný čas
	slovně pásmo	vždy +01:00	vždy +02:00	vždy +02:00	vždy +02:00	cokoliv	cokoliv

Obrázek 12 - Ukázka úpravy odeslaných parametrů pro příjem správných dat; zdroj: archiv autora

U detektorů, s výjimkou SCALA, vše funguje logicky a očekávatelně, výstup odpovídá žádanému vstupu

U detektorů SCALA, při zadávání parametru startTime, je v den a čas, který odpovídá SEČ potřeba zadat požadovaný čas a pásmo zadat jako + 2. V den a čas, který odpovídá SELČ, je potřeba zadat požadovaný čas snížený o dvě hodiny, pásmo zadat také jako +2.

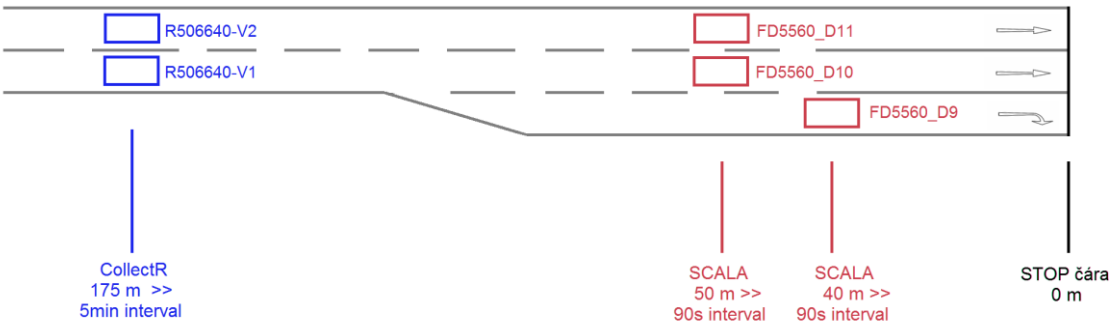
U detektorů SCALA, při zadávání parametru stopTime, vždy přijde odpověď s požadovaným časem a správným odpovídajícím pásmem, avšak systém zcela ignoruje zadané vstupní pásmo.

Verifikace a porovnání dat z detektorů SCALA a CollectR

Přestože přijatý čas někdy neodpovídá požadovanému, časové razítko u této metody odpovídá datům. Takováto anomálie byla ale popsána v metodě GetOnlineData, tudíž to není samozřejmostí a muselo to být zjištěno verifikací.

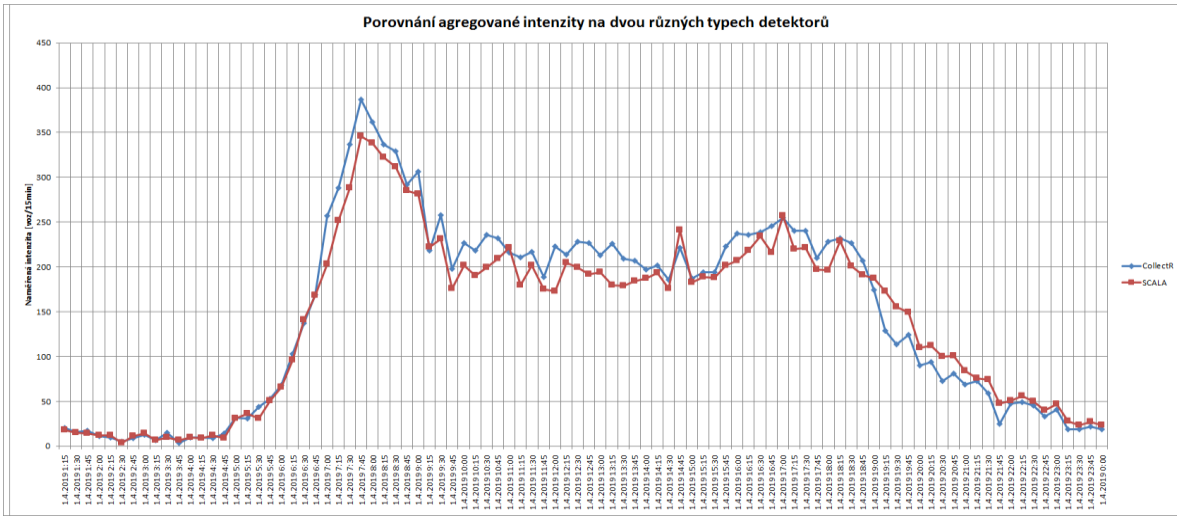
Verifikace probíhala porovnáním surových dat ze dvou typů detektorů. Z hlediska času probíhala verifikace ze surových dat jednoho dne, konkrétně 1. 4. 2019.

Byla zvolena nejvhodnější lokalita pro analýzu, konkrétně se jedná o východní vjezdové rameno do křižovatky Jeremiášova × Červeňanského v pražských Stodůlkách. Schéma lokality je k nalezení v obrázku č. 13. Tato lokalita byla vybrána, protože autor zná ID křižovatkových indukčních SCALA detektorů, takže mohly být porovnány s hypoteticky časově posunutými detektory CollectR.



Obrázek 13 - Verifikace a porovnání dat z detektorů SCALA a CollectR; zdroj: archiv autora

Jednodenní průběh součtu detektorů u 130 m vzdálených profilů lze vidět na obrázku č. 14. Časový úsek jednoho dne je nejvhodnější pro analýzu korelace dat, zejména při podezření zpoždění dat detektorů CollectR o jednu hodinu, tak jak je tomu u metody GetOnlineData. Data byla agregována do 15min intervalů, tedy do nejmenšího společného násobku 1,5 min a 5 min, coby intervaly měřených dat obou typů detektorů.



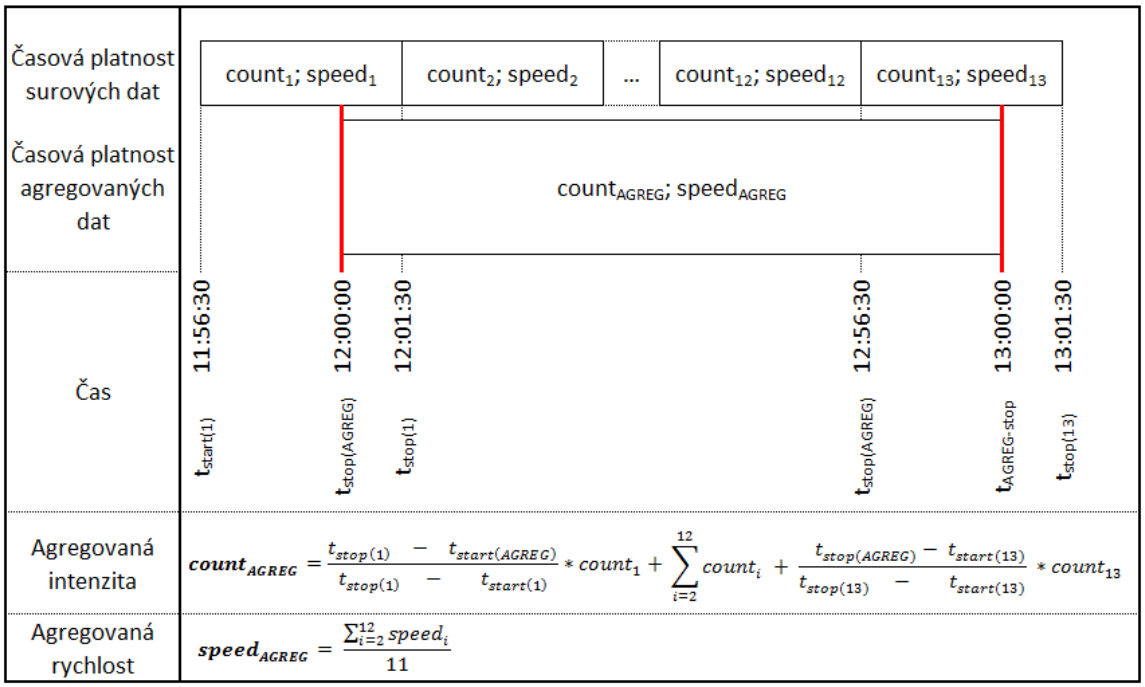
Obrázek 14 - Porovnání agregované intenzity na dvou různých typech detektorů; zdroj: archiv autora

Z grafu je patrné jednoznačně patrné, že hypotézu o hodinovém posunu u metody GetRawHistoryData lze vyloučit, korelační koeficient je roven 0,98. Je také zjevné, že oba typy detektorů měří s drobnou odchylkou. Podrobnější statistické analýzy však nebyly provedeny z důvodu nevýznamnosti, už jen proto, že detektory SCALA jsou v blízkosti stop-čáry, netvoří dokonale jeden profil, a hlavně mají rozdílný časový měřící interval.

Stručným závěrem lze tedy definovat, že data detektorů CollectR odpovídají časovému razítku, byť výstup serverové aplikace opět neodpovídá vstupu do klientské aplikace.

1.3.4 Metoda GetHistoryData

Metoda vrátí zprávu obsahující agregovaná historická data. Agregace dat slouží k přeměně množství dat do jejich jednodušší formy. V tomto případě se jedná o ucelení dat intenzity a rychlostí. Ze surových dat o intervalu 90 sekund či 5 minut se vypočítá jediná hodnota popisující veličinu v průběhu jedné hodiny. Agregovaná data veličin mají výhodu v tom, že se dají srovnávat s daty od ostatních detektorů, neboť jejich začátek i konec platnosti se u všech detektorů shodují a jejich délka je tedy stejná. Jedná se vždy o celé hodiny, například 12:00:00 až 13:00:00. Grafické vyjádření výpočtu agregovaných dat i se vzorci zjištěné z vlastní analýzy dat jsou na obrázku č.15.



Obrázek 15 - Popis a pravidla agregace surových dat; zdroj: archiv autora

Agregovaná hodinová intenzita, neboli počet vozidel za hodinu, odpovídá součtu intenzit všech celých intervalů mezi začátkem a koncem celých hodin. U intervalů, které jsou na rozmezí celých hodin, se předpokládá rovnoměrné rozložení počtu vozidel v čase. Zasahuje-li interval surových dat do požadované hodiny pro agregaci dat danou částí doby, pak intenzita pro tuto část doby bude vypočítána přímou úměrou. Totéž platí pro začáteční i pro koncový interval.

Agregovaná rychlost odpovídá aritmetickému průměru rychlostí v danou hodinu. U rychlosti na rozdíl od počtu se předpokládá, že je během krátkého intervalu surových dat relativně konstantní, tudíž je zbytečné ji rozdělovat

Pokud metoda nemá dostatek dat pro agregaci, vrátí chybovou hlášku. Chybějící data je systém schopen za určitých podmínek dopočítat. Z přílohy dokumentu popisující protokol Transmitter3 [zdroj č. 5, příloha č. 1] vyplývá, že dopočet pro data od profilových detektorů je možný pouze pro 3 intervaly v hodině, které nejsou po sobě jdoucí. Dopočet je aritmetický průměr předešlého a následujícího intervalu. Pro křižovatkové detektory jsou pravidla obdobné, avšak z důvodu kratších intervalů je maximální počet dopočítávaných intervalů roven čtyřem.

2 Požadavky pro návrh

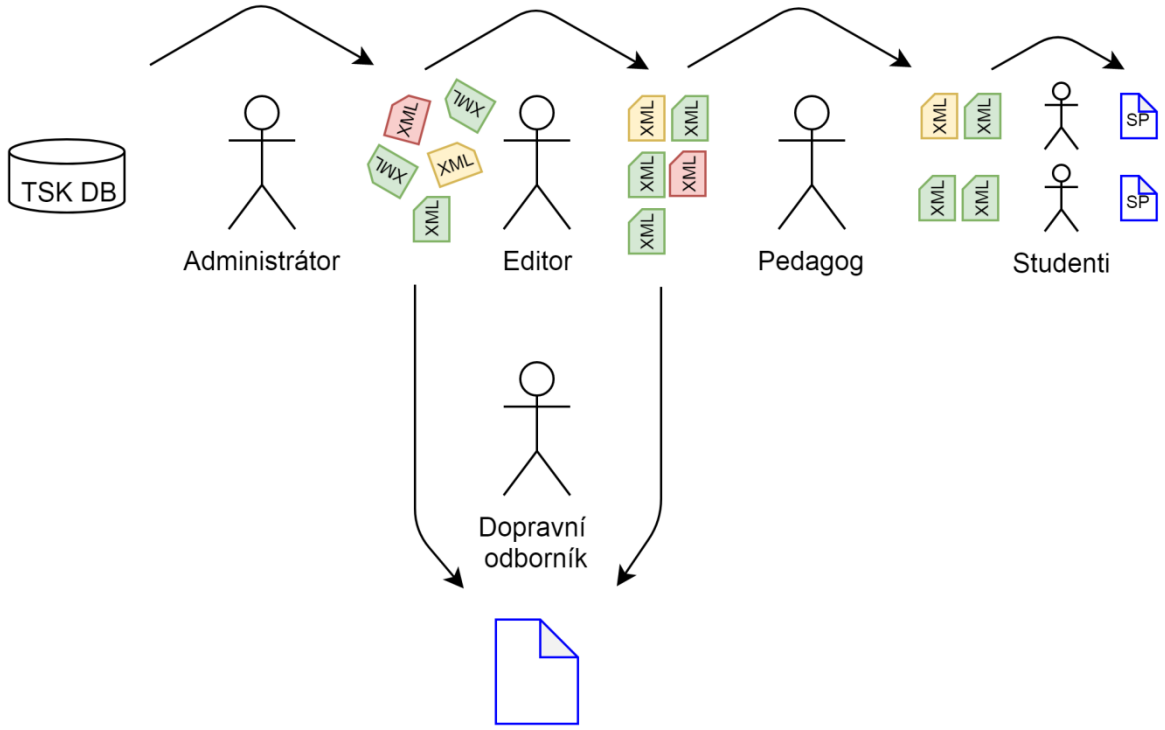
Cílem této diplomové práce je navrhnout systém databáze a aplikací která slouží nejen pedagogům odborných předmětů, ale i studentům, pracovníkům výzkumu a zároveň splňuje všechny technická pravidla pro návrh, očekávání dopravních inženýrů, funkčnost a udržitelnost.

Pro takový návrh je nezbytné analyzovat požadavky jednotlivých uživatelů. Autor tuto kapitolu zakládá na své předešlé bakalářské práci [7], kterou rozvíjí o další poznatky, které v průběhu dalších dvou let magisterského studia načerpal.

Zadání diplomové práce úzce souvisí s předmětem UITS. Pro splnění jedné z částí zápočtové práce musí studenti zpracovávat surová dopravní data z dopravních detektorů ze zmiňovaného zdroje z předchozí kapitoly. Předmět UITS je součástí akreditace platné od roku 2016 v době psaní DP již po čtvrté a od začátku předmětu je úkol zpracování těchto dat součástí zápočtové práce.

O tom, co znamenal systém přípravy a zpracování této úlohy pro všechny dotčené subjekty se lze dočíst v následujících podkapitolách této kapitoly.

2.1 Principy a funkce



Obrázek 16 - Zobrazení uživatelských rolí a jejich funkcí; zdroj: archiv autora

V popisovaném systému identifikujeme pět základních uživatelských rolí - administrátor, editor, dopravní odborník, pedagog a student.

Administrátor je zodpovědný za přístup k datovému zdroji, spravuje datový tok bez ohledu na obsah. Řeší funkčnost aplikací a vlastní datová úložiště.

Obsah dat má na starosti editor. Data zpracovává tak, aby s nimi mohli ostatní účastníci efektivně pracovat. Spolu se zpracováním dat je může zkoumat za účelem jejich popisu. Rozhoduje také o validitě dat.

Pedagog v popisovaném systému především tvoří zadání seminárních prací pro studenty odborných předmětů. Musí mít přehled o tom, zdali jsou zadání splnitelná z hlediska obsažených dat. Rovněž kontroluje velké množství hotových seminárních prací.

Student tvoří seminární práci za účelem odborného vzdělání a získání praxe se ziskem a zpracováním dopravních dat.

S výukovou větví systému nesouvisející, nicméně důležitým a nepomíjitelným účastníkem je dopravní odborník. Může to být libovolný zaměstnanec fakulty pracující na odborném výzkumu či student pracující na své závěrečné práci. Jeho nepomíjitelnost spočívá ve zkušenostech a odborných znalostech. Systému ve zpětné vazbě dodává odbornost.

2.2 Pedagogické požadavky

Ze všech požadavků pro návrh zamýšleného systému představuje skupina pedagogických požadavků zejména důležitou část z hlediska maximálního plnění účelu pro koncové uživatele. Tato skupina zahrnuje požadavky vylepšujících stávající systém jak z pohledu studentů, tak z pohledu pedagogů, respektive vyučujících odborných předmětů, jako například UITS.

Mezi cíle předmětu UITS [8] patří mimo jiné seznámit studenty se základy informačních systémů jako nadstavbou na dopravní infrastrukturu a seznámit se s prací s reálnými zdroji dopravních dat a s daty samotnými.

V dosavadním režimu výuky a tvorby seminárních zápočtových prací se pedagogové i studenti setkávali s řadou problémů způsobujících neefektivitu. Ve vstupních datech se do jisté míry objevovali výpadky, které znemožňovaly zadání splnit a tím i nesplnit cíle předmětu - práce s daty. Zpočátku se jednalo o počty zadání v řádu jednotek procent (v roce 2017 3% nevhodných zadání) avšak později se výpadky projevovaly četněji (v roce 2018 11% nevhodných zadání). Vzhledem ke stárnutí systému (především detektorů jako elektrotechnických zařízení) se dá očekávat, že trend výpadků zůstane mírně růstový.

V důsledků výpadků bylo potřeba více zadání, a tedy více času pro jejich přípravy.

Další nepříjemností pro pedagogy byla kontrola seminárních prací, respektive kontrola, že student samostatně práci zpracoval. Přestože měl každý student unikátní zadání, zkontrolovat, zda student vycházel například ze správných dat, je při velkém počtu studentů obtížné, a průběh dopravních veličin u všech zadání podobný.

I studenti museli čelit komplikacím, jako například uměle zkomplikovaná vstupní data, která se přes sebe překrývala nebo se v nich vícekrát opakovali XML deklarace. Serverová

ani klientská aplikace nebyla a není bezchybná, a řešení komplikací sice napomáhá výuce a porozumění, přesto tyto chyby byli zbytečné a bylo by žádoucí je odstranit.

2.2.1 Požadavky pro zlepšení systému v zájmu vyučujících

Jednoduchá pedagogická aplikace by měla umožnit přehlednou a snadnou tvorbu a kontrolu zadání. Vhodné je grafické vykreslení průběhu dopravních veličin ke každému zadání pro rychlou kontrolu.

Měl by být evidován přehled o tom, jak se který student dotazuje, jaké používá dotazy, kdy a na která data se ptá. Tento přehled dění může pomoci pedagogovi jako zpětná vazba například při konzultací se studentem.

Identita pedagogů bude ověřována jedním heslem sdíleným všem v komunitě cvičících předmětu UITS. Není potřeba vytvářet každému vyučujícímu své heslo a přihlašovací jméno.

Mělo by být k dispozici dostatek zadání i vzhledem k růstovému trendu výpadků dat.

2.2.2 Požadavky pro zlepšení systému v zájmu studentů

Jednoduchá studentská aplikace – stažení dat by se v zájmu seznámení s datovým zdrojem mělo uskutečňovat podobným způsobem a bez zbytečných komplikací, jako například nesmyslného doplňování kořenových tagů a odstraňování XML deklarací.

V zájmu studentů je taktéž simulace manipulace reálného nástroje pro stahování dat, tedy T3 Remote Client. Způsob dotazu do studentské aplikace by měl být identický se způsobem zadávání dotazu do T3 Remote Client

2.3 Dopravně inženýrské požadavky

Bez ohledů na předchozí účastníky je potřeba posuzovat jakoukoliv profesionální databázi dat z pohledu odbornosti. V případě zamýšlené výukové databáze dopravních dat se bude muset posoudit dopravní odbornost. Dopravního odborníka ve schématu všech uživatelů může představovat jakýkoliv dopravní odborník, který tuto databázi může použít ať už pro výzkum či odborně-vědeckou činnost.

Základním vědeckým požadavkem je předpoklad, že čím větší podrobnost dat je k dispozici, tím více a přesněji můžeme daný systém zkoumat. Tomu nasvědčuje i historie dopravního inženýrství. V roce 1937 vznikl první automatický sčítač dopravy [9], který sčítal počet vozidel za jednu hodinu a tisknul jej na proužek papíru. Byl to jeden hodinový údaj jedné veličiny na samostatném fyzickém nosiči.

Po čase, spolu s vývojem kapacity datových úložišť, výpočetního výkonu a rychlosti telekomunikací bylo umožněno přenášet údajů více, podrobněji a rychleji. Z více dat bylo možné získat více informací, například použitím dopravních modelů. Mezi nejznámější dopravní modely patří například charakteristiky dopravního proudu. Jednoduchým příkladem dopravního modelu může být třeba Greenshieldův model [10], který definuje vzájemnou závislost rychlosti, hustoty a intenzity dopravního proudu. Podobné modely umožňují například ze dvou veličin (rychlost a intenzita) určit další parametry dopravního proudu, jako například stupeň dopravy.

Právě rozšiřováním takovýchto parametrů dopravního proudu se o zkoumaném dopravním systému můžeme dozvědět více informací. Spolu s těmito typickými dopravními parametry ovlivňují zkoumaný dopravní systém i jiné, zdánlivě nesouvisející veličiny. Při návrhu nové databáze se musí myslet na snadnou propojitelnost struktury dat se strukturami ostatními souvisejících databází, jako například data z GIS systémů a podobně.

Jelikož se bude jednat o databázi dopravních dat pro zkoumání dopravních charakteristik, musejí se zohlednit veškeré faktory ovlivňující dopravní proud. Tyto faktory jsou v následující části přiblíženy, včetně zhodnocení jak mají být brány na zřetel při návrhu zamýšleného systému.

2.3.1 Faktory ovlivňující dopravní proud

Charakteristiky jako směrové a výškové vedení trasy značně ovlivňují dopravní proud. Pokud máme k dispozici data charakterizující dopravní proud z nějakého detektoru, je žádoucí mít možnost data propojit s daty o měřené trase, po které se vozidla pohybují. Pokud by byly třeba složitější geografické výpočty, nebo použití například makroskopických modelů, je vhodné použít GIS systémy, a data z výukové databáze

propojit s nějakou již existující databází dopravní sítě. K tomu musí odpovídat i struktura těchto dat.

Při zjednodušování silniční sítě většího území pro makroskopické modely se používá síť z uzlů a spojnic, které jsou upřesněny dalšími charakteristikami - uzel, například zeměpisnou polohou, která je klíčová pro párování s jakýmkoliv geografickými daty.

K datům z detektorů v zamýšleném systému by bylo vhodné mít možnost přiřadit alespoň polohu detektoru a z hlediska přehlednosti i směr dopravního proudu, který daný detektor detekuje.

Budou zavedeny profily, tedy skupiny detektorů spolu jakkoliv související.

2.3.2 Statistické veličiny

Pravidelným a podrobným ukládáním dat lze kdykoliv dopočítat přesné statistické veličiny, jako například padesátirázovou vlnu intenzity nebo roční průměr denních intenzit. Při uchování údajů o svátcích a dnech, které jim předchází či je následují, lze eliminovat nepravidelnosti a spolehlivě vypočítat variace dopravy pro dny, týdny i roky. V případě, že data budou neustále k dispozici (což je předpokladem) však není nutné, aby byly tyto statistické údaje generovány a ukládány automaticky.

2.4 Technické požadavky

2.4.1 Požadavky administrátora

Návrh nesmí obsahovat programátorské chyby a musí být funkční.

Musí být přidělena přístupová práva uživatelů. Například student skrze studentskou aplikaci bude smět zobrazit jen některé tabulky, pedagog bude smět skrze pedagogickou aplikaci přidávat studenty a zadání, administrátor a editor budou mít plný přístup.

Jedna z hlavních myšlenek, proč tvořit vlastní databázi, která je podobná již existující databázi TSK, je bezpečnost. Přístup k datům TSK má Fakulta Dopravní sjednaný, nicméně nechat zasahovat velké množství uživatelů (studentů) do databáze TSK by bylo

možné bezpečnostní riziko. Navrhovaná databáze FD bude do systému TSK přistupovat pravidelně, bez ohrožení neplánovaného zahlcení či jiného útoku.

Přístup do databáze FD bude umožněn pouze pro zaměstnance či studenty FD ČVUT a to jednotným přístupem ČVUT.

2.4.2 Požadavky editora

Aplikace bude stahovat data automaticky, a to včetně opakovaného vyžadování nezaslaných dat. To přinese úsporu času, zautomatizování monotónní lidské práce, převedení procesu do profesionálnější formy a v neposlední řadě zvýšení spolehlivosti.

O platnosti, respektive neplatnosti dat rozhodne aplikace také sama. Pravidla by měla jít definovat, a jejich definice snadno měnit.

Aplikace bude kontrolovat, zdali se neobjeví nový detektor, či typ detektoru. Bude se také ukládat přehled o stavech a chybách detektorů a jejich změnách.

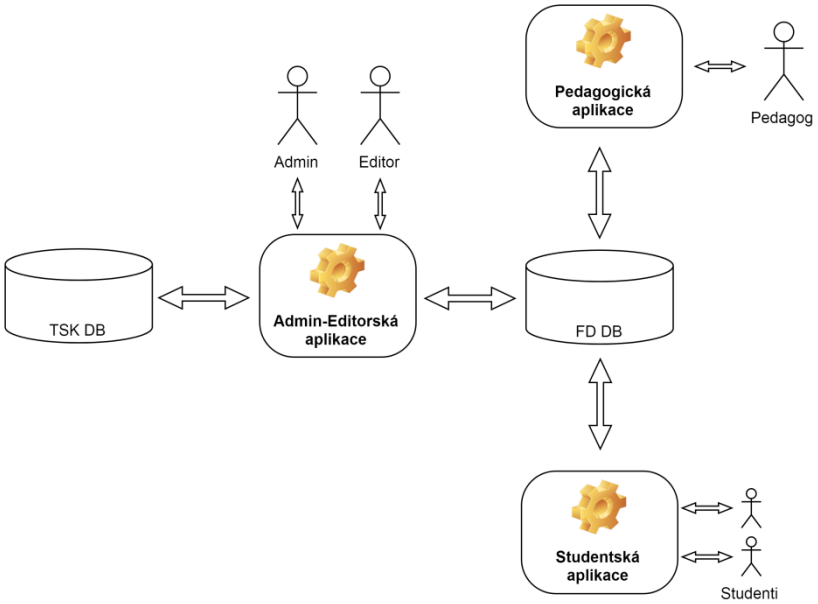
Pedagogická aplikace sama vyhodnotí zadání studentům, tudíž editorovi odpadne starost s pomocí výběru validních dat pro zadání, které tvoří pedagog.

Automatické stahování dat umožní stáhnout veškerá data, která systém může v danou chvíli nabídnout, takže editor nebude muset přemýšlet, od kterého druhu detektorů si může zpětně dovolit stáhnout dat.

3 Technická specifika návrhu

V této kapitole se čtenář seznámí s návrhem systému databáze a navazujících aplikací. Jádrem systému je relační databáze, na kterou navazují aplikace. Jedním z cílů tvorby tohoto systému je usnadnění lidské práce, přesněji práce s daty. Z toho důvodu vznikly aplikace podle skutečných lidských rolí - administrátora, editora, pedagoga a studenta. Vliv role dopravního odborníka je uplatněn jako dohled nad tvorbou aplikací a databáze, aby návrh byl správný po odborné stránce a samostatnou aplikaci nepotřebuje. Role administrátora a editora byla sloučena a zautomatizována v jednu aplikaci. Práva obou rolí jsou na podobně vysoké úrovni a oba musí přistupovat k systému opatrně, aby jej nepoškodili. Při správném návrhu se však nemusí téměř o nic starat a riziko, že systém poškodí nesprávným přístupem, je minimalizováno.

Přihlašování pro spuštění aplikace probíhá pomocí jednotného přihlašování ČVUT a technickými podrobnostmi tohoto subsystému se práce nezabývá. Do každé aplikace se přihlásí jen uživatel s daným přístupem. Co se týče přístupových práv do databáze, aplikace se tváří jako jeden uživatel. Schéma navrženého systému je zobrazeno na obrázku č. 17, uživatelé přistupují do aplikace k tomu určené.



Obrázek 17 - Nově navržené schéma uživatelů; zdroj: archiv autora

3.1 Návrh relační databáze

Nutnost vlastní databáze plyne z technických požadavků na návrh tohoto systému. V této podkapitole se čtenář seznámí se základními předpoklady návrhu, metodikou návrhu a samotným návrhem v podobě schématu databáze, popisu použitých proměnných a popisu tabulek.

3.1.1 Zadání

Základní předpoklad pro výběr databázového systému bylo nekomplikované licencování softwaru, a proto bylo nutné zvolit některý z volně dostupných. Po konzultaci s IT odborníkem Ing. Michalem Kovaljovem z Ústavu dopravní telematiky, jakožto potenciálním nastupujícím vývojářem zamýšleného komplexního systému, byl na jeho doporučení zvolen software PostgreSQL.

3.1.2 Metodika návrhu

Návrh databázového schématu probíhal podle uživatelských požadavků, běžných konvencí pro návrh databáze a se snahou dodržet co nejvíce normálních forem.

Nejprve bylo databázové schéma na obrázku č. 18 navrženo v uživatelsky přívětivém volně dostupném online prostředí dbdiagram.io, a poté převedeno do prostředí PostgreSQL v programu pgAdmin4. Vytvořený SQL kód systému tabulek lze nalézt v příloze B.

3.1.3 Přehled použitých typů proměnných

Typ proměnné varchar

Z anglického názvu typu proměnné (varying character) vyplývá, že se jedná o řetězec znaků, jehož délku lze omezit parametrem udávaným v kulatých závorkách za názvem typu proměnné pro menší velikost. Doporučuje se používat i pro číselné kódy, se kterými se nevykonávají matematické operace. V tomto případě například identifikační číslo studenta.

Typy proměnných int a smallint

Tyto typy jsou používány pro reprezentaci číselných hodnot. V případě typu integer může nabývat hodnot -2 147 483 648 až +2 147 483 647 a jedna hodnota využívá paměť o velikosti 4 byte. V případě typu small integer je možný rozsah hodnot od -32 768 do +32 767 a jedna hodnota využívá paměť o velikosti 2 byte.

Typy proměnných serial a bigserial

Tyto typy jsou také používány pro reprezentaci číselných hodnot, avšak pouze kladných. V případě typu serial může nabývat hodnot 1 až +2 147 483 647 a jedna hodnota využívá paměť o velikosti 4 byte. V případě typu big serial je možný rozsah hodnot od 1 do 9 223 372 036 854 775 804 a jedna hodnota využívá paměť o velikosti 8 byte. Využívá se především pro identifikátory (primární klíče) z důvodu snazší podpory funkce autoincrement, tedy automatické zvyšování hodnoty následujícího záznamu o 1.

Typ proměnné bool

Proměnná typu boolean nabývá pouze dvou hodnot, a to hodnoty TRUE či hodnoty FALSE. Jedna hodnota využívá paměť o velikosti 1 byte.

Typy proměnných doubleprecision a real

Tyto typy proměnných slouží k ukládání číselných hodnot s desetinnou čárkou, tedy alespoň racionálních čísel. Základní typ real zabírá 4 byte paměti a přesnost takového čísla je na 6 desetinných míst. Při potřebě větší přesnosti je vhodné použít datový typ doubleprecision. Ten zabírá 8 byte paměti a jeho přesnost dosahuje 15 desetinných míst.

Typ proměnné timestamptz

Pro reprezentaci času a dne najednou se používá datový typ timestamptz. Při používání kontinuálních dat je velmi vhodné jej používat i s časovým pásmem. Databáze má nastavené výchozí časové pásmo, které se při změně času automaticky mění. V tomto výchozím pásmu si data uchovává a při vkládání data a času v jiném časovém pásmu si databázový server automaticky čas přepočte do svého výchozího pásma. Například tedy při vložení času '2018-08-24 14:00:00 +02:00' během pásma SEČ (+01:00) uloží hodnotu jako '2018-08-24 13:00:00 +01:00'.

Typ proměnné interval

Proměnná typu interval nevyjadřuje jeden konkrétní časový okamžik, ale dobu mezi dvěma časovými okamžiky. Formát může být následující: 'P 1Y2M3D T 4H5M6S'.

Znak P označuje začátek části, která vyjadřuje datum a znak T označuje začátek části, která vyjadřuje čas. Znaky Y, M, D označují rok, měsíc a den, znaky H, M, S hodiny, minuty a vteřiny. Čísla před konkrétním znakem vyjadřují počet těchto jednotek.

3.1.4 Tabulky

Obrázek 18 - Navržené databázové schéma; zdroj: archiv autora

Databázové schéma

Na obrázku č. 18 je zobrazeno databázové schéma, které obsahuje 13 vzájemně propojených tabulek a jednu tabulku bez vazeb, která charakterizuje jednotlivé dny z hlediska státních svátků. Tabulky jsou podrobně popsány v následující části.

Tabulka typů detektorů

Tabulka s názvem detectortypes obsahuje jednotlivé typy detektorů včetně údajů, které se k nim vztahují. V době návrhu systém obsahuje jen 6 typů detektorů. Nicméně užitečnost tabulky spočívá v parametrech týkajících se vlastností, které mají všechny detektory stejného typu společné. Také je zde možnost výskytu nového typu detektoru v systému s odlišnými vlastnostmi. Evidence nového typu je díky této tabulce snazší.

Primárním klíčem této tabulky je sloupec "type", který popisuje právě typ detektoru, který je ostatními sloupci specifikovaný. Typy detektorů jsou identifikovány textovým řetězcem varchar s omezením délky na 10 znaků, neboť nejdelší současné názvy typů detektorů dosahují délky 8 znaků.

První typickou charakteristikou typu detektoru je doba intervalu, ve kterém jsou data pravidelně opakovaně měřena. Je to tedy nejmenší časové rozlišení dat. U dosavadních detektorů se jedná o dvě specifické hodnoty, a to 5 minut u strategických detektorů a 90 vteřin u detektorů křižovatkových. Tato skutečnost je zanesena ve sloupci s názvem "RawIntervalSeconds" datovým typem interval, který je povinnost vyplnit. Zároveň není do sloupce možné vložit interval delší než 1 hodina.

Další podobnou charakteristikou je zpětná časová dostupnost surových dat. U jednotlivých typů detektorů se liší, proto je udávána ve sloupci "accesibility", rovněž datovým typem interval. Tato proměnná hraje významnou roli při opakovaném dotazování po chybějících datech, aby se předešlo požadavkům, které nelze splnit.

Poslední charakteristikou spojovanou s typy detektorů je jejich klasifikování. Některé typy detektorů dokáží měřit dopravní veličiny pro různý typ vozidel. Jedná se tedy jen o textový popis této klasifikace uvedený ve sloupci "ClassDescription" s omezením na 100 znaků. Tato podmínka nijak neomezuje kreativitu popisu klasifikace a vzhledem k dosavadnímu počtu šesti záznamů v tabulce není ani zatěžující na paměť databáze.

Tabulka detektorů

Tabulka s názvem detectors obsahuje jednotlivé detektory včetně údajů, které se k nim vztahují.

Jejím primárním klíčem je sloupec "id_det", ve kterém jsou obsaženy ID detektorů. Tento sloupec je navržen jako datový typ Varchar s omezením na 30 znaků. Nejdelší ID dosud známého detektoru čítá délku 27 znaků. Nepředpokládá se, že by nějaký nový detektor mohl mít delší ID, než právě navržených 30 znaků.

Každý detektor musí mít uvedený typ, který je uvedený ve sloupci "type", což je překopírovaný (cizí) klíč z tabulky types. Vyplnit tento sloupec je povinné.

Další detektory popisující parametr je aktivace. Je-li ve sloupci "active" uvedena hodnota TRUE, znamená to, že aplikace, která automaticky stahuje data, tento detektor bere v úvahu. Pokud by byla hodnota rovna FALSE, byl by detektor pro automatické stahování dat deaktivován. Tento sloupec se nesmí rovnat hodnotě NULL.

Zeměpisná poloha detektoru je evidována dvojicí parametrů "lat" a "lon" datového typu real. Datový typ real má přesnost 6 desetinných míst a při zobrazení WGS84 v oblasti Prahy znamená toto rozlišení cca 11 cm zeměpisné šířky a přibližně 7 cm zeměpisné délky, což je pro tyto účely naprosto dostačující. Podle znaménkové konvence souřadnic se uvádí severní šířka a východní délka jako kladná hodnota, jižní šířka a západní délka jako záporná. V případě Prahy tedy obě veličiny kladné.

Pro stanovení směru dopravního proudu je vhodné uvést azimut, uložený ve stejnojmenném sloupci jako integer s omezením od 0 do 360.

Polohu detektoru lze také popsat slovně do stoznakového varchar sloupce "PositionDescription".

Při zkoumání dat rychlosti by mohla být užitečná informace o maximální dovolené rychlosti v místě detektoru, pro kterou je připraven sloupec "SpeedLimit" datového typu smallint.

Tabulka profilů

Profily jsou seskupení detektorů, které mají společné místo. Může to být například rameno křižovatky, společná signální skupina na křižovatce, celá křižovatka či strategické detektory jednoho jízdního pásu (více jízdních pruhů).

Primárním klíčem je sloupec "id_profil" datového typu serial, dále je zde název profilu ve sloupci "ProfilName" jako datový typ varchar o délce 50 znaků.

Tabulka sekcí

Pro propojení dat z detektorů s ostatními datovými zdroji slouží tabulka sekcí neboli úseků. Profily mohou mít několik identifikátorů pro sčítání dopravy různých organizací. Například v Praze sčítá dopravu TSK a identifikované mají křižovatky, hodnoty průměrných intenzit jsou tedy charakterizované počátečním bodem (křižovatkou) a koncovým bodem (křižovatkou). Podobně fungují i RDS-TMC úseky, na kterých sčítají dopravu například mobilní operátoři. V celostátním sčítání dopravy prováděné ŘSD jsou dopravní charakteristiky vztažené k úseku identifikovaným číslem úseku. Tato tabulka tedy zohledňuje veškeré možné způsoby identifikace daného úseku.

Vedle sloupce primárního klíče "id_section" datového typu serial a vloženého cizího klíče "id_profil" jsou zde ještě čtyři sloupce datového typu varchar délky 20 znaků. Sloupec "SectionType" uvádí typ značení úseku, sloupce "SectionStartID" a "SectionEndID" uvádí název počátečního a koncového bodu, sloupec "SectionID" uvádí název úseku s tím, že musí být vyplněn buďto název počátečního a koncového bodu nebo název úseku.

V případě popisu profilu vjezdových detektorů v křižovatce se jako úsek počítá vjezdové rameno.

Tabulka přiřazení detektorů do profilů

Tabulka "det2profil" je vazební tabulkou mezi tabulkami detektorů a profilů, které mají relaci m:n.

Najdeme zde primární klíč "id_det2profil" a vložené cizí klíče "id_det" a "id_profil".

Tabulka dat

Nejobsáhlejší a nejpodstatnější tabulkou je tabulka dat. Zde se jako jeden řádek ukládají dopravní charakteristiky jednoho detektoru za jeden časový interval, a to pro všechny typy detektorů. Pokud by se stahovala kontinuálně surová data ze všech dostupných detektorů, přibylo by zde cca 5 808 672 záznamů denně (dostačující pro 4,3 mld let). Proto je primárním klíčem "id_dat" proměnná datového typu bigserial, zatímco datový typ serial by byl vyčerpán po jednom roce.

Pak je zde cizí klíč "id_det" a sloupec "raw" typu bool, který uvádí, zda se jedná o surová data (TRUE) nebo data agregovaná (FALSE).

Který typ detektorů ukládá jaká data, lze vidět na obrázku č. 19.

typ detektoru	sloupce databáze																							
	id_dat	id_det	raw	confidence	validity	attemptnumber	stoptime	starttime	count_all	count_passengercar	count_lighttruck	count_heavytruck	count_alltruck	speed_all	speed_passengercar	speed_alltruck	occupancy_all	count1 až 13	speed1 až 13	weight1 až 13	weight_all	occupancy1 až 13	temperature	voltage
SCALA	A	A	A	N	A	A	A	A	A	N	N	N	N	A	N	N	A	N	N	N	N	N	N	N
CollectR	A	A	A	N	A	A	A	A	A	A	N	N	A	A	A	A	A	N	N	N	N	N	A	A
Traficam	A	A	A	N	A	A	A	A	A	N	N	N	N	N	N	N	A	N	N	N	N	N	N	N
WIM	A	A	A	A	A	A	A	A	A	N	N	N	N	A	N	N	A	A	A	A	A	A	N	N
MUR	A	A	A	A	A	A	A	A	A	A	A	A	N	A	N	N	A	N	N	N	N	N	N	N
SDDU	A	A	A	A	A	A	A	A	A	A	A	A	N	A	N	N	A	N	N	N	N	N	N	N
MP	A	A	A	A	A	A	A	A	A	A	A	A	N	A	N	N	A	N	N	N	N	N	N	N

Obrázek 19 - Ukládaná data do tabulky dat dle typu detektoru; zdroj: archiv autora

Sloupec "confidence" typu smallint má vyjadřovat důvěryhodnost dat vypočítanou systémem TSK. Tato charakteristika je dostupná v datech detektorů typu WIM, MUR, SDDU a MP a dosahuje hodnot 0 až 100.

Sloupec "validity" typu smallint vyjadřuje rovněž věrohodnost dat a podobou je inspirovaný předešlým sloupcem. Narozdíl od něj je však vypočítáván navrhovanou aplikací.

Pro záznam pořadí pokusu opakovaného stahování dat slouží sloupec "attemptnumber" typu smallint, který nabývá hodnot 0 až 5. Podrobnosti o této hodnotě lze nalézt v kapitole 3.2.4.1 v popisu procedury download_raw.

Začátek a konec časového úseku, ke kterému data platí, vyjadřují sloupce "starttime" a "stoptime" datového typu timestamptz.

Následují charakteristiky typu smallint, jedná se o intenzity (count), rychlosti (speed), obsazenost detektoru (occupancy), buďto souhrnně (all) pro automobily osobní (passengercar), lehké nákladní (lighttruck), těžké nákladní (heavytruck) anebo nákladní souhrnně (alltruck) podle obrázku č. 19.

U detektorů typu WIM se vyskytuje ještě hmotnost (weight) typu int a všechny charakteristiky jsou rozděleny do 13 kategorií.

U detektorů typu CollectR je ještě navíc uvedena teplota v rozvaděči a napětí záložní baterie ve sloupcích "temperature" a "voltage" datového typu real.

Tabulka zadání úloh

Tabulka "task" je vazební tabulkou tabulek "profiles" a "students", které jsou v relaci m:n. sloupec "id_task" typu smallint je primárním klíčem, cizí klíče jsou "id_stu" a "id_profil".

Upřesňujícími parametry zadání je sloupec "rok" typu smallint v rozmezí 2019 až 2050 a číslo týdne uvedené ve sloupci "tyden" typu smallint v rozmezí 1 až 52.

Zhodnocení kvality dat bude pedagogickou aplikací ukládáno do sloupce integrity typu smallint v rozmezí 0 až 100.

Tabulka studentů

V tabulce "students" budou uloženy osobní údaje studentů. Patří mezi ně osobní číslo studenta ve sloupci "id_stu" typu varchar s ověřením délky přesně 6 znaků, coby primární klíč, jméno a příjmení uložené ve sloupcích "FirstName" a "Surname" obě typu varchar.

Tabulka evidence dotazů

V tabulce "querylog" se nachází primární klíč zápisu "id_log" typu serial, vložený cizí klíč "id_stu", čas zápisu "logtime" typu timestamptz a varchar "query" pro uložení zadaného dotazu.

Tabulky seznam stavů a seznam chyb

Detektory posílají informace o svých stavech a chybách, což jsou velice podobné entity. Nabízelo by se sice shrnout stavy a chyby do jedné tabulky, nicméně překrývají se ve svých číselných id, uvedených i v návodu k protokolu T3 Remote Client, a právě proto budou udržovány v oddělených tabulkách "statuslist" a "errorlist".

V tabulce lze tedy nalézt jejich identifikátory "id_status"/"id_error" a jejich popis datového typu varchar o délce 100 znaků.

Všechny dosud známé stavy a chyby jsou uvedeny v manuálu T3 Remote client [zdroj č.5, str. 13-15].

Tabulky historie stavů a chyb detektorů

Pro uchování přehledu stavů a chyb jsou navrženy tabulky "statushistory" a " errorhistory", jenž jsou vazební tabulky mezi tabulkami seznamů stavů/chyb a tabulkou detektorů.

Zápis do nich provádí automatické procedury administrátorsko-editorské aplikace. Kromě primárního klíče "id_statuslog"/"id_errorlog" typu bigserial a cizích klíčů "id_det" a "id_status"/"id_error" jsou charakterizovány ještě začátkem "starttime" a koncem "stoptime" časové platnosti stavu/chyby.

V případě stavů je ještě uvedena hodnota "valuestatus" typu bool, neboť existují jisté stavy i s hodnotou. Příkladem prostého stavu bez hodnoty může být například stav "chyba spojení", příkladem stavu s hodnotou může být například stav "dveře rozvaděče otevřeny", který má hodnoty 0 nebo 1, která rozhoduje o pravdivosti názvu stavu. Pokud jsou tedy dveře zavřeny, vypadá status například následovně: status id_status = '3', valuestatus = FALSE, přitom v tabulce statuslist je statusDescription = 'Dvere otevreny'.

Tabulka charakterů dnů

Pro určení validity je důležitá tabulka charakterů dnů. V ní jsou evidovány jednotlivé dny, a informace, zda se nejedná o státní svátek či den navazující/předcházející.

Primární klíč "id_day" typu smallint, následují sloupce "StartDay" a "StopDay" pro určení začátku a konce dne v používaném datovém typu timestamptz.

Samotný charakter dne "DayCharacter" typu varchar o délce 12 znaků dává na výběr ze 4 hodnot typu dne. Hodnota 'holiday' pokud se jedná o státní svátek, hodnota 'preholiday' pokud se jedná o den předcházející státním svátku, hodnota 'postholiday' pro den následující po státním svátku a hodnota 'regular' pro běžný nesváteční den. Je také uveden den týdne ve sloupci "WeekDay" typu smallint s hodnotami 1 až 7 od pondělí do neděle.

Ve sloupci "WeekNum" typu smallint s hodnotami 1 až 53 se uvádí číslo týdne daného dne, který vychází z normy ISO 8601. Na přelomu roku dochází k situacím, kdy číslování vychází z předešlého nebo následujícího roku. Například 31. prosinec 2019 patří do týdne číslovaného jako 1. Přestože jde o rok 2019, číslování již platí pro rok 2020. Pro evidenci roku, patřícího k číslování týdne slouží sloupec "WeekNumYear" typu smallint.

Ukázka seznamu charakterů dnů včetně pravidel pro jeho vytvoření v MS Excel je k nalezení v příloze C. V této příloze je také návod pro vytvoření SQL příkazů pro nahrání tohoto seznamu.

3.2 Návrh administrátorsko-editorské aplikace

Nejpodstatnější část automatizace již zmíněných funkcí zastane aplikace, která pomáhá administrátorovi z hlediska plnění databáze a editorovi z hlediska kontroly dat. Tyto dvě role spolu úzce souvisí a nepředpokládá se, že by administrátor a editor měli mít různá přístupová práva, a je tedy zcela zbytečné aplikaci rozdělovat na dvě části podle uživatelských rolí.

Jediné rozdělení spočívá v tom, jak se aplikace chová a program spouští. Je zde automatická část, která má fungovat bez zásahu editora či administrátora, ale nesmí být opomenuty funkcionality manuální, pro cílené zásahy.

3.2.1 Zadání

První krok k této zamýšlené aplikaci vedl k již napsané a používané aplikaci T3 Remote Client. Ta stáhne XML zprávu podle zadaných parametrů a je napsána v jazyce C#. Není

potřeba, a bylo by až zbytečné, měnit programovací jazyk, je-li část již naprogramována, odladěna a používána.

3.2.2 Metodika návrhu

Vývoj programu

Přes poněkud evidentní záměr programovat aplikaci v jazyce C# se autor zabýval vývojem aplikace v obecnější rovině. Cílem práce není naprogramovat odladěný program, nýbrž shrnout principy a procesy, které by žádaná aplikace měla vykonávat. Návrh tedy bude využívat vývojové diagramy podle běžné konvence.

Pro práci s XML daty je využit adresovací jazyk XPath, který je standardizovaný organizací W3C. Byl autorem zvolen pro svou jednoduchost a vhodnost k tomuto účelu.

Vývojové diagramy

Vývojové diagramy byly navrhovány ve webové aplikaci draw.io, která je volně dostupná.

Koncepce vývojových diagramů spočívá v různých tvarech a směru zpracování. Oválný tvar značí začátek a konec podprogramu (znázorněném na jednom vývojovém diagramu), obdélník dílčí krok, kosodélník import či export dat, kosočtverec bod rozhodování a kolečko sloučení více šipek směru zpracování.

Pro zjednodušení, výše uvedená pravidla neplatí striktně. Například v bodě rozhodování se mohou načítat proměnné či dělat výpočty. To značně vývojové diagramy zjednoduší a dosáhne se tím větší přehlednosti.

Pro lepší orientaci, co ve vývojových diagramech představuje proměnnou aplikace, funkci či příkazy XPath a SQL, bylo zvoleno barevné rozlišení patrné z legendy na obrázku č. 20.

Obrázek 20 - Legenda k vývojovým diagramům; zdroj: archiv autora

Bližší popis vývojových diagramů

Z důvodu přehlednosti byly dlouhé dotazy a výpočty nahrazeny symbolem křížku s číslem (např.: #1), který zastupuje delší část popisu. Tyto zástupné symboly jsou uvedeny v příloze D.1.

Názvosloví částí programu

V rámci návrhu byly popsány dílčí části programu pomocí pojmů "procedura", "podprocedura" a "funkce".

Procedurou autor rozumí podprogram, který může mít vstupní parametry, ale nemá výstupní parametry. Díky vstupním parametrům se může zavolat z libovolného místa programu.

Podprocedura je součástí procedury, ale nemůže se zavolat libovolně - pouze v nadřazené proceduře z důvodu konkrétně načtených proměnných aplikace. Při zavolání mimo nadřazenou proceduru by nefungovala správně. Také nevrací žádnou hodnotu. Jde v podstatě o část nadřazené procedury pro zjednodušení do vývojového diagramu, kde podprocedura počítá s proměnnými aplikace inicializovanými ještě před zavoláním podprocedury, tedy v nadřazené proceduře.

Funkce je samostatná stejně jako procedura, pouze vrací nějaké hodnoty, tedy výstupní parametry.

3.2.3 Některé použité základní funkce

V návrhu jsou použity některé základní funkce, především matematické či časové.

Funkce min a max vyberou a vrátí z pole vstupních parametrů minimum či maximum.

Funkce zaokrouhlit_dolu a zaokrouhlit_nahoru zaokrouhlují vstupní parametr na celé číslo, popřípadě celou časovou jednotku.

Funkce aktuální čas je bez vstupního parametru a vrací aktuální čas.

Funkce letní čas vrací proměnnou typu boolean o pravdivosti, zda v datu a času uvedené pod vstupním parametrem této funkce platí či neplatí časové pásmo SELČ UTC+2. Pokud není vstupní parametr zadán, je myšlen aktuální čas.

Funkce download data vychází ze současné verze klienta T3 Remote Client.

3.2.4 Automaticky vykonávané funkce

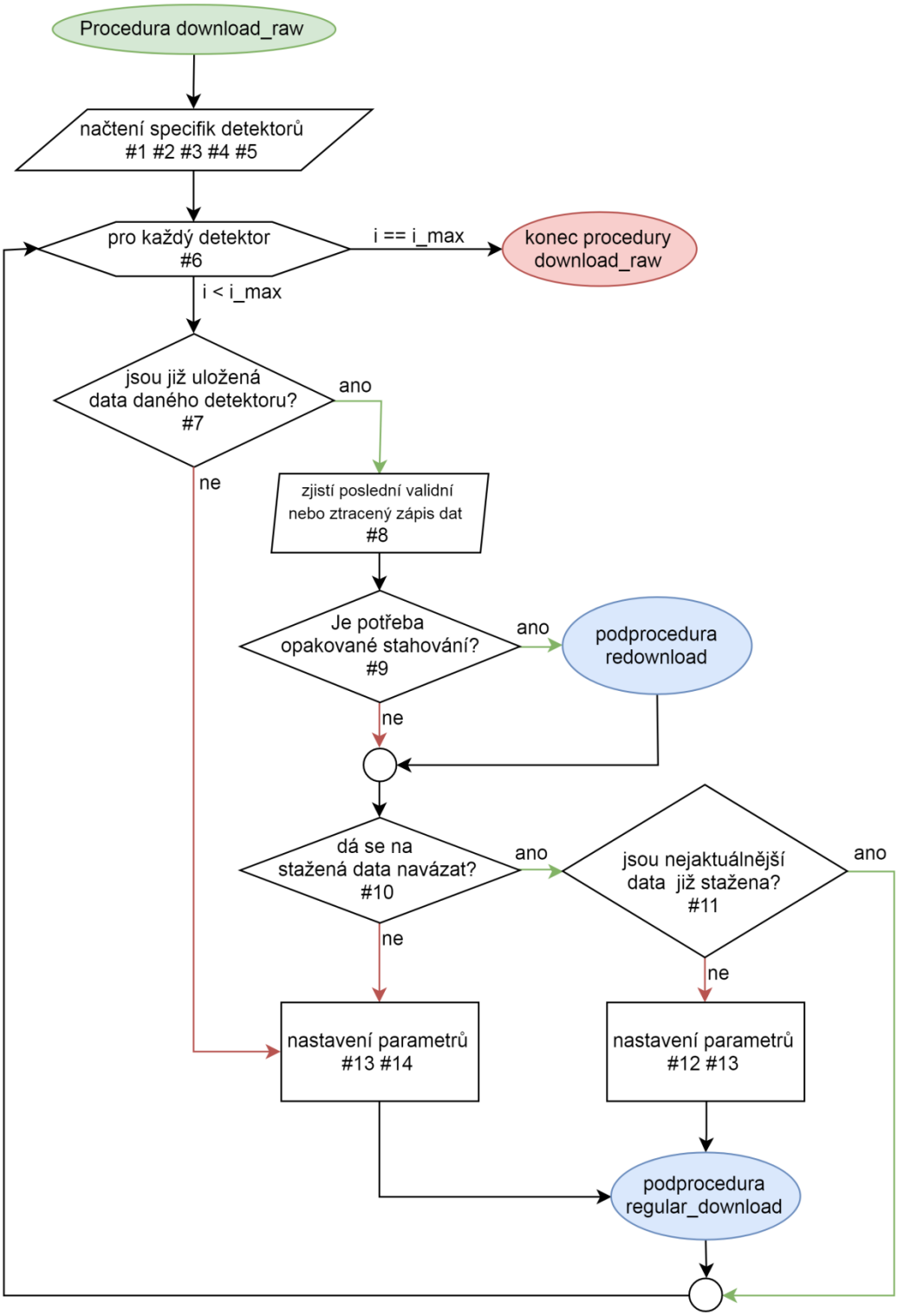
Stěžejní části automaticky vykonávaného programu jsou procedury "download_raw" a "echo", jejichž popisu se věnuje následující část práce.

Procedura download_raw

Jednou ze dvou hlavních procedur administrátorsko-editorského klienta je metoda download_raw, jejímž hlavním cílem je řídit opakované a řádné stahování.

Jedná se o automatické plnění dat, která jsou v případě neplatnosti znovu vyžadována zpětně při dalším spuštění procedury, a to opakovaně v několika pokusech. Počet pokusů byl expertním odhadem stanoven na pět. Počet pokusů by měl být volitelný, avšak jeho stanovení musí ukázat testování. Příliš vysoký počet pokusů opakování by mohl zbytečně zpomalovat program, při příliš nízkém počtu pokusů hrozí ztráta dat. Na následujícím diagramu obrázku č. 21 lze vidět algoritmus této procedury.

Po načtení seznamu aktivních detektorů se aplikace po jednom detektoru rozhoduje, zda je potřeba opakovaně stahovat některá data, nebo může přistoupit ke stahování řádnému. V tomto případě ještě zvažuje, zda nějaká data stáhnout vůbec může, a podle toho nastavuje parametry pro stahování. Před pravidelným stahováním dat zde ještě probíhá inicializace parametrů.



Obrázek 21 - Vývojový diagram procedury download_raw; zdroj: archiv autora

Podprocedura redownload

V případě, je-li potřeba se opakovaně pokusit o stažení chybných či chybějících dat, program vstoupí do části redownload. V kroku #20 zjistí časy chybějících dat, podle kterých nastaví parametry pro stahování. Samotné stahování poté probíhá najednou, a poté se po jednotlivých intervalech kontroluje a nahrává do databáze.

Podprocedura regular_download

Vstupní parametry pro stahování byly již inicializovány v nadřazené proceduře download_raw, a tato část programu začíná téměř rovnou stahováním. Nutná korekce označená symbolem #30 se týká pouze detektorů typu SCALA. Korekce tak řeší nedostatek popsaný v kapitole 1.3.3 pomocí obrázků č. 11 a 12.

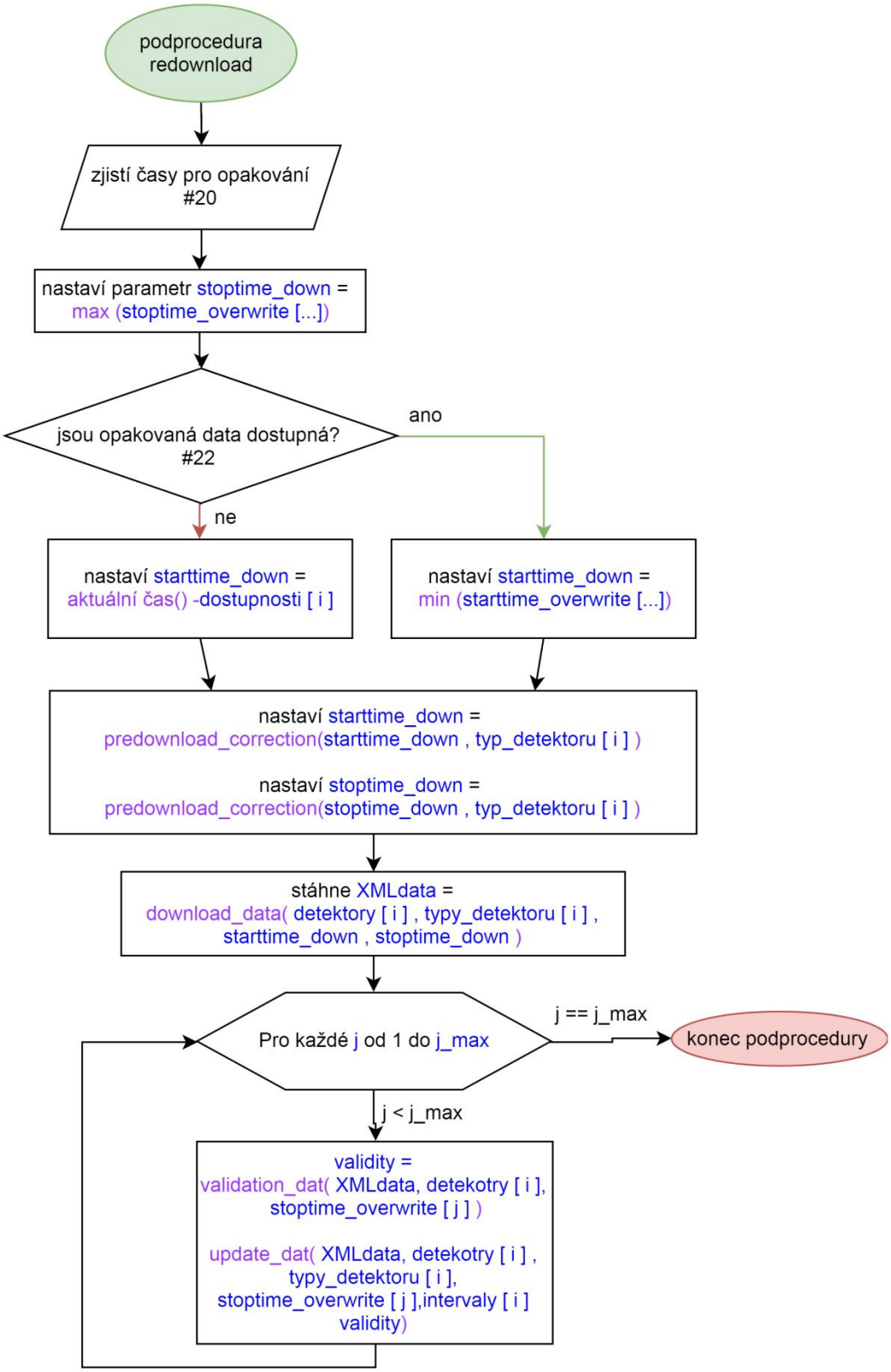
Nadále se kontrolují časové posuny, čímž můžeme rozumět první čas platnosti dat po půlnoci, který je menší než interval, ve kterém se data posílají. Je zde i část programu, která sama opraví tento časový posun v případě změny.

Poté se po jednotlivých paketech (údajích jednoho detektoru k jednomu časovému údaji) kontrolují a nahrávají data do databáze.

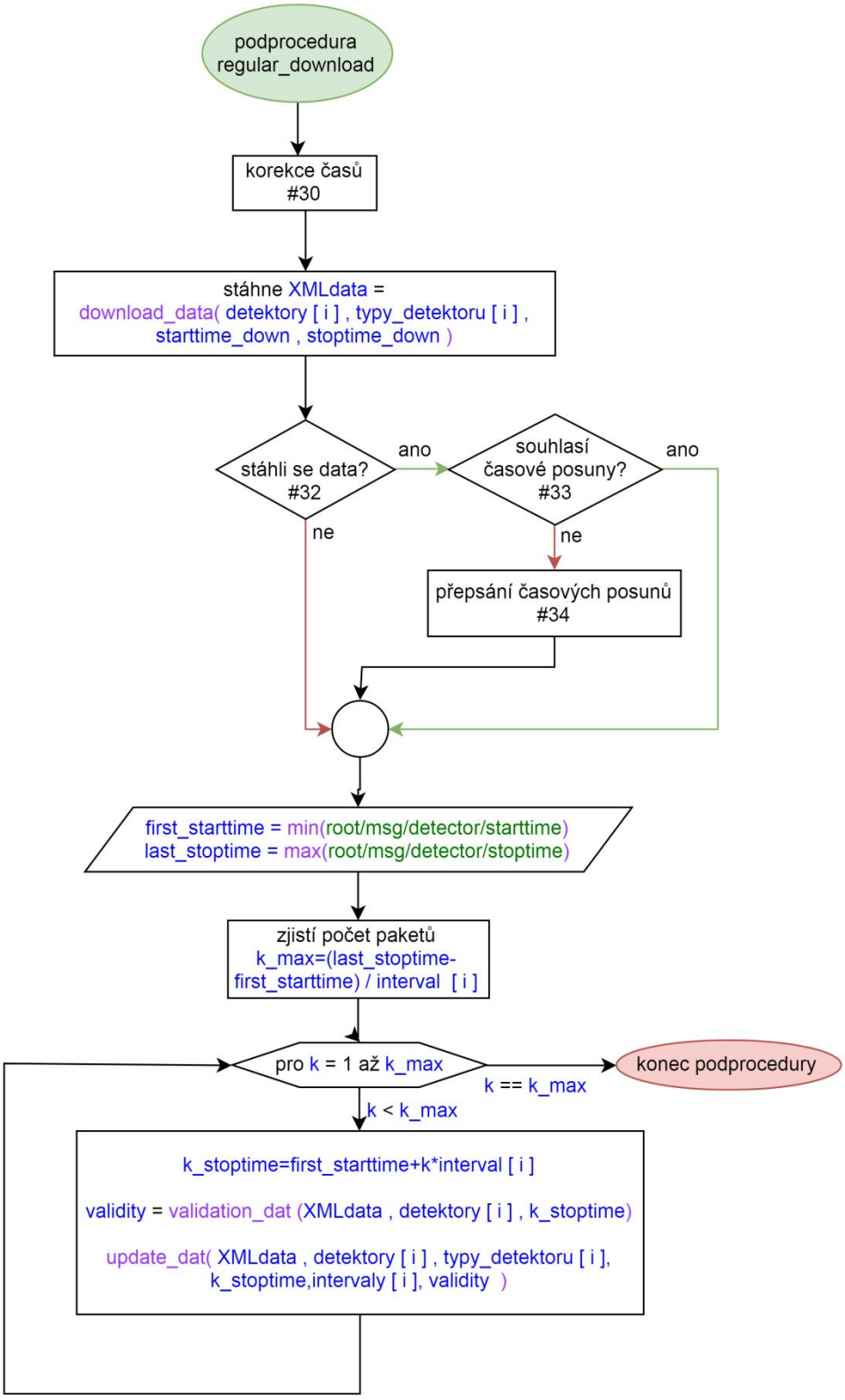
Procedura echo

Jedná se o druhou podstatnou část programu, jejíž účel je kontrolovat, zda v aplikaci T3 Remote Client nepřichází data od případného nového detektoru. Kdyby ano, je žádoucí o takovémto novém detektoru vědět, upřesnit jeho vlastnosti a co nejdříve ho zařadit do procesu běžného stahování. To aplikace udělá pomocí zprávy administrátorovi a editorovi, kteří mohou detektor ručně aktivovat.

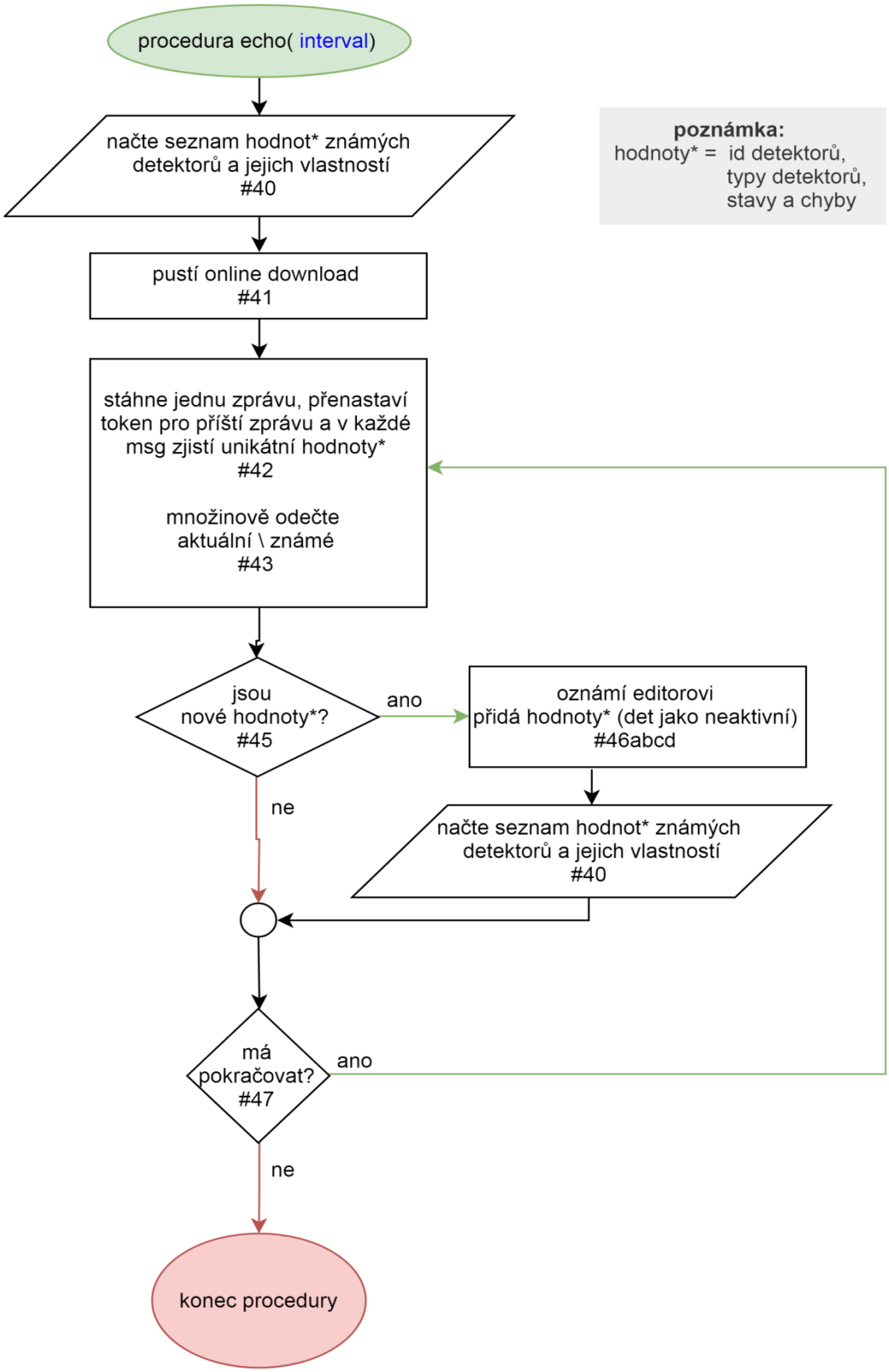
Při kontrole nových detektorů aplikace rovnou kontroluje i případný nový typ detektorů, nové stavy a chyby detektorů.



Obrázek 22 - Vývojový diagram podprocedury redownload; zdroj: archiv autora



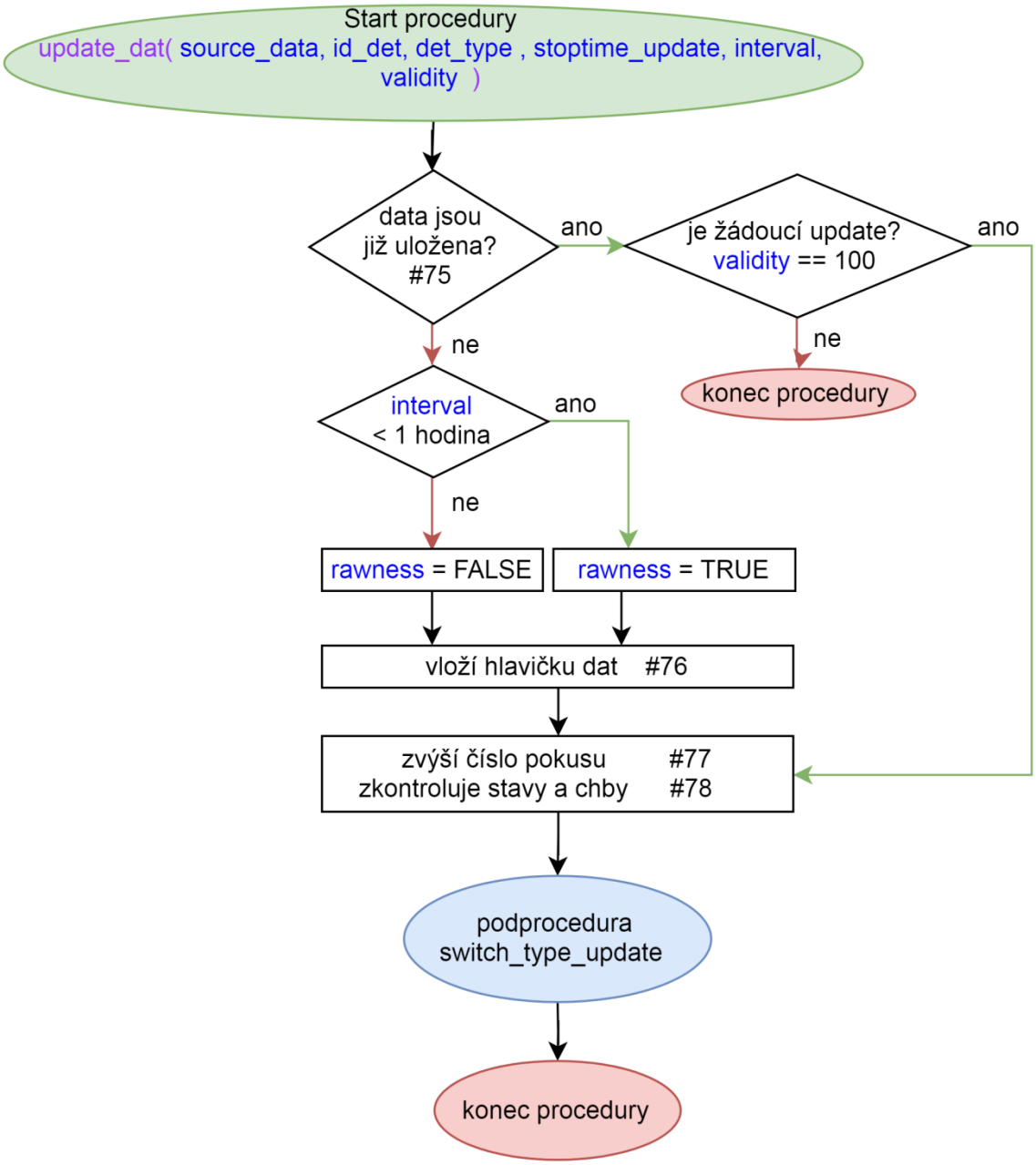
Obrázek 23 - Vývojový diagram podprocedury regular_download; zdroj: archiv autora



Obrázek 24 - Vývojový diagram procedury echo; zdroj: archiv autora

Procedura update_dat

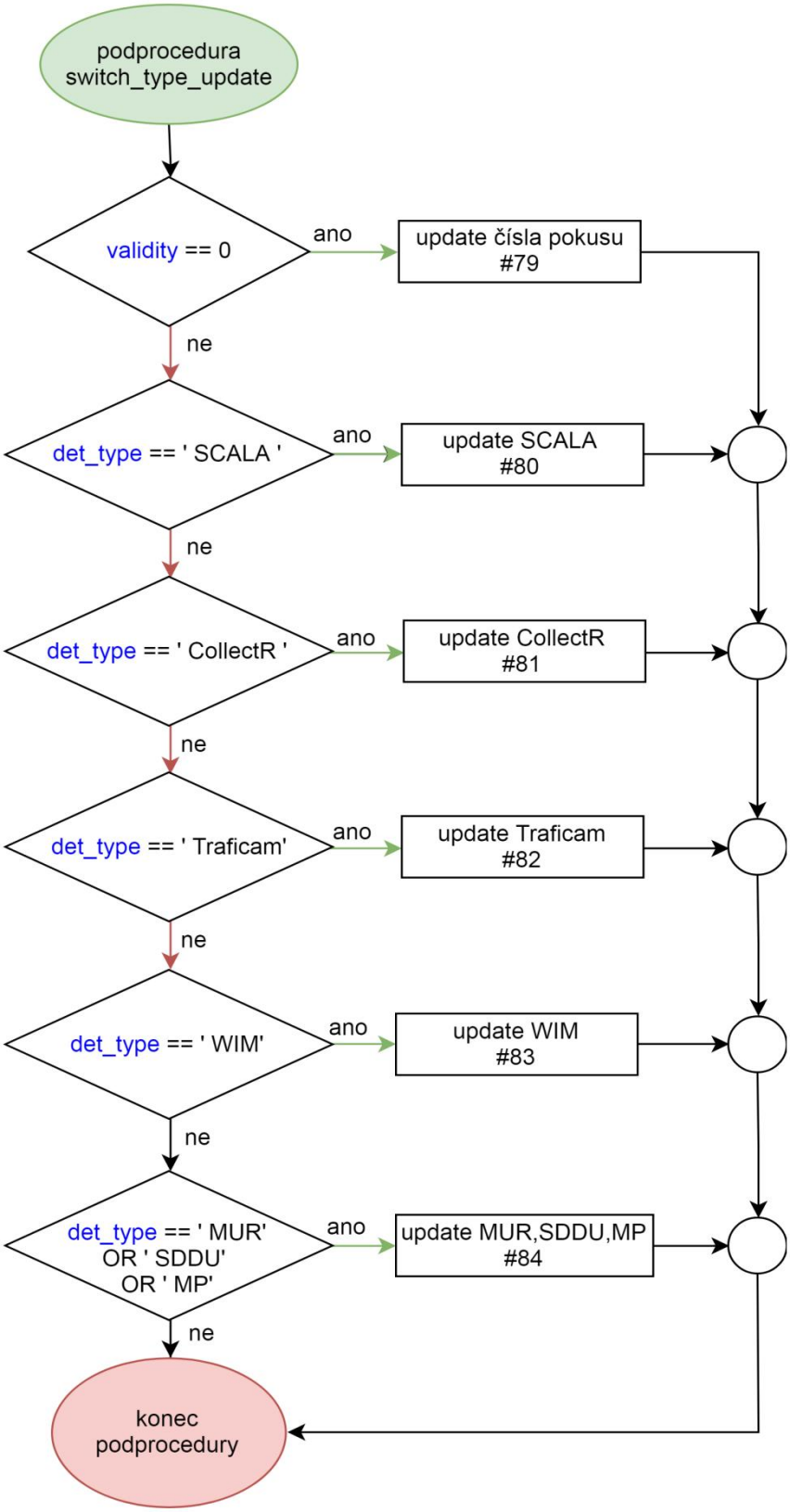
Procedura update_dat slouží k samotnému těžení dat z XML a jejich vkládání do databáze. Děje se tak opět po jednom paketu (informace k jednomu detektoru s jednou časovou platností). Nejprve aplikaci zajímá, zda jsou již data uložena v databázi, a pokud ano, je-li je potřeba obnovovat. Pokud data databáze neobsahuje, při prvním vkládání se vkládá i hodnota rawness, což je hodnota, která je pravdivá, pokud se jedná o surová data. Při každém zavolání této procedury se také zvyšuje až do maxima číslo pokusu obnovení dat. Také se zde kontrolují stavy a chyby detektorů. Poté již probíhá výběr dat a jejich nahrání do databáze, a to v podproceduře switch_type_update, která rozděluje úkony podle typu detektoru.



Obrázek 25 - Vývojový diagram procedury update_dat; zdroj: archiv autora

Podprocedura switch_type_update

Zde se program rozděluje podle typu detektoru. Každý typ detektoru udává jiná data, o čemž vypovídá tabulka sloupců databáze dle typu detektoru na obrázku č. 19 v kapitole 3.1.4. První rozhodovací bod rozhoduje, zda jsou data validní - v případě, že ne, pouze zvýší číslo pokusu a podproceduru včetně nadřazené procedury ukončuje.

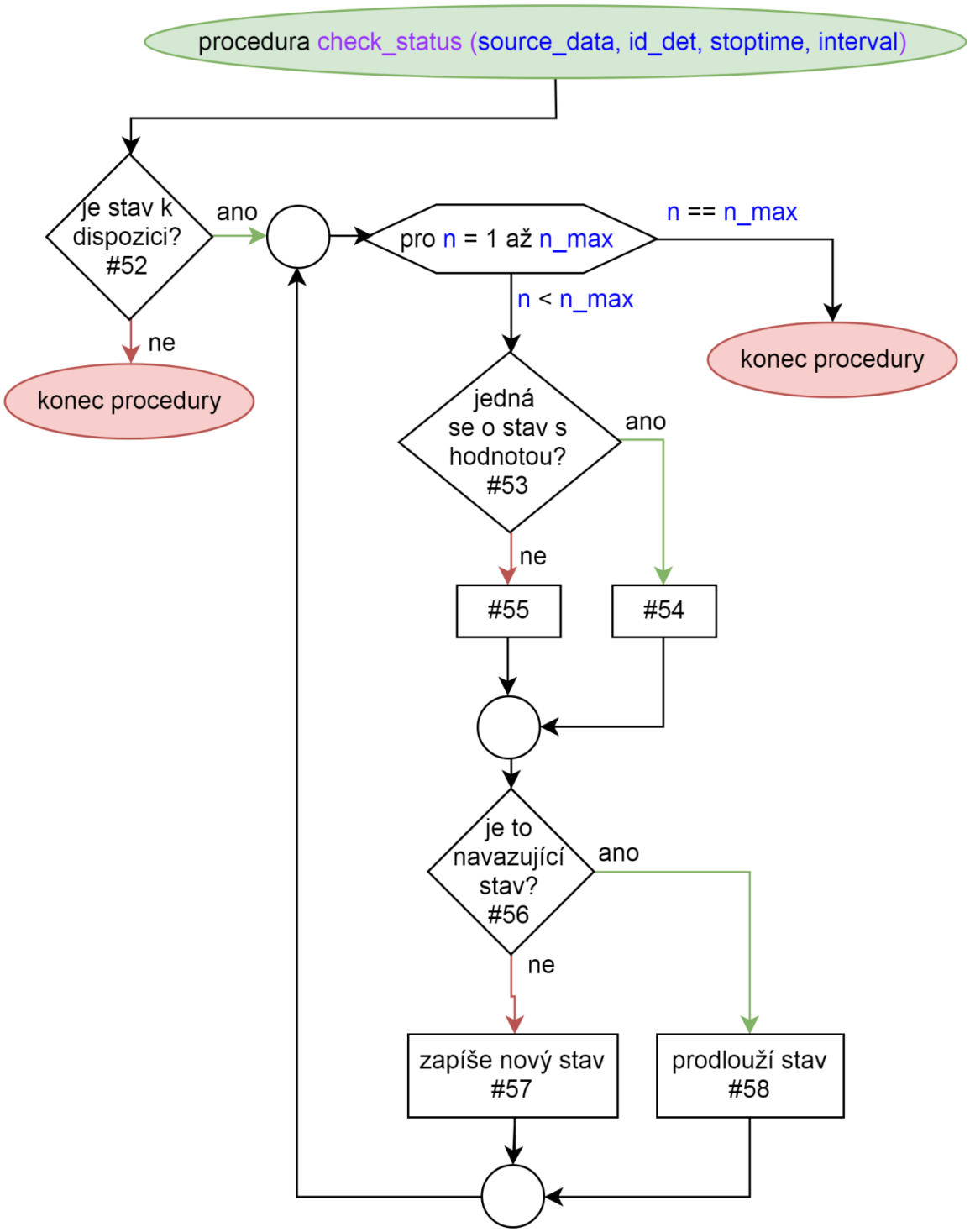


Obrázek 26 - Vývojový diagram podprocedury switch_type_update; zdroj: archiv autora

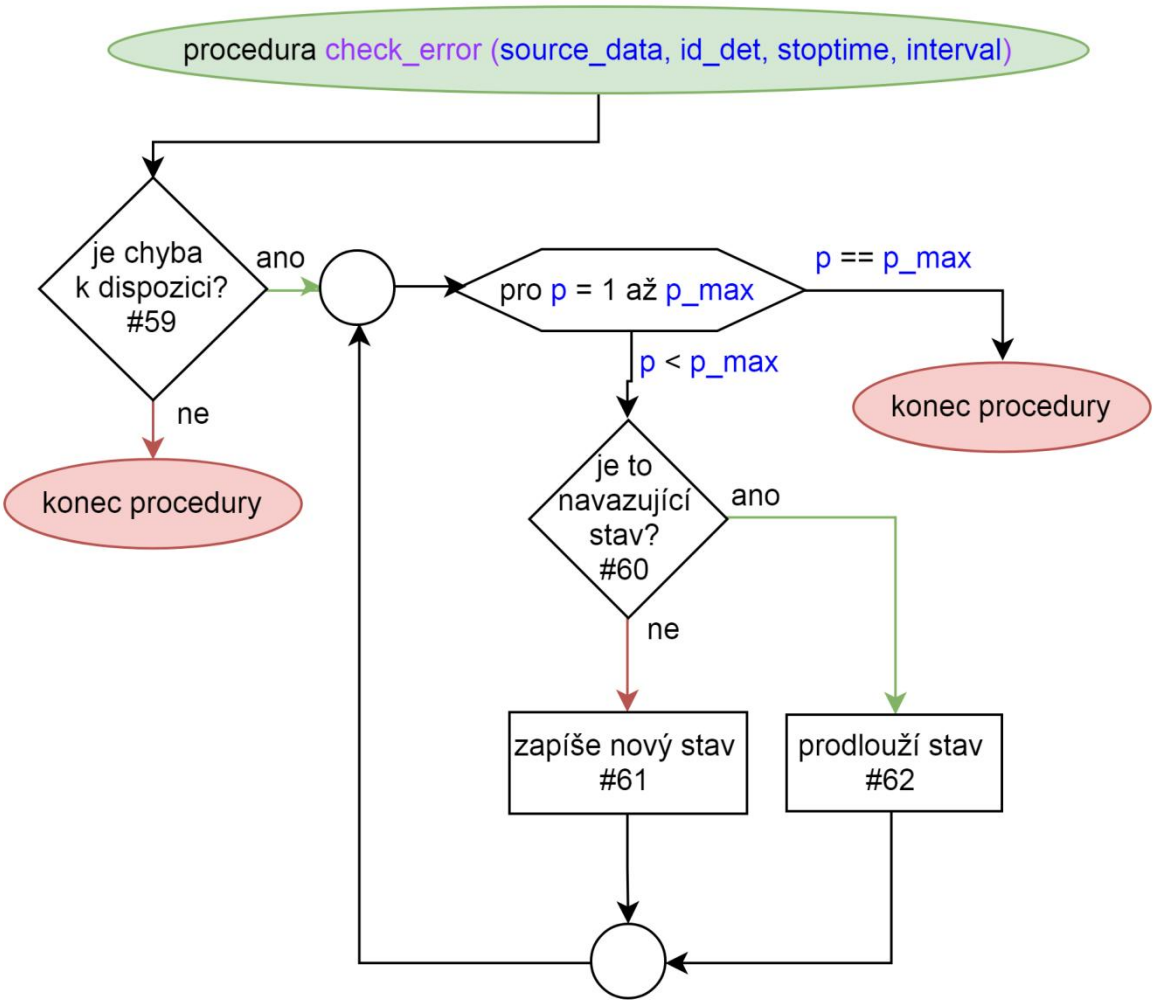
Procedura check_status a check_error

Kontrolování stavů detektorů má smysl pouze pokud detektor nějaké stavy hlásí. Pokud se tak děje, hlásí jich několik najednou a rozlišují se prosté stavy a stavy s hodnotou, které byly popsány v kapitole č. 3.1.4 v popisu tabulky historie stavů. Program tedy spočítá počet stavů a po jednom vyhodnocuje, zda se jedná o stav s hodnotou či nikoliv. Dále se stav zapisuje nebo v případě, že se nezměnil od stavu minulého, jen prodlužuje do tabulky statushistory v databázi.

V podobné proceduře check_error se řeší obdobná kontrola chyb, pouze chyby nemají hodnoty. Vývojový diagram kontroly stavů je zobrazen na obrázku č. 27 a vývojový diagram kontroly chyb je na obrázku č. 28. Seznam známých stavů je uveden v manuálu T3 Remote client [zdroj č.5, příloha č.1].



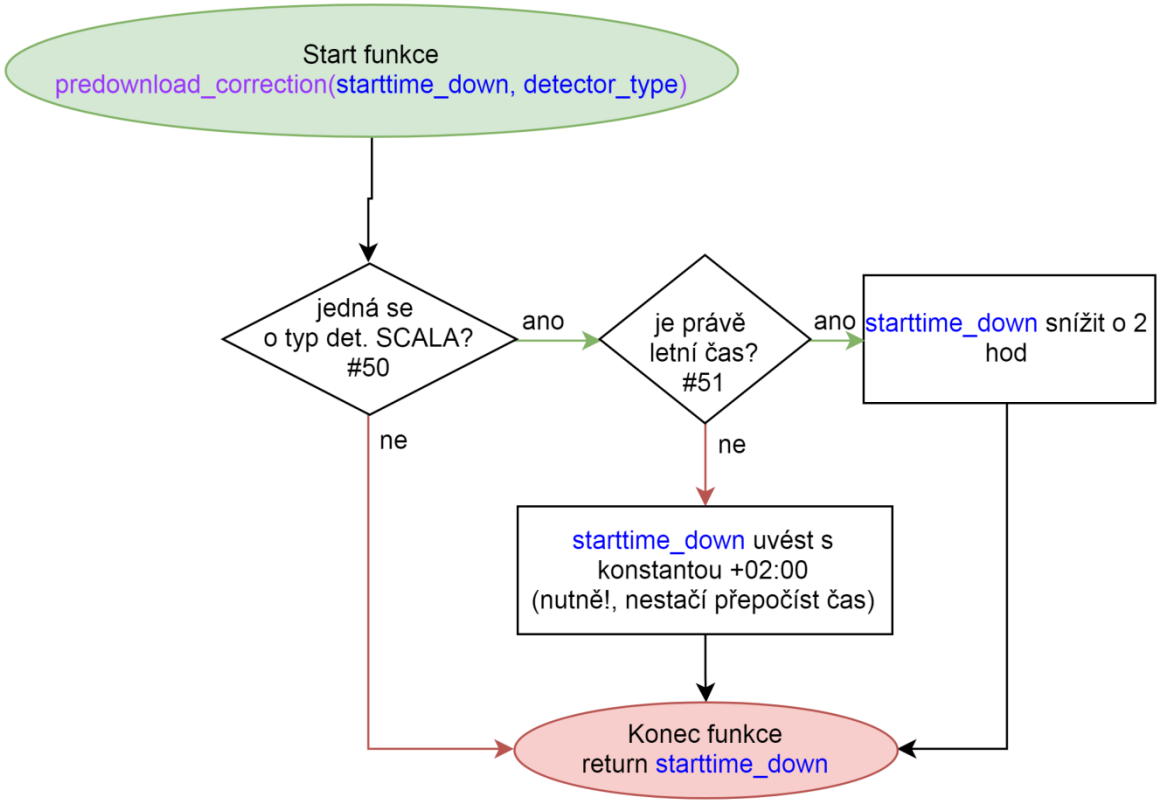
Obrázek 27 - Vývojový diagram procedury check_status; zdroj: archiv autora



Obrázek 28 - Vývojový diagram procedury check_error; zdroj: archiv autora

Funkce predownload_correction

Z kapitoly 1.3.3 vyplývá, že je zapotřebí u detektorů typu SCALA upravit žádané časy. To se řeší právě pomocí této funkce dle pravidel uvedených v kapitole 1.3.3 .



Obrázek 29 - Vývojový diagram funkce predownload_cerrection; zdroj: archiv autora

Funkce validation_dat

Ověření, zda data odpovídají reálně očekávaným hodnotám, má na starosti funkce validace dat. Funguje tak, že se nejprve ověří, zda v XML zprávě jsou vůbec data od detektorů. Poté si aplikace zkontroluje, zda databáze disponuje daty k porovnání.

Data k porovnání s daty kontrolovanými splňují následující podmínky:

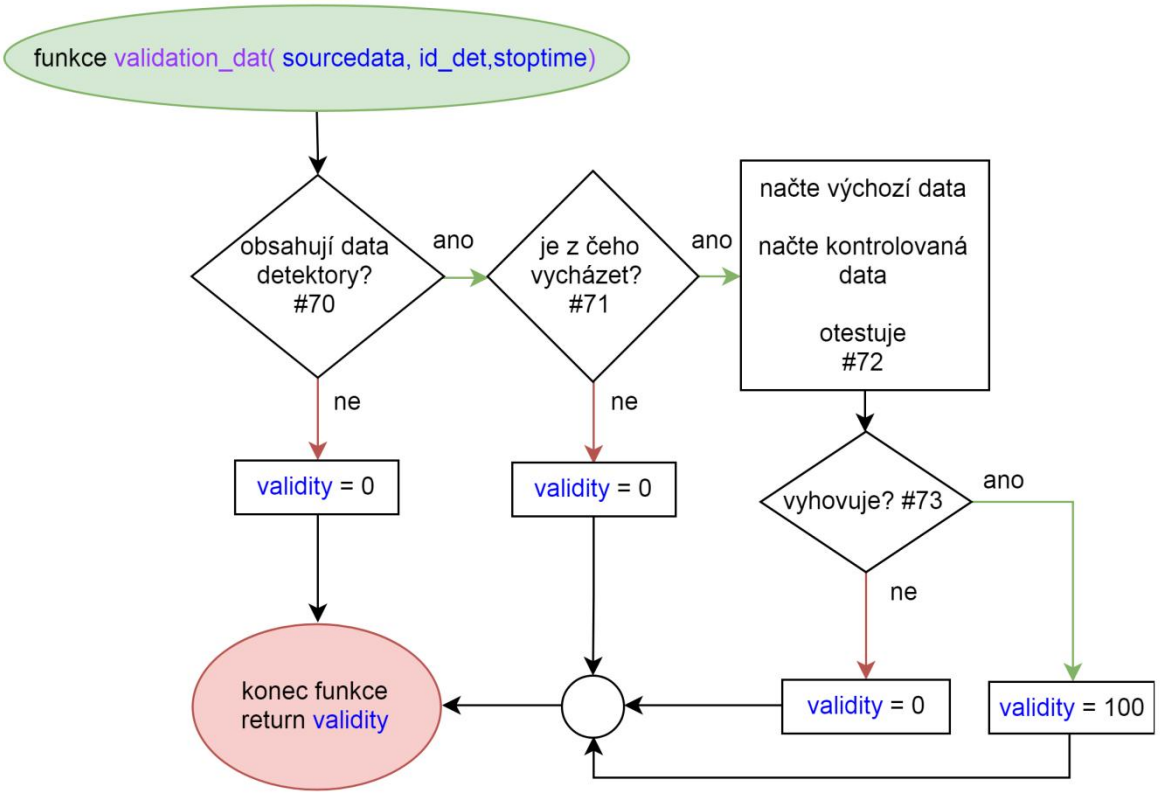
a) jedná se o data ze stejného detektoru

b) jedná se o data ze stejné denní doby, např. platné k času 12:34:56 nezávisle na časovém pásmu

c) jedná se o data ze stejného dne týdne jako den porovnávaných dat, např. středy

d) jedná se o data ze dní stejného charakteru, který je popsán v popisu tabulky "days" v kapitole 3.1.4

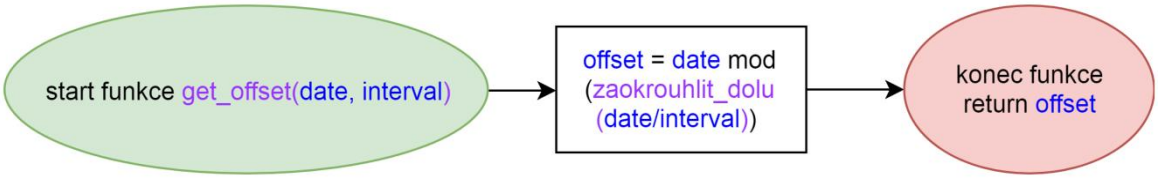
e) jedná se o intenzity všech vozidel, protože je to vhodná veličina k validaci a je detekována všemi typy detektorů



Obrázek 30 - Vývojový diagram funkce validation_dat; zdroj: archiv autora

Funkce get_offset

Funkce získání časového posunu vypočítá nejmenší časový posun oproti násobkům intervalu detektoru a vrátí jej výstupním parametrem.



Obrázek 31 - Vývojový diagram funkce get_offset; zdroj: archiv autora

3.2.5 Manuálně vykonávané funkce

V této části budou popsány nezbytné funkce, které by mohla aplikace nabízet uživatelům, pokud do systému chtějí zasáhnout. Pro veškeré následující procesy by vývojový diagram nebyl výstižný, neboť se nejedná o složité procesy, a návrh vyžaduje spíše znalost programovacího jazyka. Autor proto zvolil vhodnější postup - slovní popis.

Přidání a úpravy detektorů

Přidávání nových detektorů je sice již obsaženo v automatickém módu této aplikace, nicméně by bylo vhodné ponechat možnost přidat detektory ručně či je upravit. Například při prvním spuštění by bylo vhodnější je nahrát ručně. Důvodem jsou některé id detektorů typu SCALA, které končí jedním, dvěma či třemi apostrofy, což při vkládání do databáze PostgreSQL vyvolává obtíže. Problém se odstraní zdvojnásobením počtu apostrofů a v SQL kódu bude tedy id končit dvěma, čtyřmi či šesti apostrofy.

Aktivace a deaktivace detektorů

Pokud je detektor aktivní znamená to, že automatická se procedura download_raw se snaží stahovat data, která produkuje.

Automatická procedura echo kontroluje nové detektory. V případě, že se nějaký nový detektor objeví, přidá ho jako neaktivní a pošle zprávu editorovi. Editor by měl mít možnost jednoduchou možnost detektor aktivovat změnou hodnoty "acitve" z FALSE na TRUE.

Může vyvstat potřeba nejen aktivace, ale i deaktivace. V případě, kdy bude známo, že nějaký konkrétní detektor již nefunguje, a fungovat již nebude, je zbytečné, aby se z něj stahovala nevalidní data, a proto jej bude vhodné deaktivovat.

Ukázka SQL kódu aktivace a deaktivace je k nalezení v příloze D.2.1.

Přidání a úpravy typů detektorů, stavů a chyb

Automatická procedura echo kontroluje kromě nových detektorů také nové typy detektorů, nové stavy a nové typy, která automaticky do databáze přidává. Při programování této funkce v manuálním módu aplikace lze vycházet z SQL kódů #46b, #46c a #46d, které jsou v příloze D.1.

Přidání a úpravy profilů

Profily popsané v kapitole č. 3.1.4 se přidávají pouze v manuálním módu aplikace.

Pro výukové účely UITS se budou používat profily detektorů na dvoupruhovém jízdním pásu. Jako ukázka kódu přidání těchto profilů slouží příloha D.2.2, která obsahuje veškeré profily požívané pro předmět UITS.

Zařazení detektorů do profilů

Naplnění významu profilů lze dosáhnout přiřazením detektorů do profilu. Předpokladem je, že přiřazovaný detektor je již obsažen v tabulce "detectors" a profil, do kterého se detektor přiřazuje, je již v tabulce "profiles".

Pro ukázku kódu zařazení detektorů do profilu slouží příloha D.2.3, která obsahuje zařazení všech detektorů do profilů v rámci úlohy v předmětu UITS.

Přidání a úpravy sekcí

Pro přidání sekce (popis, co znamená sekce, je v kapitole č. 3.1.4) je předpoklad již přidaného profilu v tabulce profiles. Příklad kódu přidání jedné sekce je uveden v příloze D.2.4. Evidence sekcí TSK je k nalezení na webových stránkách TSK [12].

3.3 Návrh studentského klienta

3.3.1 Autentizace

Z důvodu ochrany dat je vyžadováno, aby do výukové databáze měly přístup pouze oprávněné osoby. Je vhodné, aby byl použit jednotný univerzitní systém přihlášení, stejný jako je do databáze KOS.

Studentská aplikace bude po úspěšném přihlášení studenta vstupovat do databáze s omezenými pravomocemi, povoleno bude pouze čtení tabulky dat.

3.3.2 Funkce

Forma zadávání dotazu

Jak vyplývá z uživatelských požadavků, je žádoucí, aby studenti sestrojovali a zadávali vstupní dotaz do studentské aplikace stejně jako je tomu u současné aplikace T3 Remote Client.

Vstupem tedy bude textový řetězec:

method=<metoda> deviceid=<id detektoru> starttime=<počáteční čas> stoptime=<koncový čas>

Evidence zadaných dotazů

Aplikace bude evidovat, který student jaký dotaz a kdy zadal, pro lepší zpětnou vazbu.

Zápis bude provádět do tabulky querylog, kam vloží identifikátor studenta, čas zadání dotazu a jeho znění. SQL kód pro vložení zápisu je k nalezení v příloze D.3.1.

Sestavení SQL dotazu

Pro dotazování aplikace do databáze vygeneruje ze vstupních parametrů zadaných studenty SQL dotaz. Student vůbec neuvidí, že se jedná o SQL databázi. Sestrojování dotazu "SELECT * FROM dat WHERE ... " budou následovat logické podmínky ze vstupních parametrů. Tomuto však musí předcházet kontrola proti tzv. SQL injektáži. Kontrola spočívá v ověření, že požadované parametry se v tabulce nacházejí. Musí se nejdřív načíst všechny uložené parametry a na úrovni aplikace porovnat.

SQL kód k vytváření a ověřování dotazu je k nalezení v příloze D.3.2.

Generování XML odpovědi

Výstupem pro studenty má být XML soubor. PostgreSQL umí vygenerovat XML soubor z příkazu SELECT. Na úrovni aplikace se pak pouze pozmění deklarace XML souboru a vyjádření hodnot NULL.

Ukázka SQL kódu pro získání XML a pravidla potřebných úprav jsou k nalezení v příloze D.3.3.

3.4 Návrh pedagogického klienta

3.4.1 Autentizace

Vzhledem k bezpečnosti je doporučeno použití jednotného univerzitního přihlášení jako u studentů.

Pedagogická aplikace bude po úspěšném přihlášení pedagoga vstupovat do databáze s omezenými pravomocemi. Pro čtení budou povoleny veškeré tabulky. Pro úplnou editaci, mimo smazání tabulky, bude zpřístupněna tabulka zadání a tabulka studentů.

3.4.2 Funkce

Vkládání studentů

Na základě vložení tabulky vytvořené MS Excel obsahující jméno, příjmení a osobní číslo studentů předmětu UITS aplikace přidá studenty do tabulky "students".

SQL kód pro vkládání studentů a zadání je k nalezení v příloze D.4.1.

Přiřazování zadání a přehled validity zadání

Vývojový diagram této části programu se nachází na obrázku č. 32.

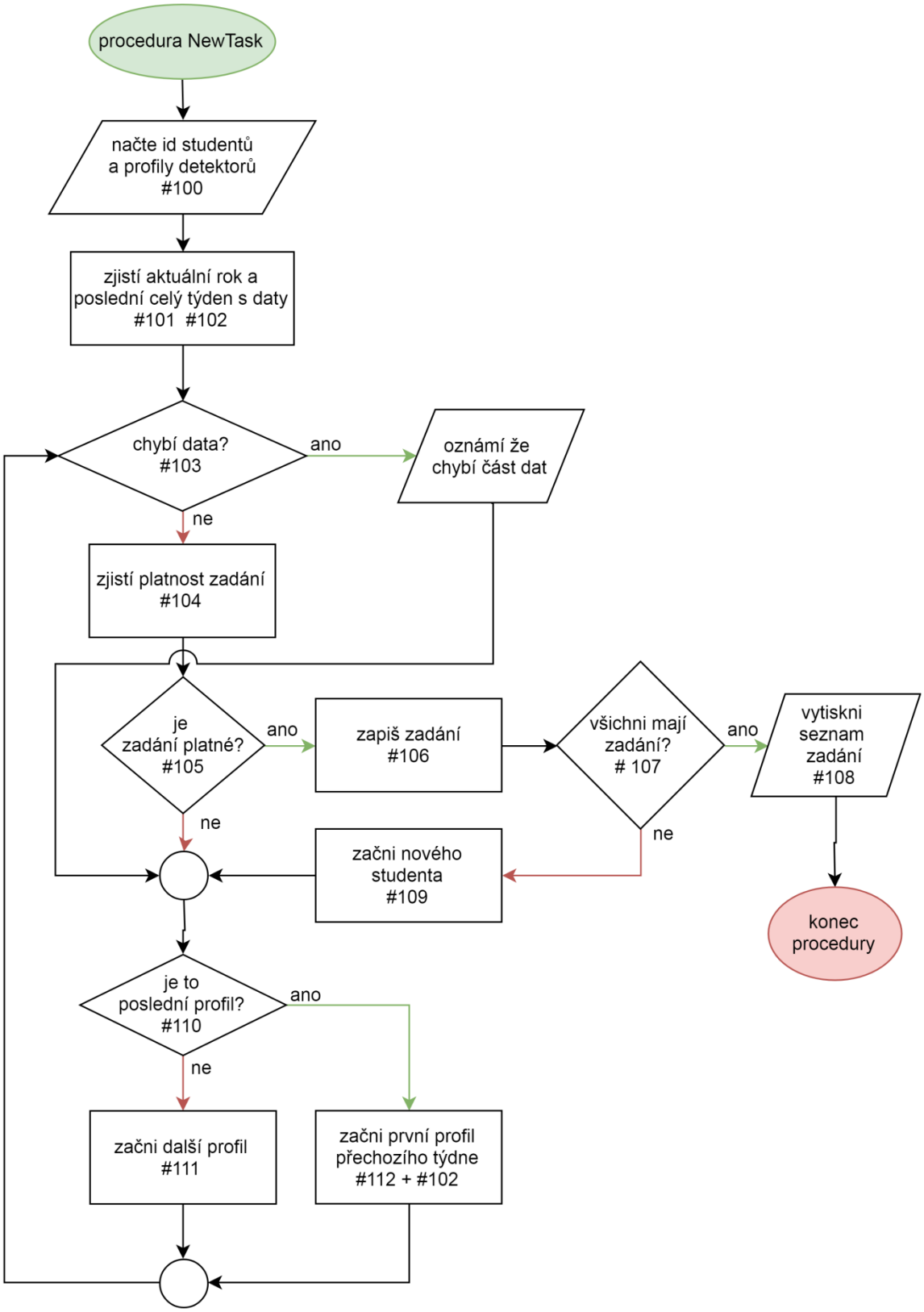
Po spuštění procedury NewTask dojde k načtení id studentů, kteří nemají přiřazené zadání úlohy a načtení profilů detektorů uvažovaných pro UITS. Poté zjistí, který poslední celý týden obsahuje stažená data od detektorů využívaných pro UITS a začne je přiřazovat, pokud jsou validní.

Validita zadání, nebo vhodněji integrita dat, je procentní vyjádření, kolik dat z celku je plně validní. Všechna zadání obsahují profily o dvou detektorech, která posílají data po pěti minutách, tzn. kolik procent datových paketů z 4032 možných je plně validní. Pro vyhodnocení vhodného zadání je potřebná integrita alespoň 80 %.

Program hledá vhodné zadání postupně pro každého studenta. Nejprve prohledá všechny profily v nejnovějším týdnu, poté proces opakuje v předchozím týdnu atd. dokud všichni studenti nemají přiřazené zadání.

Po úspěšném přiřazení zadání všem studentům vytiskne tabulku všech zadání v daném roce, která obsahuje veškeré podstatné údaje o studentovi, zadání úlohy včetně údaje o integritě.

SQL kódy výpočtu validity zadání zastoupeny znaky #100 až #112 jsou k nalezení v příloze D.4.2.



Obrázek 32 - Vývojový diagram procedury new_task; zdroj: archiv autora

Generování grafu

Funkce generování grafu zobrazí grafy průběhů intenzit a rychlostí. Vstupním parametrem přitom bude pouze číslo zadání. Hodnoty pro zobrazení jednoduchého grafu budou generovány databází, na základě propojení tabulek zadání-profil-det2prof-detectors-dat a budou sečteny intenzity a zprůměrovány rychlosti všech detektorů v profilu.

Ukázka SQL kódu pro generování dat pro graf je k nalezení v příloze D.4.3.

Přehled evidence zadaných dotazů

Pedagog bude moci zobrazit veškeré dotazy zadané studenty. Jako vstupní parametry bude moci být použit identifikátor studenta, jméno studenta, časové rozmezí od-do, nebo jejich kombinace.

Ukázka SQL kódu pro přehled zadaných dotazů je k nalezení v příloze D.4.4.

Závěr

Výuková databáze dat zahrnuje relační databázi a tři hlavní aplikace pro správný chod systému. Každá aplikace usnadňuje nezáživnou činnost svému přiřazenému uživateli, kterou musel dosud provádět.

Byl proveden zásadní rozbor výchozího systému databáze TSK se serverovou aplikací T3 Remote Client a po obsáhlých zpětných analýzách bylo odhaleno nemalé množství nesprávně fungujících funkcionalit.

Shrnuté uživatelské požadavky daly jasný směr, kudy se má návrh odvíjet. Vypsání veškerých požadovaných funkcionalit od každé uživatelské role vytvořilo schéma navrhovaného systému, které se nadále rozvíjelo.

Za pomoci odborných zdrojů, a především pomocí zkušeností s chováním výchozího systému, vznikl navrhovaný systém v podobě vytvořeného základu SQL databáze a navrhnutých aplikací na úrovni blížící se zdrojovému kódu. Nemohl být řádně implementován a odlazen pro výpadek výchozího systému, který přetrvával po značnou dobu v průběhu tvorby databáze a vyvolával obrovskou nejistotu, zda dojde k obnovení provozu.

Implementace a odlazení jsou následující vývojové kroky, které bude potřeba udělat.

Autor věří, že jím vytvořená publikace poslouží jako vhodná příručka pro implementaci systému a usnadní tak úsilí, které je obecně spjato s tvorbou nového automatizovaného systému, který usnadňuje práci uživatelům a zvyšuje efektivitu.

Seznam literatury

[1] Smlouva na provádění údržby strategických dopravních detektorů v řezových (SDDR) a klimatických vozovkových detektorů (KVD) na území hl. m. Prahy. Praha, 2016. Dostupné z: https://smlouvy.gov.cz/smlouva/soubor/3398586/TSKRP006Q49K.pdf

[2] FLIR TrafiCam Vehicle presenc detector. *FLIR Systems* [online]. Wilsonville [cit. 2019-05-28]. Dostupné z: https://www.flir.com/products/traficam/

[3] ETSI TETRA: TErrestrial Trunked RAdio. *ETSI* [online]. Valbonne, France [cit. 2019-05-28]. Dostupné z: https://www.etsi.org/technologies/tetra?highlight=YToxOntpOjA7czo1OiJ0ZXRyYSI7fQ==

[4] *Hospodářské noviny* [online]. Praha, ČR [cit. 2019-05-28]. Dostupné z: https://domaci.ihned.cz/c1-65437200-komunikace-strazniku-a-zachranky-neni-kodovana-napichnout-se-do-ni-muze-kazdy

[5] KABELKA. *TRANSMITTER 3 komunikační protokol*. Praha, 2016.

[6] Dopravní modely. *TSK Praha* [online]. Praha, ČR [cit. 2019-05-28]. Dostupné z: https://www.tsk-praha.cz/wps/portal/root/dopravni-inzenyrstvi/dopravni-modely-mesta

[7] KREJČÍ, Jakub. ANALÝZA POŽADAVKŮ PRO TVORBU VÝUKOVÉ DATABÁZE DOPRAVNÍCH DAT. PRAHA, 2017. Bakalářská práce. ČVUT Praha, Fakulta Dopravní. Vedoucí práce Langr Martin.

[8] ČARSKÝ, Jiří. *Bílá kniha ČVUT FD*. Praha, 2017.

[9] ROESS, Roger P., Elena S. PRASSAS a William R. MCSHANE. *Traffic engineering*. 4th ed. Upper Saddle River, NJ: Pearson, c2011. ISBN 0-13-613573- 0. kapitola 8

[10] Greenshields Models. *University of Idaho - TransportationEngineering* [online]. Idaho, USA [cit. 2019-05-28]. Dostupné z: https://www.webpages.uidaho.edu/niatt_labmanual/chapters/trafficflowtheory/theoryandconcepts/GreenshieldsModel.htm

[11] *Strategické detektory TSK* [online]. Eltodo [cit. 2017-06-25]. Dostupné z: stratdet.eltodo.cz

[12] Intenzity dopravy. *TSK Praha* [online]. Praha, ČR [cit. 2019-05-28]. Dostupné z: https://www.tsk-praha.cz/wps/portal/root/dopravni-inzenyrstvi/intenzity-dopravy

Seznam obrázků

Seznam příloh

Příloha A - Seznam poloh detektorů SDDU

V této příloze lze nalézt seznam poloh detektorů SDDU, avšak není známo ke kterým konkrétním detektorům náleží. Uvedený kód TSK nijak nesouvisí s id detektorů, které protokol T3 poskytuje. Seznam slouží k budoucímu zkoumání těchto detektorů.

Popis polohy	Kód TSK	Souřadnice
V Olšinách z centra	U000626-Zc	E14°29'2,19", N50°4'15,97"
Vinohradská do centra	U00145-Zc	E16°45'22", N52°21'22"
Švehlova do centra	U007494-Zc	E14°31'26,14", N50°3'10,99"
Wilsonova do centra	U102611-Jc	E14°26'6,52", N50°5'1,4"
Koněvova z centra	U300272-Vc	E14°28'57,61", N50°5'32,29"
Jana Želivského z centra	U302021-Sc	E14°28'18,6", N50°4'55,12"
Modřanská z centra	U401777-Jc	E14°24'8,7", N50°1'4,16"
Gen. Šišky do centra	U407223-Zc	E14°25'12,27", N50°0'18,5"
5. Května II z centra	U409911-Sc	E14°27'53,22", N50°2'31,7"
Spořilovská směr Brno	U415029-Sc	E14°29'7,14", N50°2'47,03"
Michelská do centra	U431829-Sc	E14°27'43,47", N50°2'33,78"
Radlická II. z centra	U501131-Jc	E14°21'51,09", N50°3'1,66"
Plzeňská do centra	U501483-Vc	E14°20'53,85", N50°4'4,85"
K Barrandovu do centra	U512200-Sc	E14°22'54,82", N50°1'56,47"
Strakonická z centra	U515576-Jc	E14°23'44,31", N50°0'2,78"
Strakonická do centra	U520330-Sc	E14°24'40,35", N50°3'35,19"
Patočkova z centra	U600738-Vc	E14°21'48,32", N50°5'6,71"
Evropská z centra	U617130-Zc	E14°22'42,8", N50°5'59,35"
U Uránie z centra	U702060-Zc	E14°27'1,8", N50°6'26,9"
Wilsonova z centra	U801087-Sc	E14°26'15,37", N50°5'29,23"
V Holešovičkách do centra	U802561-Jc	E14°27'2,72", N50°7'1,71"
Průmyslová z centra	U908146-Sc	E14°32'1,1", N50°5'29,78"
Novosibřinská do centra	U912879-Vc	E14°39'47,51", N50°4'32,73"

PŘÍLOHA B - SQL kód systému databáze

```
DROP TABLE IF EXISTS detectors CASCADE;
DROP TABLE IF EXISTS detectortypes CASCADE;
DROP TABLE IF EXISTS profiles CASCADE;
DROP TABLE IF EXISTS det2profil CASCADE;
DROP TABLE IF EXISTS sections CASCADE;
DROP TABLE IF EXISTS dat CASCADE;
DROP TABLE IF EXISTS students CASCADE;
DROP TABLE IF EXISTS querylog CASCADE;
DROP TABLE IF EXISTS task CASCADE;
DROP TABLE IF EXISTS statuslist CASCADE;
DROP TABLE IF EXISTS statushistory CASCADE;
DROP TABLE IF EXISTS errorlist CASCADE;
DROP TABLE IF EXISTS errorhistory CASCADE;
DROP TABLE IF EXISTS days CASCADE;
CREATE TABLE "detectors" (
  "id_det" varchar(30) PRIMARY KEY,
  "type" varchar(10) NOT NULL ,
  "active" bool NOT NULL,
  "lat" float4,
  "lon" float4,
  "azimut" smallint CHECK (azimut<=360),
  "PositionDescription" varchar(100),
  "SpeedLimit" smallint,
  "TimeOffset" time
);
CREATE TABLE "detectortypes" (
  "type" varchar(10) PRIMARY KEY,
  "RawInterval" time NOT NULL CHECK ("RawInterval"<='01:00:00' AND "RawInterval">='00:00:00' ),
  "accessibility" interval NOT NULL,
  "ClassDescription" varchar(100)
);
CREATE TABLE "profiles" (
  "id_profil" varchar(20) PRIMARY KEY,
```

```
  "ProfilName" varchar(50)
);
CREATE TABLE "sections" (
  "id_section" serial PRIMARY KEY,
  "id_profil" varchar(20),
  "SectionType" varchar(20) NOT NULL,
  "SectionStartID" varchar(20),
  "SectionEndID" varchar(20),
  "SectionID" varchar(20)
        CONSTRAINT chk_section CHECK (("SectionStartID" IS NOT NULL AND "SectionEndID" IS
NOT NULL) OR("SectionID" IS NOT NULL))
);
CREATE TABLE "det2profil" (
  "id_det2profil" serial PRIMARY KEY,
  "id_det" varchar(30) NOT NULL,
  "id_profil" varchar(20) NOT NULL
);
CREATE TABLE "dat" (
  "id_dat" bigserial PRIMARY KEY,
  "id_det" varchar(30) NOT NULL,
  "raw" bool NOT NULL,
  "confidence" smallint CHECK(("confidence" IS NULL) OR ("confidence"<=100 AND "confidence">=0)),
  "validity" smallint CHECK(("validity" IS NULL) OR ("validity"<=100 AND "validity">=0)),
  "attemptnumber" smallint DEFAULT 1 CHECK ("attemptnumber"<=5 AND "attemptnumber">=0),
  "stoptime" timestamptz NOT NULL,
  "starttime" timestamptz NOT NULL,
  "count_all" smallint,
  "count_passengercar" smallint,
  "count_lighttruck" smallint,
  "count_heavytruck" smallint,
  "count_alltruck" smallint,
  "speed_all" smallint,
  "speed_passengercar" smallint,
  "speed_alltruck" smallint,
  "occupancy_all" smallint,
  "count1" smallint,
  "count2" smallint,
```

"count3" smallint,
"count4" smallint,
"count5" smallint,
"count6" smallint,
"count7" smallint,
"count8" smallint,
"count9" smallint,
"count10" smallint,
"count11" smallint,
"count12" smallint,
"count13" smallint,
"speed1" smallint,
"speed2" smallint,
"speed3" smallint,
"speed4" smallint,
"speed5" smallint,
"speed6" smallint,
"speed7" smallint,
"speed8" smallint,
"speed9" smallint,
"speed10" smallint,
"speed11" smallint,
"speed12" smallint,
"speed13" smallint,
"weight1" int,
"weight2" int,
"weight3" int,
"weight4" int,
"weight5" int,
"weight6" int,
"weight7" int,
"weight8" int,
"weight9" int,
"weight10" int,
"weight11" int,
"weight12" int,
"weight13" int,

```
  "occupancy1" smallint,
  "occupancy2" smallint,
  "occupancy3" smallint,
  "occupancy4" smallint,
  "occupancy5" smallint,
  "occupancy6" smallint,
  "occupancy7" smallint,
  "occupancy8" smallint,
  "occupancy9" smallint,
  "occupancy10" smallint,
  "occupancy11" smallint,
  "occupancy12" smallint,
  "occupancy13" smallint,
  "temperature" float4,
  "voltage" float4
);
CREATE TABLE "task" (
  "id_task" serial PRIMARY KEY,
  "id_stu" varchar(6),
  "id_profil" varchar(20),
  "year" smallint NOT NULL CHECK("year">=2019 AND "year"<=2050),
  "week" smallint NOT NULL CHECK("week">=1 AND "week"<=53),
  "integrity" smallint CHECK("integrity">=0 AND "integrity"<=100)
);

CREATE TABLE "students" (
  "id_stu" varchar(6) PRIMARY KEY CHECK(Length("id_stu")=6),
  "FirstName" varchar(30) NOT NULL,
  "LastName" varchar(30) NOT NULL
);
CREATE TABLE "querylog" (
  "id_log" serial PRIMARY KEY,
  "id_stu" varchar(6) NOT NULL,
  "logtime" timestamptz NOT NULL,
  "query" varchar(100) NOT NULL
);
CREATE TABLE "statuslist" (
```

```
  "id_status" varchar(10) PRIMARY KEY,
  "statusdescription" varchar(100) NOT NULL
);
CREATE TABLE "statushistory" (
  "id_statuslog" bigserial PRIMARY KEY,
  "id_status" varchar(10),
  "id_det" varchar(30),
  "startstatus" timestamptz,
  "stopstatus" timestamptz,
  "valuestatus" bool
);
CREATE TABLE "errorlist" (
  "id_error" varchar(10) PRIMARY KEY,
  "errordescription" varchar(100) NOT NULL  );
CREATE TABLE "errorhistory" (
  "id_errorlog" bigserial PRIMARY KEY,
  "id_error" varchar(10),
  "id_det" varchar(30),
  "starterror" timestamptz,
  "stoperror" timestamptz);
CREATE TABLE "days" (
  "id_day" serial PRIMARY KEY,
  "StartDay" timestamptz,
  "StopDay" timestamptz,
  "DayCharacter" varchar(12) CHECK ("DayCharacter" IN('workingday','holiday','preholiday','postholiday')),
  "WeekDay" smallint CHECK ("WeekDay" >=1 AND "WeekDay" <=7),
  "WeekNum" smallint CHECK ("WeekNum" >=1 AND "WeekNum" <=53),
  "WeekNumYear" smallint CHECK ("WeekNumYear" >=2019 AND "WeekNum" <=2050)
);
ALTER TABLE "detectors" ADD FOREIGN KEY ("type") REFERENCES "detectortypes" ("type");
ALTER TABLE "det2profil" ADD FOREIGN KEY ("id_det") REFERENCES "detectors" ("id_det");
ALTER TABLE "det2profil" ADD FOREIGN KEY ("id_profil") REFERENCES "profiles" ("id_profil");
ALTER TABLE "task" ADD FOREIGN KEY ("id_profil") REFERENCES "profiles" ("id_profil");
ALTER TABLE "sections" ADD FOREIGN KEY ("id_profil") REFERENCES "profiles" ("id_profil");
ALTER TABLE "task" ADD FOREIGN KEY ("id_stu") REFERENCES "students" ("id_stu");
ALTER TABLE "querylog" ADD FOREIGN KEY ("id_stu") REFERENCES "students" ("id_stu");
ALTER TABLE "dat" ADD FOREIGN KEY ("id_det") REFERENCES "detectors" ("id_det");
```

```
ALTER TABLE "statushistory" ADD FOREIGN KEY ("id_det") REFERENCES "detectors" ("id_det");
ALTER TABLE "errorhistory" ADD FOREIGN KEY ("id_det") REFERENCES "detectors" ("id_det");
ALTER TABLE "statushistory" ADD FOREIGN KEY ("id_status") REFERENCES "statuslist"
("id_status");
ALTER TABLE "errorhistory" ADD FOREIGN KEY ("id_error") REFERENCES "errorlist" ("id_error");
--Filling table detectortypes
INSERT INTO detectortypes VALUES ('CollectR','00:05:00','7 D','1-2');
INSERT INTO detectortypes VALUES ('SCALA','00:01:30','1 D','0');
INSERT INTO detectortypes VALUES ('Traficam','00:05:00','1 D','0');
INSERT INTO detectortypes VALUES ('MUR','00:05:00','1 D','0-1-2-3');
INSERT INTO detectortypes VALUES ('SDDU','00:05:00','1 D','0-1-2-3');
INSERT INTO detectortypes VALUES ('WIM','00:05:00','1 D','1-13');
INSERT INTO detectortypes VALUES ('MP','00:05:00','1 D','0-1-2-3');
```

PŘÍLOHA C - Pomůcka k tvorbě tabulky dní a svátků

Vkládání státních svátků do přílohy by bylo příliš obsáhlé, zde je návod, jak v MS Excel udělat seznam odpovídajících dat pro tabulku "days"

ukázka listu svátky:

LIST SVÁTKY			
VELIKONOCE		PEVNÉ SVÁTKY	
datum	svátkovost	datum	svátkovost
13.4.2001	PRAVDA	1.1.2000	PRAVDA
16.4.2001	PRAVDA	1.5.2000	PRAVDA
29.3.2002	PRAVDA	8.5.2000	PRAVDA
1.4.2002	PRAVDA	5.7.2000	PRAVDA
18.4.2003	PRAVDA	6.7.2000	PRAVDA
21.4.2003	PRAVDA	28.9.2000	PRAVDA
9.4.2004	PRAVDA	28.10.2000	PRAVDA
12.4.2004	PRAVDA	17.11.2000	PRAVDA
25.3.2005	PRAVDA	24.12.2000	PRAVDA
28.3.2005	PRAVDA	25.12.2000	PRAVDA
14.4.2006	PRAVDA	26.12.2000	PRAVDA
17.4.2006	PRAVDA		
6.4.2007	PRAVDA		
9.4.2007	PRAVDA		

Data pohyblivých státních svátků jsou určeny dle zvyklostí a de-facto normou ISO 8601. Jedná se o data Velikonočního pátku a Velikonočního pondělí v následujících letech, která jsou jasně definovány.

Data pevných státních svátků jsou definovány zákonem o státních svátcích, o ostatních svátcích, o významných dnech a o dnech pracovního klidu č. 245/2000 Sb. .

Každý den je určen začátkem dne (sloupec A), koncem dne (B) a dnem v týdnu (C), který je obecně chápán a netřeba vysvětlovat (po=1; ne=7). Pro propojení funkce velikonoc (zda daný den je velikonoční pátek či pondělí) slouží sloupec velikonoce (D) = IFERROR(SVYHLEDAT(A3;svátky!A:B;7;NEPRAVDA);NEPRAVDA)

Sloupec pevný svátek (E) označuje, zda je či není v daný den pevný svátek =IFERROR(SVYHLEDAT(DATUM(2000;MĚSÍC(A3);DEN(A3)); svátky!$C$4:$D$14;2;NEPRAVDA);NEPRAVDA)

Sloupec svátek (F) je pouze disjunkce předchozích sloupců =NEBO(D3;E3)

Sloupec charakter (G) označuje zda se jedná o svátek, den po svátku či před svátkem =KDYŽ(F4=PRAVDA;"HOLIDAY";KDYŽ(F3=PRAVDA;"POSTHOLIDAY";KDYŽ(F5=PRAVDA;"PREHOLIDAY";"REGULAR")))

Sloupec týden (H) určuje číslo týdne =ISOWEEKNUM(A3;2)

Sloupec rok číslování týdne (I) určuje rok číslování týdne =KDYŽ(H3=WEEKNUM(A3:A32);ROK(A3);KDYŽ(H3=1;ROK(A3)+1;KDYŽ(H3=52;ROK(A3)-1;KDYŽ(H3=53;ROK(A3)-1))))

ukázka listu dny:

LIST DNY								
ZAČÁTEK DNE	KONEC DNE	DEN TÝDNE	VELIKONOCE	PEVNÝ SVÁTEK	SVÁTEK	CHARAKTER	TÝDEN	ROK ČÍSLOVÁNÍ TÝDNE
2019-09-23 00:00:00	2019-09-23 23:59:59	1	NEPRAVDA	NEPRAVDA	NEPRAVDA	#REF!	39	2019
2019-09-24 00:00:00	2019-09-24 23:59:59	2	NEPRAVDA	NEPRAVDA	NEPRAVDA	REGULAR	39	2019
2019-09-25 00:00:00	2019-09-25 23:59:59	3	NEPRAVDA	NEPRAVDA	NEPRAVDA	REGULAR	39	2019
2019-09-26 00:00:00	2019-09-26 23:59:59	4	NEPRAVDA	NEPRAVDA	NEPRAVDA	REGULAR	39	2019
2019-09-27 00:00:00	2019-09-27 23:59:59	5	NEPRAVDA	NEPRAVDA	NEPRAVDA	PREHOLIDAY	39	2019
2019-09-28 00:00:00	2019-09-28 23:59:59	6	NEPRAVDA	PRAVDA	PRAVDA	HOLIDAY	39	2019
2019-09-29 00:00:00	2019-09-29 23:59:59	7	NEPRAVDA	NEPRAVDA	NEPRAVDA	POSTHOLIDAY	39	2019
2019-09-30 00:00:00	2019-09-30 23:59:59	1	NEPRAVDA	NEPRAVDA	NEPRAVDA	REGULAR	40	2019
2019-10-01 00:00:00	2019-10-01 23:59:59	2	NEPRAVDA	NEPRAVDA	NEPRAVDA	REGULAR	40	2019

PŘÍLOHA D - Pomůcka k tvorbě zdrojového kódu aplikací

Legenda vysvětlující barevnost částí kódu odpovídá legendě pro vývojové diagramy - obrázek č. 20 v diplomové práci.

PŘÍLOHA D.1.1 - Pomůcka k tvorbě zdrojového kódu Administrátorskoeditorské aplikace - automatická část

#1

načte pole `detektory [i až i_max] = SELECT id_det FROM detectors WHERE active=TRUE`

#2

načte stejně velké pole `typy_detektoru[i až i_max] = SELECT type FROM detectors WHERE active=TRUE`

#3

načte stejně velké pole `posuny [ i až i_max ] = SELECT TimeOffset FROM detectors WHERE active=TRUE`

#4

načte stejně velké pole `dostupnosti [ i až i_max ]= SELECT "accessibility" FROM detectors AS d, detectortypes AS dt WHERE d.active=TRUE AND d.type = dt.type`

#5

načte stejně velké pole `intervaly [ i až i_max ] = SELECT "RawInterval" FROM detectors AS d, detectortypes AS dt WHERE d.active=TRUE AND d.type = dt.type`

#6

`Pro každé i od 1 do i_max`

#7

zjistí zda-li má detektor uložená nějaká data

```
SELECT COUNT(stoptime) FROM dat WHERE id_det=detektory[i]  >
0
```

#8

zjistí poslední validní nebo ztracený zápis dat a převezme jeho stoptime

```
last_saved_stoptime = SELECT max(stoptime) FROM dat WHERE
det_id= detektory [i] AND (validity > 0 OR AttemptNumber=5)
```

#9

zjistí jestli před last_saved_stoptime se nachází nějaké nevalidní intervaly k opakovanému stahování

```
last_saved_stoptime > SELECT min (stoptime) FROM dat
WHERE det_id= detektory  [ i ]  AND
validity = 0 AND AttemptNumber<5
```

#10

zjistí jestli je schopný navázat na dosud uložená data

```
aktuální čas - dostupnosti[i] < last_saved_stoptime
```

#11

zjistí jestli jsou staženy nejaktuálnější možná data

```
aktuální čas - intervaly[i] >= last_saved_stoptime
```

#12

nastaví `starttime_down = last_stop_time`

#13

nastaví `stoptime_down = aktuální čas`

#14

nastaví `starttime_down = aktuální čas - dostupnosti[i]`

#20

zjistí pole začátků intervalů k opakování

```
starttime_overwrite[j až j_max]= SELECT starttime FROM dat
WHERE det_id=detektory[i] AND validity =0 AND AttemtNumber <
5
```

zjistí pole konců intervalů k opakování

```
stoptime overwrite[j až j_max]= SELECT stoptime FROM dat
WHERE det_id = detektory[i] AND validity =0 AND AttemtNumber
< 5
```

#21

nastaví `stoptime_down = max (stoptime_overwrite [...])`

#22

jsou opakovaná data dostupná?

```
starttime_down >= aktuální čas() - dostupnosti[i]
```

#23

nastaví `starttime_down = aktuální čas() - dostupnosti[i]`

#24

nastaví `starttime_down = min(starttime_overwrite [...])`

#30

korekce časů `starttime_down =`

```
predownload_correction ( starttime_down , typ_detektoru [ i
])
```

```
stoptime_down =
```

```
predownload_correction ( stoptime_down , typ_detektoru [ i ]
)
```

#32

dotaz jsou-li k dispozici nějaká data v `XMLdata`

```
count(root/msg/detector) > 0
```

#33

kontrola jestli sedí posuny

```
get_offset(root/msg/detector/ stoptime[1],
interval[i]) == posuny[i]
```

#34

aktuální posun

```
aktuální posun = get_offset(root/msg/detector/
stoptime [ 1 ] , interval [ i ])
```

se přepíše

```
UPDATE detectors SET "TimeOffset" = aktuální posun
WHERE id_det = detektory[i]
```

#40

načte seznam známých detektorů a jejich vlastností

načte pole `detid[ ] = SELECT id_det FROM detectors`

načte pole `dettypes[ ] = SELECT type FROM detectortypes`

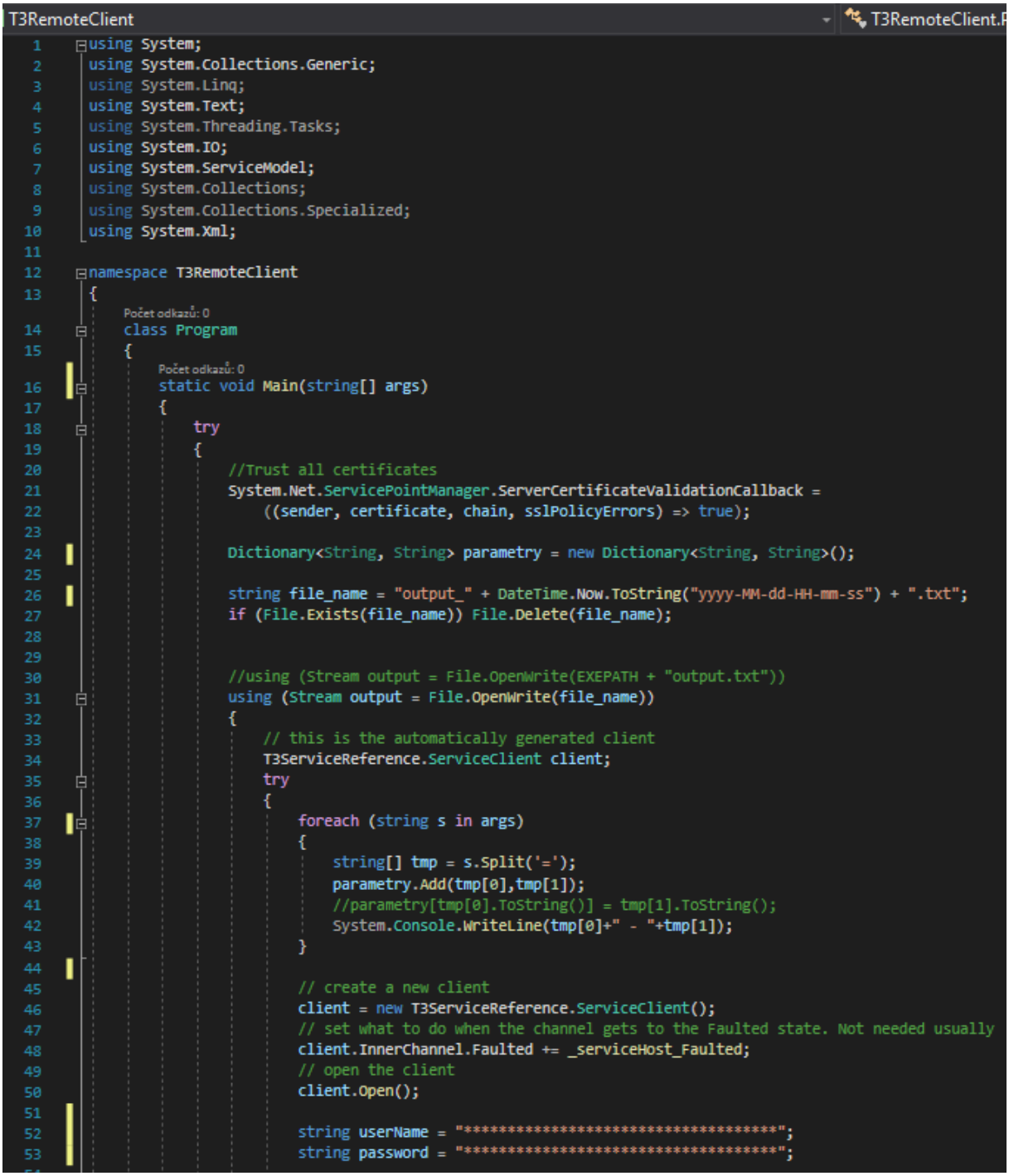
načte pole `status[ ] = SELECT id_status FROM statuslist`

načte pole `error[ ] = SELECT id_error FROM errorlist`

#41

pustí online download

následující část kódu vychází přímo z hotového kódu klienta T3:



```
T3RemoteClient                                                    T3RemoteClient.F
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6   using System.IO;
7   using System.ServiceModel;
8   using System.Collections;
9   using System.Collections.Specialized;
10  using System.Xml;
11
12  namespace T3RemoteClient
13  {
        Počet odkazů: 0
14      class Program
15      {
            Počet odkazů: 0
16          static void Main(string[] args)
17          {
18              try
19              {
20                  //Trust all certificates
21                  System.Net.ServicePointManager.ServerCertificateValidationCallback =
22                      ((sender, certificate, chain, sslPolicyErrors) => true);
23
24                  Dictionary<String, String> parametry = new Dictionary<String, String>();
25
26                  string file_name = "output_" + DateTime.Now.ToString("yyyy-MM-dd-HH-mm-ss") + ".txt";
27                  if (File.Exists(file_name)) File.Delete(file_name);
28
29
30                  //using (Stream output = File.OpenWrite(EXEPATH + "output.txt"))
31                  using (Stream output = File.OpenWrite(file_name))
32                  {
33                      // this is the automatically generated client
34                      T3ServiceReference.ServiceClient client;
35                      try
36                      {
37                          foreach (string s in args)
38                          {
39                              string[] tmp = s.Split('=');
40                              parametry.Add(tmp[0],tmp[1]);
41                              //parametry[tmp[0].ToString()] = tmp[1].ToString();
42                              System.Console.WriteLine(tmp[0]+" - "+tmp[1]);
43                          }
44
45                          // create a new client
46                          client = new T3ServiceReference.ServiceClient();
47                          // set what to do when the channel gets to the Faulted state. Not needed usually
48                          client.InnerChannel.Faulted += _serviceHost_Faulted;
49                          // open the client
50                          client.Open();
51
52                          string userName = "************************************";
53                          string password = "************************************";
```

```
77                  XmlDocument doc = new XmlDocument();
78                  Int32 token = 0;
79                  DateTime endTime = DateTime.Now.AddSeconds(Int32.Parse(parametry["interval"]));
80                  Byte[] mdata = Encoding.UTF8.GetBytes("<?xml version=\"1.0\" encoding=\"utf-8\"?>");
81                  output.Write(mdata, 0, mdata.Length);
82                  mdata = Encoding.UTF8.GetBytes("<root>");
83                  output.Write(mdata, 0, mdata.Length);
```

#42

stáhne jednu zprávu, přenastaví token pro příští zprávu a v každé msg zjistí unikátní hodnoty*

následující část kódu vychází přímo z hotového kódu klienta T3:

```
87              Byte[] mydata = Encoding.UTF8.GetBytes(client.GetOnlineData(userName, password, token));
88              doc.LoadXml(Encoding.UTF8.GetString(mydata));
89              XmlNode node = doc.DocumentElement.SelectSingleNode("/msg");
90              token = Int32.Parse(node.Attributes["token"].InnerText);
```

```
detid_msg = mydata.(root/msg/detector/@id)

odstraní duplikované hodnoty v (detid_msg)

detid_msg = mydata.(root/msg/detector/@type)

odstraní duplikované hodnoty a uloží jako dettypes_msg

status_msg= mydata.(root/msg/detector/status_list/status/@id)

odstraní duplikované hodnoty v (status_msg)

error_msg = mydata.(root/msg/detector/error_list/error//@id)

odstraní duplikované hodnoty v (error_msg)
```

#43

množinově odečte aktuální \ známé (výsledek jsou nové)

poznámka: symbol " \ " značí množinové odčítání

```
detid_new = detid_msg \ detid

dettypes_new = dettypes_msg \ dettypes

status_new = status_msg \status

error_new = error_msg \ error
```

#45

jsou-li množiny nových hodnot `detid_new`, `dettypes_new`, `status_new`, `error_new` neprázdné, nastává přidání hodnot #46a, #46b, #46c nebo #46d podle toho, které množiny jsou neprázdné. Budou zde tedy 4 podobné podmínky, pro jednoduchost vývojového diagramu shrnuto do jedné.

#46a

vypíše zprávu pro editora o novém detektoru, pokud je jich více, pak pro každý m-tý:

zjistí jeho typ pomocí XPath příkazu `detid_new_type[m] = msg/detector[@id=detid_new[m]]/@type`

* pokud se jedná o detektor SCALA, je třeba dát pozor na mezery a apostrofy - pokud končí apostrofem, počet apostrofů je do SQL kódu nutné zdvojnásobit

přidá ho jako neaktivní pomocí SQL příkazu:

```
INSERT INTO detectors (id_det, type, active) VALUES (detid_new[m], detid_new_type[m],FALSE);
```

#46b

vypíše zprávu pro editora o novém typu detektoru, přidá ho pomocí SQL příkazu `INSERT INTO detectortypes ( type) VALUES (detid_new_type[m]);`

#46c

vypíše zprávu pro editora o novém stavu, přidá ho pomocí SQL příkazu

```
INSERT INTO statuslist (id_status) VALUES (status_new[m]);
```

#46d

vypíše zprávu pro editora o nové chybě, přidá ji pomocí SQL příkazu

```
INSERT INTO errorlist (id_error VALUES (error_new[m]);
```

Zpráva editorovi slouží k oznámení nového detektoru, typu, stavu či chyby. Editor by měl rozšířit popis přidané hodnoty, a v případě nového detektoru po rozšíření popisu jej aktivovat v manuálním módu Administrátorsko-Editorské aplikace.

#47

porovná čas: `DateTime.Now < endTime`

poznámka: `endTime` inicializován v bodě #41 na řádce programu 79

#50

```
detector_type == 'SCALA'
```

#51

```
letní_čas() == TRUE
```

#52

je alespoň jeden stav k dispozici?

```
n_max = count(msg/detector[@id='id_det' and
stop='stoptime']/status_list/status)
```

```
n_max>0
```

#53

```
msg/detector[@id='id_det' and
stop='stoptime']/status_list/status[n]/@value > 0
```

#54

```
value =msg/detector[@id='id_det' and
stop='stoptime']/status_list/status[n]/@value
```

#55

```
value = NULL
```

#56

inicializace:

```
previousstop = stoptime - interval
```

```
previousstatuses = SELECT id_status FROM statushistory WHERE
id_det = id_det AND stoptime=previousstop
```

```
actualstatus = msg/detector[@id='id_det' and
stop='stoptime']/status_list/status[n]/@id
```

```
previousvalue = SELECT valuestatus FROM statushistory WHERE
id_det = id_det AND stoptime=previousstop AND id_status =
actualstatus
```

```
actualvalue = msg/detector[@id='id_det' and
stop='stoptime']/status_list/status[n]/@value
```

hlavní podmínka:

```
(actualstatus je podmnožinou previousstatuses AND value=NULL)
OR (previousvalue == actualvalue AND value=NOT NULL)
```

#57

vloží se stav

```
INSERT INTO statushistory VALUES (DEFAULT, actualstatus,
id_det, previousstop, stoptime, actualvalue );
```

#58

prodlouží se stav

```
UPDATE statushistory SET "stopstatus" = stoptime WHERE id_det
= id_det AND stoptime=previousstop AND id_status =
actualstatus
```

#59

inicializace:

```
p_max = count(msg/detector[@id='id_det' and
stop='stoptime']/error_list/error)
```

hlavní podmínka:

```
p_max>0
```

#60

inicializace:

```
previousstop = stoptime - interval
```

```
previouserrors = SELECT id_error FROM errorhistory WHERE
id_det = id_det AND stoptime=previousstop
```

```
actualerror = msg/detector[@id='id_det' and
stop='stoptime']/error_list/error[n]/@id
```

hlavní podmínka: `actualerror je podmnožinou previouserrors`

#61

```
INSERT INTO errorhistory VALUES (DEFAULT, actualerror,
id_det, previousstop, stoptime);
```

#62

```
UPDATE errorhistory SET "stoperror" = stoptime WHERE id_det =
id_det AND stoptime=previousstop AND id_error = actualerror
```

#70

```
count(msg/detector[@id='id_det' and stop='stoptime' and
count/all>=0]) > 0
```

#71

```
DROP TABLE IF EXISTS validation_temp CASCADE;
```

```
CREATE TABLE validation_temp  AS SELECT days.StartDay +
'daytime' AS stop FROM days WHERE days.WeekDay=weekday AND
days.DayCharacter=daycharacter;
```

```
validdays_available = SELECT COUNT(dat.count_all)  FROM
dat,validation_temp WHERE validation_temp.stop = dat.stoptime
AND dat.validity=100
```

```
validdays_available > validdays_check
```

#72

```
current_intensity = msg/detector[@id='id_det' and
stop='stoptime']/count/all
```

```
average_intensity = SELECT AVG(dat.count_all)  FROM
dat,validation_temp WHERE validation_temp.stop = dat.stoptime
AND dat.validity=100 ORDER BY validation_temp.stop DESC LIMIT
validationdays
```

#73

externí funkce `GetTolerance (count)` vrátí procentuální odchylku `tolerance`

Pro nedostatek dat nebylo pravidlo pro určení tolerance určeno. Obecně bude platit, že u nižších intenzit bude relativní odchylka větší a u vyšších intenzit bude relativní odchylka vyšší.

hlavní podmínka: `current_intensity / average_intensity < 1 + tolerance AND current_intensity / average_intensity > 1 - tolerance`

#75

```
SELECT COUNT(count_all) FROM dat WHERE stoptime =
stoptime_update AND id_det=id_det > 0
```

#76

```
starttime = stoptime - interval
```

```
INSERT INTO dat (id,dat, id_det, raw, attemptnuber) VALUES
(DEFAULT, id_det, rawness, 0)
```

#77

```
starttime = stoptime - interval
```

zvýší číslo pokusu

```
pokus = SELECT "AttemptNumber" FROM dat
WHERE id_det = id_det AND stoptime = stoptime_update

pokus = pokus + 1
```

#78

zkontroluje stavy a chyby

```
check_status (source_data, id_det, stoptime_update, interval)

check_error (source_data, id_det, stoptime_update, interval)
```

#79

update čísla pokusu

```
UPDATE dat SET "AttemptNumber" = pokus
WHERE id_det = id_det AND stoptime = stoptime_update
```

#80

update SCALA

načtení dat XML -> program

```
count_all = msg/detector[@id='id_det' and
stop='stoptime']/count/all

speed_all = msg/detector[@id='id_det' and
stop='stoptime']/speed/all

occupancy_all = msg/detector[@id='id_det' and
stop='stoptime']/occupancy/all
```

uložení dat program -> DB

```
UDATE dat SET (  "id_det" , "raw" ,"validity" ,
"attemptnumber" ,  "stoptime" ,  "starttime" ,  "count_all"
,  "speed_all" ,"occcupancy_all" ) =
 (  id_det , raw  ,validity , pokus ,  stoptime ,  starttime
,  count_all ,  speed_all ,occcupancy_all )
```

#81

update CollectR

načtení dat XML -> program

```
count_all = msg/detector[@id='id_det' and
stop='stoptime']/count/all

count_passengercar = msg/detector[@id='id_det' and
stop='stoptime']/count/class[@id='1']

count_alltruck = msg/detector[@id='id_det' and
stop='stoptime']/count/class[@id='2']

speed_all = msg/detector[@id='id_det' and
stop='stoptime']/speed/all

speed_passengercar = msg/detector[@id='id_det' and
stop='stoptime']/speed/class[@id='1']

speed_alltruck = msg/detector[@id='id_det' and
stop='stoptime']/speed/class[@id='2']

occupancy_all = msg/detector[@id='id_det' and
stop='stoptime']/occupancy/all

temperature = msg/detector[@id='id_det' and
stop='stoptime']/status_list/status[@id='9']/@value

voltage = msg/detector[@id='id_det' and
stop='stoptime']/status_list/status[@id='8']/@value
```

uložení dat program -> DB

```
UDATE dat SET (  "id_det" , "raw" ,  "validity",
"attemptnumber" ,  "stoptime" ,  "starttime" ,  "count_all"
,  "count_passengercar",  "count_alltruck" ,  "speed_all" ,
"speed_passengercar" ,  "speed_alltruck",  "occcupancy_all"
,  "temperature" ,  "voltage" )
```

```
= (id_det , raw , validity, pokus, stoptime , starttime
, count_all , count_passengercar, count_alltruck ,
speed_all , speed_passengercar , speed_alltruck,
occcupancy_all ,

temperature , voltage )
```

#82

update Traficam

načtení dat XML -> program

```
count_all = msg/detector[@id='id_det' and
stop='stoptime']/count/all

occupancy_all = msg/detector[@id='id_det' and
stop='stoptime']/occupancy/all
```

uložení dat program -> DB

```
UDATE dat SET ( "id_det" , "raw", "validity" ,
"attemptnumber" , "stoptime" , "starttime" ,
"count_all","occcupancy_all" ) =
 ( id_det , raw ,validity , pokus , stoptime , starttime
, count_all,occcupancy_all )
```

PŘÍLOHA D.2 - Pomůcka k tvorbě zdrojového kódu Administrátorskoeditorské aplikace - manuální část

PŘÍLOHA D.2.1 Aktivace a deaktivace detektoru

Aktivace

aktivace detektoru které má `id_detektoru`

```
UPDATE detectors SET "active" = TRUE WHERE
id_det=id_detektoru
```

Deaktivace

deaktivace detektoru které má `id_detektoru`

```
UPDATE detectors SET "active"= FALSE WHERE id_det=id_detekto
```

PŘÍLOHA D.2.2 Vytvoření profilu

vytvoření profilu které má `id_profil` a `popis`

```
INSERT INTO profiles VALUES (id_profil,popis);
```

PŘÍLOHA D.2.3 Zařazení detektorů do profilu

zařazení detektoru které má `id_detektoru` do profilu které má `id_profil`

```
INSERT INTO det2profil VALUES (DEFAULT,id_detektor,
id_profil);
```

PŘÍLOHA D.2.4 Přidání sekcí

pro lepší pochopení je tento příklad názorný

přidání sekcí: profil evidovaný jako UITS-000 na ulici Bucharova z Rozvadovské směrem na Plzeňskou který TSK eviduje pomocí počátečního uzlu 5029 a koncového uzlu 5024.

SQL kód pro přidání sekce:

```
INSERT INTO sections VALUES (DEFAULT,'UITS-000', 'TSK',
'5029','5024', NULL);
```

PŘÍLOHA D.3 - Pomůcka k tvorbě zdrojového kódu Studentské aplikace

PŘÍLOHA D.3.1 Evidence dotazů

Zadaný dotaz `query` studentem `id_stu` v čase `logtime` lze vložit dle následujícího způsobu:

```
INSERT INTO querylog VALUES (DEFAULT, id_stu, logtime, query)
```

PŘÍLOHA D.3.2 Kontrola a vytvoření SQL dotazu

kontrola:

pro ověření zadaných parametrů `id_det , stoptime , starttime` tedy ověření, že jsou zadány správně, nejprve aplikace stáhne pole vše id_detektorů

```
alldet = SELECT id_det FROM detectors
```

a pokud se na úrovni aplikace ověří že `id_det je součástí alldet` tak je detektor schválen. U času lze využít kontrolu pomocí pevně umístěných znaků v datu (například sedmí znak je pomlčka atd.).

vytvoření dotazu:

```
SQL_query = SELECT * FROM dat WHERE id_det = id_det AND
starttime >  AND stoptime < stoptime
```

PŘÍLOHA D.3.3 Generování XML odpovědi

```
XML_response = SELECT
query_to_xml('SQL_query',false,false,'')
```

```
odstranění tagu(xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance")
```

```
vložení tagu(<?xml version="1.0" encoding="UTF-8"?>)
```

PŘÍLOHA D.4 - Pomůcka k tvorbě zdrojového kódu Pedagogické aplikace

PŘÍLOHA D.4.1 Vkládání studentů

Zadané id_studenta `id_stu` se jménem `firstname` a příjmením `surname` lze vložit dle následujícího způsobu:

```
INSERT INTO querylog VALUES (id_stu, firstname, surname)
```

PŘÍLOHA D.4.2 Přiřazení validního zadání

#100

načte id studentů které nemají zadání jako pole `"studenti"` o velikosti `stud_max`

```
SELECT id_stu FROM students WHERE students.id_stu NOT IN
(select id_stu FROM task)
```

načte profily detektorů které začínají kódem "UITS" do pole `"profily"` o velikosti `prof_max`

```
SELECT id_profil FROM profiles WHERE SUBSTRING(id_profil,
1,4)='UITS'
```

#101

nastaví `stud`=1

nastaví `prof`=1

zapíše do numerické proměnné "`rok`" aktuální rok

```
select days."WeekNumYear" from days WHERE "StartDay" <=
(select now()) AND "StopDay" >= (select now())
```

zapíše poslední celý týden s daty pro uits do numerické proměnné "`tyden`" =

```
select days."WeekNum" from days WHERE "StartDay" <= (select
max(stoptime) from dat, det2profil WHERE substring(id_profil,
```

```
1,4)='UITS' AND det2profil.id_det = dat.id_det AND
dat.validity>=80) AND "StopDay" >= (select max(stoptime) from
dat, det2profil WHERE substring(id_profil, 1,4)='UITS' AND
det2profil.id_det = dat.id_det AND dat.validity>=80)- 1
```

#102

zjistí začátek týdne zac_tydne=

```
SELECT  min(StartDay) FROM days WHERE WeekNum=tyden AND
WeekNumYear=rok
```

zjistí konec týdne kon_tydne=

```
SELECT  max(StopDay) FROM days WHERE WeekNum=tyden AND
WeekNumYear=rok
```

#103

zjistí jestli jsou v týdnu data

```
SELECT COUNT(validity) FROM dat,det2profil WHERE
det2profil.id_profil = profily[prof] AND det2profil.id_det =
dat.id_det AND dat.starttime<=zac_tydne AND
dat.stoptime>=kon_tydne >0
```

#104

zjistí celistvost zadání

```
integrtity = SELECT COUNT(validity) FROM dat WHERE
det2profil.id_profil = profily[prof] AND det2profil.id_det =
dat.id_det AND dat.starttime<=zac_tydne AND
dat.stoptime>=kon_tydne AND dat.validity=100 /(7 dní* 2
detektory * 288 dat denně) * 100
```

#105

Porovná zda poměr validních dat s maximálním možným počtem dat je větší jak 80%

```
integrity > 80
```

#106

```
Zapíše zadání
```

```
INSERT INTO task VALUES( DEFAULT, studenti[stud],
profily(prof], rok, tyden, integrity);
```

#107

```
Mají všichni zadání?
```

```
stud==stud_max
```

#108

```
Vytiskne zadání
```

```
SELECT task.id_task, task.id_stu, students."FirstName",
students."LastName", id_profil, task.year, task.week,
task.integrity FROM task, students WHERE
task.id_stu=students."id_stu" AND WHERE task.year=rok
```

#109

Přeskočí na dalšího studenta

```
stud=stud+1
```

#110

Jedná se o poslední profil?

```
prof==prof_max
```

#111

Přeskočí na další profil

```
prof=prof+1
```

#112

Začne poslední profil předchozího týdne

```
prof=1

tyden=tyden-1
```

PŘÍLOHA D.4.3 Generování grafu

Vytiskne graf dat získané SQL dotazem:

```
SELECT count_all FROM dat WHERE starttime>odkdy AND
stoptime=odkdy AND id_det=iddet
```

kde:

```
odkdy =

SELECT MIN(days.startday) FROM days, task WHERE
task.id_task=Id_ulohy AND task.year=days.weeknumyear AND
task.week=days.weeknum

dokdy =

SELECT MAX(days.stopday) FROM days, task WHERE
task.id_task=Id_ulohy AND task.year=days.weeknumyear AND
task.week=days.weeknum
```

iddet jsou postupně:

```
SELECT id_det FROM det2profil WHERE id_profil=profil
```

v profilu

```
SELECT id_profil FROM task WHERE id_task=Id_ulohy
```

PŘÍLOHA D.4.4 Zobrazení evidence dotazů

Podle Id_studenta:

```
SELECT * FROM querylog WHERE id_stu=Id_studenta
```