Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra informatiky a výpočetní techniky

Bakalářská práce

Systém pro mobilní zařízení pro manuální zaznamenávání průjezdů vozidel křižovatkou

Plzeň, 2012 Martin Kožíšek

PROHLÁŠENÍ

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 1. 5. 2012 ……...

Martin Kožíšek

PODĚKOVÁNÍ

Za odborné vedení, vstřícný přístup a věcné připomínky při tvorbě této práce děkuji panu Ing. Tomáši Potužákovi Ph.D. Dále děkuji panu Ing. Ladislavu Pešičkovi za cenné rady týkající se vývoje aplikací pro mobilní zařízení.

ABSTRACT

The objective of this work is to develop a system that provides the user statistics about traffic crossroads especially about branching probabilities. It involves the creation of two applications. First one is for mobile devices and its purpose is to enable manual recording of passages of the vehicles through a traffic crossroad in particular traffic lanes. The second appliaction processes the data from mobile application and displays statistics.

The first theoretical part of this thesis is about traffic simulation in general, programming for mobile devices (especially for Android) and about XML. The second part describes the analysis, implementation and testing of the applications.

OBSAH

PROHLÁŠENÍ **2**
PODĚKOVÁNÍ **3**
ABSTRACT **4**
OBSAH **5**
1 ÚVOD **7**
2 POČÍTAČOVÁ SIMULACE **8**
2.1 CO JE POČÍTAČOVÁ SIMULACE 8
2.2 MODEL 8
2.3 ROZDĚLENÍ 8
2.3.1 Podle účelu simulace *8*
2.3.2 Podle času *9*
2.4 SIMULACE DOPRAVY 9
2.4.1 K čemu slouží *9*
2.4.2 Rozdělení podle míry abstrakce *10*
2.4.3 Simulační čas *10*
2.4.4 Dopravní modely *10*
2.4.5 JUTS *11*
2.4.6 DUTS *11*
2.4.7 Souvislost *12*
3 VÝVOJ APLIKACÍ PRO MOBILNÍ ZAŘÍZENÍ **13**
3.1 OBECNÉ INFORMACE 13
3.2 BEZPEČNOST 13
3.3 ODLIŠNOSTI OPROTI VÝVOJI DESKTOPOVÝCH APLIKACÍ 14
3.4 VÝBĚR PLATFORMY PRO MOBILNÍ APLIKACI 14
3.4.1 Symbian *15*
3.4.2 iOS *15*
3.4.3 Windows Phone 7 ..... *16*
3.4.4 Android *16*
4 ANDROID **17**
4.1 OBECNÉ INFORMACE 17
4.2 VERZE 17
4.3 APLIKACE 17
4.4 VÝVOJ APLIKACÍ 17
4.4.1 Aplikace *18*
4.4.2 Struktura projektů *19*
4.4.3 Layout *19*
4.4.4 Oprávnění *19*
4.4.5 Podepisování aplikací *20*
5 PŘENOS DAT MEZI APLIKACEMI **21**

5.1 Možnosti ... 21
5.2 XML ... 21
5.2.1 Zpracování XML ... 22
6 Analýza ... 23
6.1 Popis křižovatky ... 23
6.1.1 Atributy křižovatky ... 23
6.1.2 Atributy ramen ... 24
6.1.3 Atributy vjezdových pruhů ... 24
6.1.4 Atributy výjezdových pruhů ... 25
6.2 XML soubor s popisem křižovatky ... 25
6.2.1 Struktura ... 25
6.3 Mobilní aplikace ... 26
6.3.1 Specifikace požadavků ... 26
6.3.2 Vykreslení křižovatky ... 26
6.3.3 Návrh GUI ... 27
6.4 Desktopová aplikace ... 27
6.4.1 Specifikace požadavků ... 27
6.4.2 Zobrazované statistiky ... 28
6.4.3 Návrh GUI ... 28
7 Implementace ... 29
7.1 Mobilní aplikace ... 29
7.1.1 GUI ... 29
7.1.2 Programový kód ... 31
7.2 Desktopová aplikace ... 34
7.2.1 Datová reprezentace křižovatky ... 34
7.2.2 Generování XML souborů ... 35
7.2.3 Načítání XML souborů ... 35
7.2.4 Výpočet statistik ... 36
7.2.5 GUI ... 36
7.2.6 Grafika ... 38
7.2.7 Vykreslování grafů ... 39
8 Testování aplikací ... 40
8.1 Křižovatka u náměstí Míru ... 40
8.2 Křižovatka v Přešticích ... 41
9 Závěr ... 43
Přehled zkratek ... 44
Literatura a zdroje ... 45
Příloha A – Uživatelská příručka k mobilní aplikaci ... 46
Příloha B – Uživatelská příručka k desktopové aplikaci ... 49
Příloha C – UML diagramy ... 51
Příloha D – Ukázka XML souboru s křižovatkou ... 53

1 Úvod

Podle statistik Sdružení automobilového průmyslu se počet osobních aut na českých silnicích za 20 let od roku 1989 téměř zdvojnásobil. Se vzrůstajícím počtem aut vzrůstají i nároky na řízení dopravy, jehož úkolem je zajištění co možná nejlepší průjezdnosti městských částí, snižování prostojů na křižovatkách a eliminace dopravních zácep. Chceme-li efektivně řídit složitý dopravní systém, potřebujeme vědět, jak se zachová při změně určitých parametrů. Pro takové případy je vhodná počítačová simulace, pomocí níž můžeme namodelovat různé situace a zjištěné poznatky o chování systému následně aplikovat při skutečném řízení dopravy.

Rychle se zvyšující výkon výpočetní techniky umožňuje stále detailnější simulace, které ve svém výpočtu zahrnují až jednotlivá vozidla. Příkladem je mikroskopický simulátor JUTS vyvinutý na Katedře informatiky a výpočetní techniky ZČU v Plzni, který simuluje dopravní situaci ve městě. K tomu potřebuje přesná statistická data, mezi něž patří pravděpodobnosti odbočování vozidel na jednotlivých křižovatkách. Vzhledem k tomu, že senzory pod povrchem vozovky poskytují údaje pouze o počtu vozidel projetých daným pruhem (a nesledují už, jakým směrem vozidla pokračovala), je nutné k získání pravděpodobností odbočení pozorovat danou křižovatku a ručně zaznamenávat průjezdy vozidel. S rychlým vývojem mobilních technologií a příchodem chytrých telefonů (smartphonů) se naskýtá možnost, využít potenciálu jejich dotykových displejů a tím proces manuálního měření usnadnit.

Výsledkem této práce by měl být systém, který poskytne informace o odbočování vozidel na křižovatce. To zahrnuje vytvoření dvou aplikací. Tou první je aplikace pro mobilní zařízení umožňující zaznamenávání vjezdového a výjezdového pruhu pro vozidla projíždějící křižovatkou. Druhou částí je desktopová aplikace, v níž se budou jednak vytvářet vstupní data pro mobilní aplikaci, jednak bude zpracovávat již naměřená data a zobrazovat pravděpodobnosti odbočení.

2 POČÍTAČOVÁ SIMULACE

Nejprve v krátkosti zmiňme základní informace o počítačové simulaci.

2.1 Co je počítačová simulace

Počítačová simulace je imitace reálného nebo hypotetického systému a jeho chování v průběhu času s využitím výpočetní techniky. V obou případech je účelem simulace umožnit nám vyvozovat poznatky o chování systému v určitých situacích, aniž bychom s ním museli manipulovat, nebo ho sledovat [HPN96].

2.2 Model

Základem simulace je model, který popisuje vlastnosti a chování zkoumaného systému. Tímto modelem je část reálného světa zahrnující pouze ty vlastnosti a rysy, které jsou v daném případě důležité. Každý model je totiž pouze zjednodušením reálného systému, neboť jsme vždy limitováni zdroji (výpočetní výkon, čas). Je proto velmi důležité určit, které vlastnosti mají v daném případě podstatný vliv na chování systému a které naopak ovlivňují chování systému v zanedbatelné míře a nemusíme je tak uvažovat. Budeme-li chtít například simulovat dopravu na určité křižovatce, budeme muset do modelu zahrnout informace o počtu ramen křižovatky, počtu pruhů atd. Naopak nás pravděpodobně nebude zajímat výška okolních budov.

2.3 Rozdělení

Počítačové simulace lze klasifikovat na základě několika hledisek. Těmi nejdůležitějšími jsou dělení podle účelu simulace a dělení podle toho, jak plyne simulační čas [FMT00].

2.3.1 Podle účelu simulace

Analytická simulace

Simulace, která se používá k modelování existujícího nebo vyvíjeného systému z reálného světa. Zkoumá chování systému v současných podmínkách, případně jak se bude chování měnit při změně podmínek. K tomu, aby simulace odpovídala skutečnosti, potřebuje přesná statistická data z modelovaného systému. Výsledky pak mohou být použité pro vylepšení systému [FMT00].

Virtuální prostředí

Simulace, která vytváří počítačem generovaný svět podobný tomu reálnému. Nejběžnějším použitím je trénink (letecké simulátory pro piloty) nebo zábava (3D akční hry). Je zřejmé, že taková simulace musí být vykonávána v reálném čase, aby byla pro člověka věrohodná [FMT00].

2.3.2 Podle času

Z hlediska času lze simulace dělit na **spojité** a **diskrétní**. V případě spojitých simulací je chování systému pospáno diferenciálními rovnicemi, na jejichž základě se spojitě mění simulační stav. Diskrétní simulace lze dále rozdělit na simulace s pevnými časovými kroky a událostmi řízené simulace [FMT00].

Pevné časové kroky

V tomto případě je simulační čas rozdělen na stejně velké časové intervaly (kroky). V každém kroku jsou pak přepočteny hodnoty všech stavových proměnných, čímž dojde ke změně simulačního stavu.

Událostmi řízená simulace

Na rozdíl od simulace s pevnými časovými kroky, je zde čas realizován spouštěním událostí podle toho, jak jsou naplánovány v seznamu událostí. Každá událost obsahuje akci, což je inkrementální změna stavu simulace.

2.4 Simulace dopravy

Nyní se budeme zabývat konkrétně simulací dopravy.

2.4.1 K čemu slouží

Se vzrůstajícím počtem aut vzrůstají, především ve městech, nároky na řízení dopravy, jehož úkolem je zajištění co možná nejplynulejšího provozu. Pro efektivní řízení složitého dopravního systému potřebujeme vědět, jak se zachová v různých situacích. Například změní-li se nastavení intervalů na semaforech křižovatky, postaví-li se nový kruhový objezd, nebo omezí-li se rychlost na určitém úseku. Praktické experimenty jsou většinou nerealizovatelné. Právě pro takové případy je vhodná počítačová simulace, pomocí níž můžeme namodelovat různé situace a zjištěné poznatky o chování systému následně aplikovat při skutečném řízení dopravy. Je tedy možné nejen lépe řídit stávající dopravní síť, ale také předvídat důsledky výstavby nových silnic nebo jiných objektů významně ovlivňujících dopravní situaci (např. nákupních center).

2.4.2 Rozdělení podle míry abstrakce

Základní metodou klasifikace simulací dopravy je dělení podle míry abstrakce [HAR03]:

- Makroskopické
- Mezoskopické
- Mikroskopické

Makroskopické simulace

Jedná se o nejjednodušší typ simulací s nejvyšší úrovní abstrakce. Nepočítá s jednotlivými vozidly, ale pouze s dopravními proudy v ulicích reprezentované proměnnými, jako jsou tok, hustota a střední rychlost (tyto pojmy jsou podrobně vysvětleny v [HAR03]). Vzhledem k malé přesnosti jsou makroskopické simulace použitelné jen ve speciálních případech.

Mezoskopické simulace

Mezoskopické simulace již většinou reprezentují všechny entity jako mikroskopické, ale interakce mezi nimi popisují na menší úrovni detailu [HAR03].

Mikroskopické simulace

Mikroskopické simulace uvažují jak entity (jednotlivá vozidla) systému, tak interakci mezi nimi. Například při změně pruhu jsou brána v potaz okolní vozidla a může zahrnovat i rozhodování řidiče a rychlost jeho reakcí. Tyto simulace se nejvíce přibližují realitě, ale jejich nevýhodou je vysoká výpočetní náročnost a složitá implementace [HAR03].

2.4.3 Simulační čas

Základní mechanismy běhu simulačního času byly popsány v kapitole 2.3.2. V oblasti dopravních simulací je mnohem používanějším mechanismus pevného časového kroku, kde je simulační čas rozdělen na stejně velké intervaly dlouhé typicky jednu sekundu. V každém kroku jsou přepočítány stavové proměnné, což například u mikroskopické simulace znamená změnu pozic jednotlivých vozidel na základě jejich aktuální rychlosti.

2.4.4 Dopravní modely

Zaměříme-li se na mikroskopické simulace s pevnými časovými kroky, je ve většině případů používán buď model buněčných automatů, nebo model vodícího vozidla [POT09].

Model buněčných automatů

V tomto modelu jsou jízdní pruhy rozděleny na stejně velké 7,5 metru dlouhé buňky, přičemž každá buňka může být buď prázdná, nebo obsazená vozidlem. Pozice vozidel jsou v každém časovém kroku přepočítány na základě jejich rychlosti. Pohyb vozidla se skládá ze čtyř kroků – zrychlení, zpomalení, randomizace, posun vozidla. Více v [POT09].

Model vodícího vozidla

Na rozdíl od modelu buněčných automatů zde nejsou jízdní pruhy rozděleny na buňky a vozidlo tak může být umístěno v pruhu libovolně. První vozidlo v pruhu se snaží zrychlit na svoji maximální rychlost, pokud nenarazí na překážku. Další vozidlo v pruhu také zrychluje, ale musí svoji rychlost přizpůsobit předcházejícímu vozidlu. [POT09].

2.4.5 JUTS

JUTS (*Java Urban Traffic Simulator*) je projekt zpracovávaný v rámci bakalářských, diplomových a disertačních prací na Katedře informatiky a výpočetní techniky (KIV) na Fakultě aplikovaných věd (FAV) Západočeské univerzity (ZČU) jako jeden z projektů výzkumné skupiny Distribuované systémy, softwarové inženýrství a simulace (DSS).

Jedná se o pseudoparalelní simulační nástroj s grafickým výstupem zaměřený na městskou dopravní síť. Je použita diskrétní simulace založená na pevných časových krocích a mikroskopické úrovni abstrakce.

Umožňuje sledovat simulaci dopravy na mapě, upravovat její parametry a tím modelovat různé situace (např. uzavření některého pruhu, nebo celé komunikace). Více o projektu JUTS lze najít na [WEB3].

2.4.6 DUTS

DUTS (*Distributed Urban Traffic Simulator*) je další projekt zpracovaný na Katedře informatiky a výpočetní techniky. Jedná se o distribuovaný simulátor městské dopravy na mikroskopické úrovni detailu, který je určen pro testování různých komunikačních protokolů v distribuovaném prostředí, ale dá se zároveň použít i jako klasický dopravní simulátor.

Simulátor obsahuje dva modely buněčných automatů a jeden založený na modelu vodícího vozidla. K dispozici je přes deset různých komunikačních protokolů pro komunikace mezi jednotlivými procesy distribuované simulace.

Ze statistik (kromě těch, které jsou zaměřeny na meziprocesovou komunikaci) je možné zjistit i údaje o počtu vozidel v jednotlivých pruzích a křižovatkách [WEB3].

2.4.7 Souvislost

Jak již bylo zmíněno, v obou případech se jedná o mikroskopickou simulaci, což znamená, že uvažujeme jednotlivá vozidla. Vozidlu v simulaci každá křižovatka (kterou projíždí) určí, kterým směrem se má vydat. Tento směr ale nemůže být vygenerován zcela náhodně, neboť by to neodpovídalo reálnému světu, kde je každá komunikace jinak vytěžovaná a od toho se odvíjí i to, kterým směrem se vydá vozidlo přijíždějící do křižovatky. Proto si každá křižovatka uchovává ve svých parametrech pravděpodobnosti odbočení pro jednotlivá ramena a pruhy. Na základě těchto pravděpodobností je určen směr, kterým je vozidlo projíždějící křižovatkou posláno.

Nástrojem pro získávání a zpracování potřebných dat z reálné křižovatky budou aplikace popsané ve druhé části této práce.

3 VÝVOJ APLIKACÍ PRO MOBILNÍ ZAŘÍZENÍ

3.1 Obecné informace

Jestliže vývoj v oblasti informačních technologií nabral v posledních desetiletích raketové tempo, pro mobilní zařízení to platí dvojnásob. V posledních letech navíc velký boom v oblasti mobilních technologií znamenal příchod chytrých mobilních telefonů (smartphonů). Smartphone je mobilní telefon, který disponuje vlastním operačním systémem a uživateli nabízí pokročilé funkce a možnost instalace dalších aplikací. Většinou se také vyznačuje dotykovým displejem. Kromě telefonování a psaní SMS zpráv uživatelé stále více využívají mobilní telefony pro práci s internetem (prohlížení webových stránek, komunikace přes sociální sítě, přijímání a odesílání emailů), jako GPS navigaci, MP3 přehrávač atd. Pro uživatele je jednou z nejdůležitějších vlastností smartphonů právě možnost doinstalování potřebných aplikací, které značně rozšíří funkcionalitu telefonu.

Z hlediska vývojářů má ale velká rozmanitost a rychlý pokrok v oblasti mobilních zařízení i své stinné stránky. Zatímco na osobních počítačích a noteboocích je jednoznačně nejrozšířenějším operačním systémem Windows a v menší míře se vyskytují různé linuxové distribuce, na mobilních zařízeních je situace mnohem rozmanitější a nejednoznačnější. Nepříjemnost spočívá v tom, že není možné vyvíjet platformě nezávislé aplikace, jelikož se pro každou platformu většinou používají jiné vývojové nástroje, jiné programovací jazyky a odlišné filosofie překladu a spouštění aplikací. Programátor proto musí učinit rozhodnutí, pro jakou platformu bude vyvíjet. Situaci navíc komplikuje fakt, že procentuální zastoupení jednotlivých platforem na mobilních zařízeních se velmi rychle a výrazně mění a volba vhodné platformy se proto neobejde bez rizika „šlápnutí vedle“. To může být vidět na příkladu *Symbianu*, který byl v roce 2010 nejrozšířenější platformou na smartphonech, ale v současné době (rok 2012) se jedná v podstatě o „mrtvý“ systém.

Pozitivem naopak je, že vývojové nástroje jsou většinou zdarma, jelikož se tvůrci operačních systémů snaží vyjít vstříc vývojářům. Uvědomují si totiž, že úspěšnost systému v dnešní době přímo souvisí s množstvím dostupných aplikací.

3.2 Bezpečnost

Vzhledem k zvyšující se funkcionalitě, inteligenci a také stále většímu využívaní připojení k internetu, se mobilní zařízení stávají stále častěji terčem útoků. Proto je nutné klást čím dál větší důraz na otázku bezpečnosti. Viry na mobilním zařízení mohou navíc mít pro uživatele horší důsledky, než viry na klasickém stolním počítači. Mohou

totiž kromě běžných škodlivých činností způsobit navíc uživateli i finanční ztráty, které mohou být důsledkem rozesílání prémiových SMS, vytáčení prémiových čísel, případně stahování nevyžádaného obsahu.

Tvůrci operačních systému proto vytvářejí různé ochranné mechanismy, například povolení stahování aplikací pouze přes oficiální služby (kde jsou aplikace kontrolovány), nebo zamezení přístupu k systémovým zdrojům, přístupu k internetu atd., pokud to není uživatelem explicitně povoleno [MIK07]. Otázkou bezpečnosti na mobilních zařízeních se také stále více zabývají renomované společnosti, které vyvíjejí antivirové programy. V dnešní době proto existují mobilní aplikace od společností *McAfee*, *Kaspersky*, *AVG Technologies*, *Avast software* a další. Tyto antivirové aplikace poskytují klasické funkce – skenování souborů, ochranu proti malwaru a spywaru, firewall, bezpečné procházení webu a mnohé další. Kromě toho navíc některé programy nabízejí ochranu proti ztrátě nebo krádeži, kdy má uživatel možnost zařízení lokalizovat, případně na dálku zablokovat (s možností například vypsat text s informací, kam má případný nálezce zařízení vrátit).

3.3 Odlišnosti oproti vývoji desktopových aplikací

Při vývoji mobilních aplikací musí programátor mnohem více dbát na záležitosti, které plynou z omezené velikosti mobilních zařízení. Zejména je třeba brát v potaz mechanické i hardwarové limity mobilních zařízení, pro které je aplikace určena [MIK07]. Displej má mnohem menší rozměry než u stolních počítačů, navíc ovládání prsty není tak přesné jako ovládání myší. Stejně tak je mnohem horší výkon procesoru a velikost paměti. Navíc musí být aplikace schopna reagovat na události specifické pro mobilní zařízení. Například při příchozím hovoru by měla aplikace bez problému přerušit svoji činnost a přejít do pozadí, aby po skončení hovoru opět pokračovala tam, kde skončila.

3.4 Výběr platformy pro mobilní aplikaci

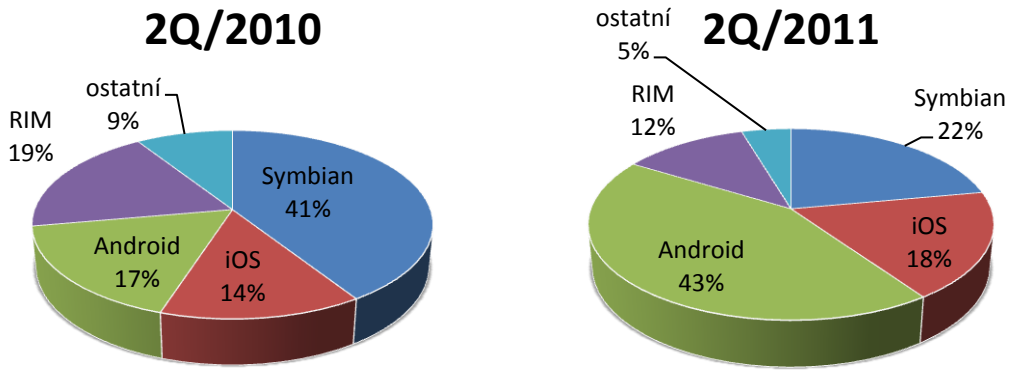
Jak již bylo zmíněno v kapitole 3.1, před započetím vývoje mobilní aplikace je nejprve nutné vybrat cílovou platformu. K tomu je potřeba si ujasnit kritéria, která jsou v našem případě důležitá.

Hlavní kritéria pro výběr:

- vybraná platforma by měla být dostatečně rozšířená,
- je nutné vybrat platformu, u které bude možnost otestovat funkčnost aplikace na reálném zařízení.

Na chytrých mobilních telefonech se objevovaly (na podzim roku 2011) především 4 operační systémy (specifické telefony BlackBerry s OS RIM neuvažujeme, neboť by pro vyvíjenou aplikaci nebyly vhodné).

Jak vypadá podíl operačních systémů na mobilních zařízeních, a jak se vyvinul v průběhu jednoho roku, ukazují grafy na obrázku 3.1.



Obr. 3.1: Podíl operačních systémů na trhu mobilních zařízení (zdroj: Gartner)

3.4.1 Symbian

Telefony: Nokia

Jazyk: J2ME

Klady a zápory:

+ pro vývoj by byla použita J2ME, což bylo původním plánem,
– společnost Nokia, která používá ve svých produktech OS Symbian v první půlce roku 2011 oznámila, že od tohoto systému přechází na Windows Phone 7.

Závěr: Vzhledem k tomu, že se Symbian stal v podstatě „mrtvým“ a i z grafů je patrný jeho úbytek v mobilních zařízeních, nemělo by smysl pro tento systém aplikaci vyvíjet.

3.4.2 iOS

Telefony: Apple iPhone

Jazyk: Objective C

Klady a zápory:

+ telefony iPhone jsou obecně považovány za kvalitní zařízení,
– z důvodu své vysoké ceny nejsou příliš rozšířeny, což platí dvojnásob pro uživatele, kteří budou s největší pravděpodobností s aplikací pracovat – studenty

- aplikace pro iPhone lze vyvíjet pouze ve vývojovém prostředí Xcode, které je ale dostupné pouze pro Mac OS X, což je operační systém na počítačích od firmy Apple,
- iOS je velmi uzavřený systém a instalace aplikací je (bez zásahu do systému - Jailbreak) možná pouze přes oficiální AppStore, což ztěžuje testování.

Závěr: Vyvíjet pro iOS by vzhledem k výše uvedeným nevýhodám nebylo možné a ani by to, vzhledem k cílové skupině uživatelů, nebylo rozumné.

3.4.3 Windows Phone 7

Telefony: Nokia

Jazyk: .NET CF

Klady a zápory:

+ je pravděpodobné, že v budoucnu bude patřit mezi tři nejrozšířenější OS,
- v současnosti tvoří tento systém jen malé procento v rozvržení OS na mobilech.

3.4.4 Android

Telefony: Samsung, HTC, Sony Ericsson a další

Jazyk: upravená Java

Klady a zápory:

+ v současnosti jasně nejrozšířenější systém na chytrých mobilních telefonech,
+ jeho procentuální podíl se neustále zvyšuje (což je vidět i z grafu)
+ snadný vývoj aplikací
+ bezproblémové otestování na reálném zařízení.

Závěr k výběru

Po vyhodnocení jednotlivých možností se v naší situaci jeví jako použitelné systémy Windows Phone 7 a Android. Při pohledu na výhody a nevýhody těchto dvou systémů je jednoznačným vítězem Android, a proto bude naše aplikace vyvíjena právě pro platformu Android.

4 ANDROID

4.1 Obecné informace

Android je rozsáhlá open-source platforma pro mobily a jiná zařízení (například tablety). Je založen na linuxovém jádře verze 2.6, které zajišťuje správu procesů, paměti, síťování a bezpečnost. Je vyvíjen konsorciem *Open Handset Alliance* vedeným společností *Google* [WEB1].

Vize byla následující: Google vyvine platformu, která umožní výrobcům získat OS s nižšími náklady, vývojářům nabídne pohodlný vývoj a zároveň poskytne kanál pro distribuci nových aplikací [WEB2].

První telefon s OS Android byl na trh uveden na přelomu let 2008 a 2009, jedná se tudíž o poměrně mladou platformu, která se ale během tří let stala nejrozšířenější na mobilních telefonech. Na obrázku 4.1 je zelený maskot Androidu.

Obr. 4.1: Maskot Androidu (zdroj: android.com)

4.2 Verze

V současnosti (1Q/2012) je poslední verzí Android 4.0 s kódovým označením *Ice Cream Sandwich*, který byl vydán koncem roku 2011, ale v současné době je jen na necelých 3% zařízení s Androidem. Nejrozšířenější verzí je stále Android 2.3 s označením *Gingerbread*, jehož podíl činí téměř 65% [WEB1].

4.3 Aplikace

Aplikace jsou psány v programovacím jazyce *Java*. Java Source Code je Java kompilátorem převeden na Java Byte Code (soubory `.class`), který je následně Dalvik kompilátorem převeden na Dalvik Byte Code (soubory `.dex`). Každá aplikace v systému běží jako samostatný proces a má svoji instanci *Dalvik Virtual Machine*, ve které jsou spuštěny soubory ve formátu `.dex`. Dalvik je virtuální stroj, který byl přizpůsoben požadavkům mobilních zařízení – úspora energie, omezená paměť, omezený výkon CPU [WEB2].

4.4 Vývoj aplikací

Android aplikace využívají syntaxi programovacího jazyka Java, knihovnu tříd, která je podmnožinou knihovny *Java SE* a navíc některá rozšíření specifická pro systém

Android [MUR10]. Nejprve je potřeba stáhnout a nainstalovat balíček *Android Software Development Kit (SDK)*, který je dostupný pro Windows, Linux i Mac. Pro správu nainstalovaného softwaru má vývojář k dispozici 2 programy.

SDK Manager

Slouží ke správě nainstalovaných balíků, stahování nových verzí platformy, updatů a dalších rozšíření.

AVD Manager (Android Virtual Device)

Slouží k vytváření vlastních virtuálních zařízení, která se spouštějí v emulátoru. Je možnost nakonfigurovat libovolné Android zařízení s možností nastavení celé řady parametrů.

Lze nastavit například:

- verzi cílové platformy (API Level),
- velikost paměťové karty,
- velikost RAM paměti,
- rozlišení displeje,
- hustotu bodů displeje.

Díky tomu je možné otestovat vyvíjenou aplikace na zařízeních s různými verzemi Androidu a s různými hardwarovými parametry.

Pro komunikaci se spuštěným emulátorem je k dispozici nástroj *Dalvik Debug Monitor Service (DDMS)*. Pomocí něj můžeme prohlížet informace o událostech (log výpisy), simulovat příchozí hovor nebo SMS zprávu, kopírovat soubory do a ze zařízení, aktualizovat zeměpisnou polohu zasíláním GPS souřadnic atd.

Silně doporučená je instalace *ADT pluginu* pro vývojové prostředí Eclipse. To poskytuje vývojáři například grafický návrhář, pomocí něhož je možné vytvořit grafické uživatelské rozhraní (GUI) metodou „drag and drop“. Výhodou je také integrovaný AVD a SDK Manager a DDMS, což vývojáři přináší snadný přístup ke všem potřebným nástrojům z jednoho místa.

4.4.1 Aplikace

Základním stavebním blokem každé aplikace je aktivita (Activity), jenž představuje jednu obrazovku uživatelského rozhraní. Celá aplikace se skládá z jedné, nebo několika aktivit. Každá aktivita obsahuje metodu `onCreate()`, která je volána při vytvoření, a dále může obsahovat metody (např.: `onPause()`, `onRestart()`, `onDestroy()`), které jsou volány při přechodu mezi stavy aktivity vyvolané například přerušením aplikace při příchozím hovoru.

4.4.2 Struktura projektů

Projekty pro Android mají specifickou, ne úplně triviální, stromovou strukturu adresářů a souborů, kterou za nás ale vývojové prostředí při vytváření nového Android projektu vygeneruje.

Význam některých důležitých položek:

- `src` – složka se zdrojovými kódy aplikace,
- `gen` – složka se zdrojovými kódy vygenerovanými nástroji systému Android,
- `bin` – složka obsahující zkompilovanou aplikaci,
- `res` – složka, která uchovává prostředky,
  - `drawable` – složka pro umístění obrázků, ikon atd.,
  - `layout` – obsahuje XML soubory definující uživatelské rozhraní,
  - `menu` – obsahuje XML soubory definující menu,
  - `values` – obsahuje XML soubory definující řetězce používané v aplikaci,
- `AndroidManifest.xml` – XML soubor, který popisuje aplikaci (jméno balíku, cílový API Level, požadovaná práva atd.) a aktivity, ze kterých se aplikace skládá.

4.4.3 Layout

Android používá návrh založený na XML, což znamená, že nastavení vlastností ovládacích prvků (widgetů) a jejich vzájemných vztahů je zapsáno v XML formátu a tím také odděleno od programové logiky. Souborů s layoutem může být více, přičemž ten, podle kterého je sestaveno GUI, je vybrán při vytvoření aktivity například v závislosti na orientaci telefonu (landscape[1], portrait[2]), nebo denní době (noční režim, denní režim). Jak již bylo výše zmíněno, pomocí při tvorbě GUI je grafický návrhář, díky kterému není nutné psát obsah XML souboru, ale lze si návrh GUI „naklikat“. Pravdou ovšem je, že se vývojář při dolaďování mnohých detailů často nevyhne explicitní úpravě XML souboru. Android samozřejmě umožňuje provádět veškeré operace týkající se uživatelského rozhraní i ze zdrojového kódu, ale tato varianta se používá obvykle jen v případech, kdy je nutné vytvářet a měnit jednotlivé ovládací prvky dynamicky za běhu aplikace.

4.4.4 Oprávnění

S bezpečnostních důvodů nemá žádná aplikace implicitně přístup k jakýmkoliv zdrojům, ať už je tím míněn zápis a čtení z paměťové karty, nebo přístup k internetu.

[1] Mód, kdy je telefon orientován na šířku.

[2] Mód, kdy je telefon orientován na výšku.

Pokud aplikace vyžaduje nějaký zdroj, musí si o něj zažádat prostřednictvím souboru `AndroidManifest.xml`.

Ukázka – žádost o oprávnění k zápisu na externí úložiště:

```
<uses-permission
     android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

Při instalaci aplikace je uživatel upozorněn jaké zdroje jsou požadovány a má možnost je buď všechny potvrdit, nebo odmítnout a tím pádem aplikaci nenainstalovat.

4.4.5 Podepisování aplikací

Systém Android vyžaduje, aby byly všechny aplikace podepsané certifikátem s privátním klíčem vývojáře dané aplikace. Jedná se o další bezpečnostní prvek, který slouží k snadné identifikaci autora.

Aby bylo možné testovat aplikace při vývoji, jsou v této fázi podepisovány tzv. „debug klíčem“. S tím ovšem nelze aplikace publikovat, navíc po relativně krátké době vyprší. Proto je nutné při exportu finálního instalačního balíku přiložit vlastní privátní klíč vygenerovaný například nástroji Keytool a Jarsigner [WEB1].

5 PŘENOS DAT MEZI APLIKACEMI

Vzhledem k tomu, že se systém budou tvořit dvě aplikace, je nutné zvolit, v jakém formátu budou data mezi aplikacemi přenášena.

5.1 Možnosti

Řešíme-li otázku, jak převést data mezi aplikacemi, máme 3 základní možnosti [HER07]:

- binární soubor,
- textový soubor,
- XML dokument.

Výhoda binárních souborů je v úsporném zápisu (velikosti souboru), má ale řadu zásadních nevýhod – např. data nejsou nijak strukturována. Vzhledem k tomu, že objem přenášených dat nebude nijak závratný, nemá smysl o binárním souboru uvažovat. Ani textový soubor nemá data nijak strukturována. Oproti tomu v XML dokumentu je zřejmý význam uložených dat, nejsou problémy s kódováním a navíc existují knihovní programy, které zpracování XML dokumentu usnadní. Pro přenos dat bude proto použit jazyk XML.

5.2 XML

XML (*eXtensible Markup Language*) je standardem W3C (*World Wide Web Consortium*) a v současné době je jedním z nejdůležitějších formátů výměny dat strukturovaným způsobem. Jedná se o rozšířený značkovací jazyk vhodný pro uchovávání a zpracovávání textových dokumentů. Mezi jeho hlavní výhody patří například:

- strukturovanost dat,
- pomocí značek je datům přiřazen význam,
- nezávislost na platformě,
- existují nástroje (*parsery*), které usnadňují zpracování XML dokumentů.

XML dokument se skládá z textového obsahu ohraničeného značkami (tagy), které lze libovolně pojmenovávat. Ke každé počáteční značce musí existovat koncová značka, dohromady se spolu s informací mezi nimi označují jako element. Ona informace se pak nazývá hodnota elementu. Elementy mohou navíc obsahovat atributy ve tvaru `název="hodnota"` umístěné v počáteční značce.

V předešlých řádcích byla popsána pouze základní terminologie využívaná v následujících kapitolách. Podrobnější popis jazyka XML včetně pravidel, kterými se řídí, je v [HER07].

Příklad XML dokumentu:

```
<?xml version="1.0" encoding="UTF-8"?>
<zamestnanec>
  <jmeno>Jan</jmeno>
  <prijmeni>Nový</prijmeni>
  <plat mena="CZK">25000</plat>
</zamestnanec>
```

5.2.1 Zpracování XML

Díky rozšířenosti XML existuje v Javě[3] množství nástrojů, které s ním usnadní práci. Těmto nástrojům se říká *parsery* a jejich úkolem je rozdělení dokumentu na jednotlivé části. Kromě toho provedou rovněž kontrolu správné strukturovanosti XML (viz [HER07]). Na následujících řádcích je základní rozdělení a příklady konkrétních parserů.

Proudové čtení

Parser čte postupně XML dokument a pro každou ucelenou část vyvolá událost, na kterou lze programově reagovat. Výhodou je především vysoká rychlost načítání a malá paměťová náročnost, nevýhodou je nízkoúrovňové zpracování a nemožnost se v XML dokumentu vracet [HER07].

Příklady:

- SAX – pro čtení,
- StAX – na rozdíl od parseru SAX umožňuje i zápis.

Práce se stromovou reprezentací dokumentu

Princip spočívá načtení celého XML dokumentu do stromové struktury v paměti. Výhodou je možnost přistupovat k libovolným údajům, modifikovat je a následně zapsat zpět do XML dokumentu. Nevýhodou je vyšší paměťová náročnost [HER07].

Příklady:

- DOM
- JAXB

[3] A nejen v Javě, ale v každém moderním programovacím jazyce.

6 ANALÝZA

Následující části se již věnují vyvíjenému systému, který se skládá ze dvou aplikací, mezi nimiž se budou data přenášet prostřednictvím XML. Schéma systému je na obrázku 6.1.

Obr. 6.1: Schéma systému

6.1 Popis křižovatky

Vzhledem k malému displeji mobilních telefonů je potřeba stanovit omezení týkající se velikosti křižovatky. Aby byla zachována dostatečná velikost tlačítek pro snadné zadávání průjezdů, byl omezen maximální počet ramen křižovatky na 4 a maximální počet vjezdových pruhů v jednom rameni na 3.

Dále je nezbytné zavést označení, jakým budou popisovány parametry křižovatky potřebné pro správné vykreslení v mobilní aplikaci.

6.1.1 Atributy křižovatky

Název

U každé křižovatky je vhodné znát její název (vlastní pojmenování). Není to sice nezbytný údaj, ale usnadní orientaci v datech.

Typ dat: řetězec.

Počet ramen

Jak bylo již výše napsáno, v našem případě může být křižovatka tří nebo čtyřramenná.

Typ dat: číselná hodnota udávající počet ramen (3 nebo 4).

Časy měření

Aby bylo možné ze statistik vyčíst, jak se mění pravděpodobnosti odbočování v závislosti na denní době (případně kolik projede aut za časový interval), je vhodné mít k dispozici údaje o čase začátku a konce měření.

Typ dat: řetězec s datumem.

Dalšími atributy křižovatky jsou její ramena.

6.1.2 Atributy ramen

Název ulice

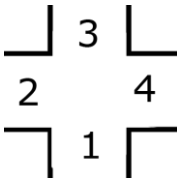
Údaj, který umožní identifikaci ramen.

Typ dat: řetězec.

Označení ramena

Aby bylo možné určit, ke kterému rameni se vztahují konkrétnější údaje (například o pruzích), je nutné si zavést jejich označení. Nejjednodušší možností je číslování ramen. Jako počátek je zvoleno spodní rameno (z hlediska světových stran se jedná o jih).

Typ dat: číslo 1 až 4 (viz obr. 6.2).



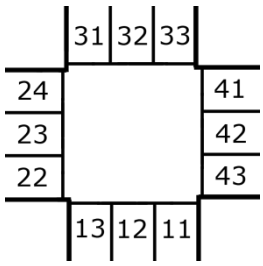
Obr. 6.2: Označení ramen

Ramena se skládají z jednotlivých pruhů.

6.1.3 Atributy vjezdových pruhů

Označení pruhu

Stejně jako tomu je v případě ramen, i každý pruh musí mít unikátní identifikátor. Aby bylo na první pohled patrné, v jakém je pruh rameni a na jaké pozici, skládá se označení ze dvou částí (čísel). Z čísla ramena, ve kterém se pruh nachází a čísla pruhu, přičemž číslování pruhu je v rámci ramena od 1 do 3 zprava doleva. Například prostřední pruh ve třetím rameni má v tomto případě označení „32“.



Obr. 6.3: Označení pruhů

Typ dat: dvouciferné číslo (viz obr. 6.3).

Směrové šipky

U každého vjezdového pruhu musí být zřejmé, do jakých směrů mohou projíždějící auta zabočit. To určují směrové šipky. V tomto případě se pro popis jako nejvhodnější jeví použití znaků, které charakterizují zmíněné směry.

Typ dat: Řetězec se znakem (znaky) představující směr:

- R – rovně,
- P – vpravo,
- L – vlevo.

Pokud jeden pruh umožňuje více směrů jízdy, zapíší se znaky za sebe do jednoho řetězce (např. RP pro směrovou šipku „rovně a vpravo“).

6.1.4 Atributy výjezdových pruhů

Označení pruhu

U výjezdových pruhů je situace jednodušší než u vjezdových, neboť nerozlišujeme mezi pruhy v rámci jednoho ramena. Je proto možné použít pro výjezdové pruhy stejné označení jako pro ramena (viz obr. 6.2).

6.2 XML soubor s popisem křižovatky

Jak bylo rozhodnuto v kapitole 5.1, data se budou mezi aplikacemi přenášet prostřednictvím XML souborů. Nejprve je proto nutné vytvořit vhodnou strukturu XML, která bude obsahovat všechny důležité informace popisující křižovatku.

Struktura XML souboru je stejná jak pro přenos dat z desktopové aplikace na mobilní, tak pro přenos z mobilní aplikace na desktopovou, jinak řečeno jedná se o stejný soubor. Ten bude vygenerován desktopovou aplikací na základě uživatelova návrhu, následně přenesen do mobilní aplikace – zde do něj budou zaznamenány počty průjezdů, a nakonec bude přenesen zpět do desktopové aplikace, jež ho zpracuje a vypíše statistiky.

6.2.1 Struktura

Kořenový element `krizovatka` nese prostřednictvím atributů informaci o počtu ramen a názvu křižovatky:

```
<krizovatka pocetRamen="4" nazev="Název křižovatky">
…
</krizovatka>
```

V prvním vnořeném elementu `cas` je obsažena informace o časech měření (na ukázce je zároveň vidět, v jakém formátu jsou časové údaje):

```
<cas casZacatku="2012-04-13 11:20" casKonce="2012-04-13 11:30" />
```

Následuje *n* elementů (kde *n* je počet ramen) `rameno` s údaji o jednotlivých ramenech. V atributech je uvedeno číslo ramena a jméno ulice:

```
<rameno jmenoUlice="tř. 1.máje - jih" cislo="1">
…
</rameno>
```

Tyto elementy obsahují zpravidla několik elementů `pruhVjezd` (pro vjezdové pruhy) a jeden element `pruhVyjezd` (pro výjezdové pruhy).

Vjezdové pruhy mají jako atributy identifikátor pruhu (označení) a směrové šipky (viz kapitola 6.1.3). Pro každý směr (daný směrovými šipkami), který daný pruh nabízí, je vnořen element `pocetPrujezdu`:

```
<pruhVjezd oznaceni="21" sipky="RP">
      <pocetPrujezdu smer="R">0</pocetPrujezdu>
      <pocetPrujezdu smer="P">0</pocetPrujezdu>
</pruhVjezd>
```

Naproti tomu u výjezdových pruhů se nemá smysl zabývat směrem. Ale vzhledem k tomu, že výjezdové pruhy uvažujeme v rámci ramena dohromady, je nutné nést informaci o jejich počtu pomocí atributu:

```
<pruhVyjezd oznaceni="4" pocet="1">
      <pocetPrujezdu>0</pocetPrujezdu>
</pruhVyjezd>
```

6.3 Mobilní aplikace

Mobilní aplikace bude sloužit pro manuální zaznamenávání průjezdů vozidel v „terénu“, případně z videozáznamu.

6.3.1 Specifikace požadavků

- Vykreslit křižovatku pospanou v souboru.
- Možnost snadného zadávání vjezdového a výjezdového pruhu.
- Zápis naměřených dat do souboru.

Po spuštění aplikace by uživatel měl mít možnost vybrat vstupní soubor. Na jeho základě se vykreslí křižovatka a rozmístí ovládací prvky představující jednotlivé pruhy.
V případě, že uživatel zadá průjezd vjezdovým pruhem, u kterého je jen jedna možnost směru, automaticky se dopočte výjezdový pruh. Pakliže existuje více možností, je nutné následně zadat výjezdový pruh (výjezdové rameno), kterým vozidlo křižovatku opustilo.

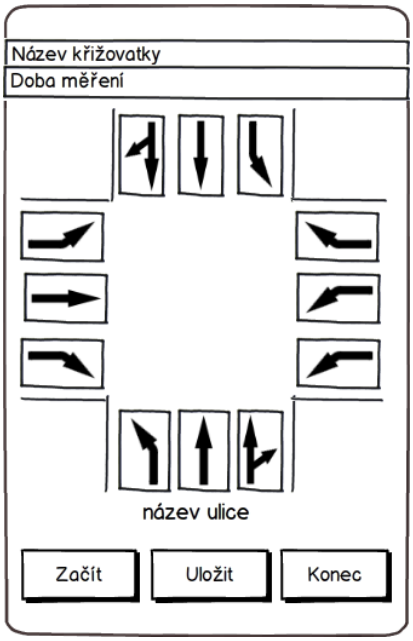
6.3.2 Vykreslení křižovatky

Vykreslování křižovatky je možné realizovat dvěma způsoby – vhodným rozmístěním tlačítek se směrovými šipkami nebo využitím grafické knihovny a nakreslení celé

křižovatky na `Canvas`[4]. V druhém případě by bylo kromě samotného vykreslení křižovatky navíc nutné naprogramovat zpracování události `onTouch()`, kde by se na základě souřadnic dotyku vypočítalo, na který pruh bylo „klepnuto“. Tato varianta je ovšem náročnější na implementaci a výsledek by byl téměř stejný, proto byla zvolena varianta s tlačítky.

6.3.3 Návrh GUI

Návrh grafického uživatelského rozhraní je vidět na obrázku 6.4.



Obr. 6.4: Náčrt GUI mobilní aplikace

6.4 Desktopová aplikace

Desktopová aplikace bude část systému, jejímž úkolem je navržení vlastní křižovatky a zobrazení statistik. Bude napsána v programovacím jazyce *Java*, čímž bude docíleno plné přenositelnosti na libovolnou platformu [HER00].

6.4.1 Specifikace požadavků

- Aplikace bude umožňovat navržení klasické (tří nebo čtyřramenné) křižovatky se zadáním všech podstatných parametrů.
- Při návrhu bude vykreslovat aktuální podobu křižovatky.
- Zápis navržené křižovatky do souboru (který bude zpracovatelný mobilní aplikací).

[4] Plátno na kreslení.

- Aplikace bude dále umět zpracovat více vstupních souborů s naměřenými daty a přehledně vypsat souhrnné statistiky (pravděpodobnosti odbočení, vytížení pruhů).
- Vypočtené statistiky bude možno exportovat do souboru.

6.4.2 Zobrazované statistiky

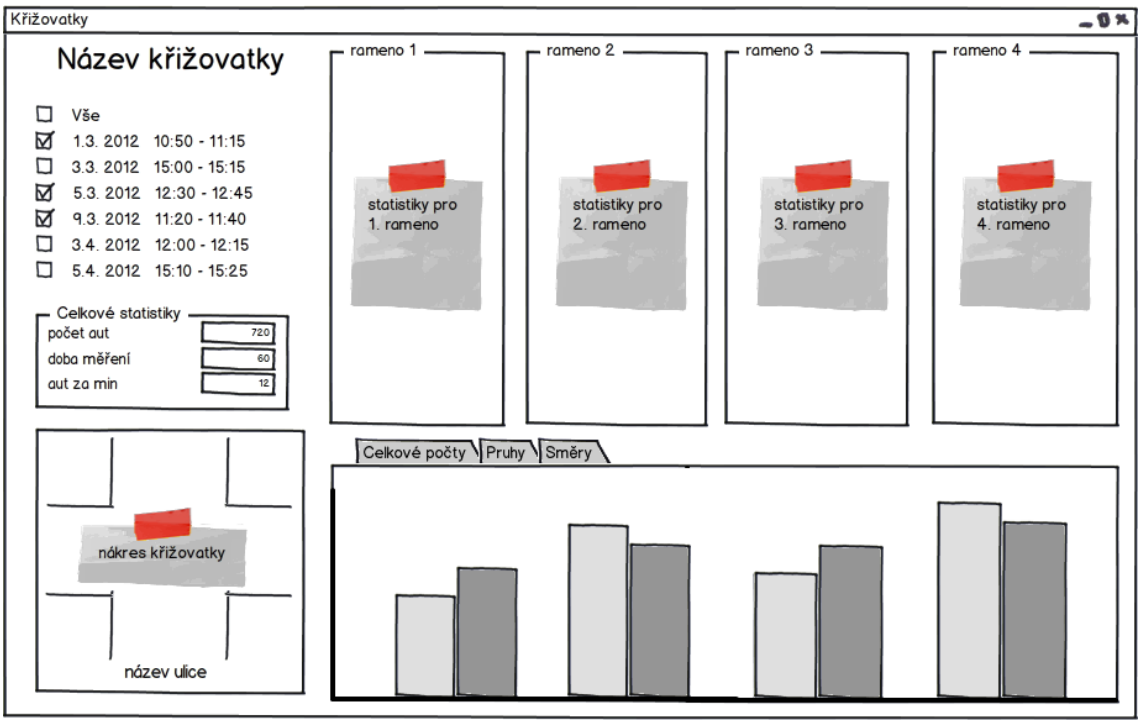
Aplikace bude vypisovat pro každé rameno zvlášť následující statistiky o počtu projetých vozidel:

- **Celkové počty** – celkový počet aut, která projela všemi vjezdovými pruhy a celkový počet aut, která projela výjezdovými pruhy.
- **Směry** – počty aut pro jednotlivé směry, bez ohledu na pruhy.
- **Pruhy** – počty projíždějících aut jednotlivými pruhy.
- **Pruh *n*** – u každého vjezdového pruhu zvlášť zobrazí počty aut pro jednotlivé směry v rámci daného pruhu.

K počtům budou vždy uvedena procenta, aby byly zřejmé pravděpodobnosti odbočení jak v rámci celého ramena, tak v rámci jednotlivých pruhů.

6.4.3 Návrh GUI

Vzhledem k požadavkům je nezbytné, aby aplikace měla grafické uživatelské rozhraní. Návrh části se statistikami je na obrázku 6.5.



Obr. 6.5: Náčrt GUI desktopové aplikace

7 IMPLEMENTACE

7.1 Mobilní aplikace

V této kapitole je popsána implementace mobilní aplikace. Jak již bylo rozhodnuto v kapitole 3.4, mobilní aplikace je vyvinuta pro platformu Android. Základní charakteristiky této platformy jsou popsány v kapitole 4.

7.1.1 GUI

Orientace

Mobilní telefony s Androidem jsou v drtivé většině případů vybaveny akcelerometrem, což je zařízení, které detekuje orientaci telefonu. Na základě toho mění módy zobrazení (na šířku **x** na výšku). V našem případě by přepínání orientace nebylo vhodné, neboť by při zobrazení na šířku vznikly problémy s umístěním textových polí a zbylo by méně místa na zobrazení křižovatky (viz návrh na obr. 6.4). Navíc by měnící se orientace uživatele mátla. Zákaz přepínání mezi módy orientace je proveden explicitním zadáním zvoleného módu jako atributu `android:screenOrientation="portrait"` elementu `activity` v souboru `AndroidManifest.xml`.

Obrázky

Veškeré obrázky používané aplikací se nachází ve složce `res/`. Zde jsou tři další složky – `drawable-hdpi`, `drawable-mdpi`, `drawable-ldpi`, pro telefony s různými hustotami bodů displeje. Obrázky se natahují ze složky odpovídající displeji daného telefonu, avšak pokud se v ní nenajdou, prohledají se zbývající složky. To znamená, že není bezpodmínečně nutné vytvářet obrázky v různých rozlišeních pro různé typy displeje. V našem případě je využita složka `drawable-hdpi`, kde jsou kromě ikon a obrázků křižovatek především směrové šipky pro pruhy. Pro snadnou identifikaci v programu jsou pojmenovány podle jednotného tvaru:

`sipka_<číslo ramena><směry>`,

takže například šipka pro třetí (severní) rameno se směry rovně a vpravo je pojmenována:

`sipka_3rp`.

Všechny obrázky jsou ve formátu PNG, který umožňuje nastavit průhlednost, což znamená, že okolí šipek nebude například bílé, ale bude mít barvu podkladu.

Řetězce

Řetězce, se kterými aplikace pracuje, jsou umístěny v souboru `strings.xml` ve složce `res/values`. Jsou zde jak chybová hlášení, tak popisky na tlačítka nebo název celé aplikace. Například popisek tlačítka „Začít“ je v XML souboru zapsán následovně:

```
<string name="zacit">Začít</string> .
```

Pro přístup k řetězci z XML souboru s layoutem se používá zápis `@string/zacit`. K jiným řetězcům (např. chybová hlášení) je potřeba přistupovat z programového kódu (z `java` souboru). To je umožněno pomocí generované třídy v souboru `R.java`[5]:

```
getString(R.string.nepodariloSeNacistData) .
```

Menu

Položky menu jsou definovány v souboru `menu.xml` ve složce `res/menu`. Po stisku tlačítka „Menu“ na mobilním zařízení je volána metoda `onCreateOptionsMenu()`, která sestaví menu na základě položek v souboru `menu.xml`. Po výběru položky je zavolána metoda `onOptionsItemSelected()`. Z té se na základě vybrané položky volají další metody pro obsloužení dané události.

Layout

Samotný layout s ovládacími prvky (widgety), jejich rozmístěním a vlastnostmi je umístěn v souboru `main.xml` ve složce `res/layout`.

Jak bylo rozhodnuto v kapitole 6.3.2, pruhy křižovatky jsou reprezentovány tlačítky `ImageButton`, která mají jako obrázek nastaveny směrové šipky. Tlačítka jsou vhodně rozmístěna na `RelativeLayout`. Každé tlačítko má identifikátor ve tvaru:

```
IBPruh<označení pruhu>.
```

Aby bylo možné reagovat na stisk tlačítka v programovém kódu, má každé tlačítko nastaven atribut `onClick`. Hodnotou tohoto atributu je řetězec představující jméno metody, která bude po stisku tlačítka volána v programovém kódu.

Příklad tlačítka reprezentující pruh s označením „32“ a směrem „rovně“:

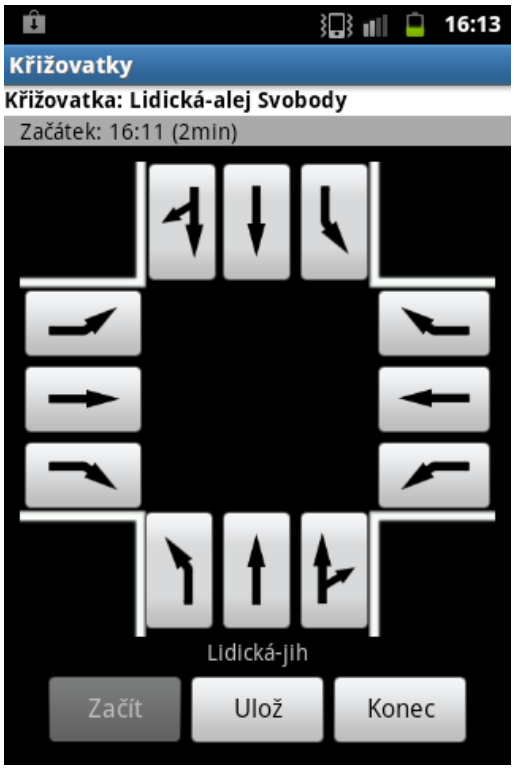
```
<ImageButton
      android:id="@+id/IBpruh32"
      android:onClick="StisknutoPruh32"
      android:src="@drawable/sipka_3r"
```

[5] Automaticky generovaná třída R, pomocí níž se přes identifikátory přistupuje ke zdrojům (řetězce, obrázky, ovládací prvky layoutu) z programového kódu.

```
        ... „atributy pro pozicování“
    </ImageButton>
```

GUI dále obsahuje tři komponenty `TextView`. V horní části je textové pole pro zobrazení informace o „názvu“ křižovatky, hned pod ním je další pole, které informuje o čase začátku měření a o tom, jak dlouho již měření probíhá. Poslední textové pole informuje o jménu ulice spodního (jižního) ramena.

V spodní části jsou tři tlačítka pro ovládání aplikace – Začít, Ulož, Konec. Vzhled aplikace je vidět na obrázku 7.1.



Obr. 7.1: Vzhled mobilní aplikace

7.1.2 Programový kód

Programový kód aplikace se skládá ze 4 zdrojových souborů s příponou `.java`. Ty jsou umístěny v balíku `cz.kozisek.krizovatky` ve složce `src`.

Datové třídy

Tři zdrojové soubory – `VjezdovyPruh.java`, `VyjezdovyPruh.java`, `Smer.java`, představují třídy, jejichž hlavním úkolem je uchovávat naměřené hodnoty počtu průjezdů.

`VjezdovyPruh` – Kromě označení pruhu a označení směrových šipek si s sebou nese i proměnnou s odpovídajícími výjezdovými pruhy. To jsou pruhy, kterými může vozidlo, projíždějící tímto vjezdovým pruhem, při daných směrech křižovatku opustit. Posledním atributem je seznam směrů (instancí třídy `Smer`). Každý směr, kterým je možné jet, má svoji instanci.

`VyjezdovyPruh` – Na rozdíl od vjezdových pruhů má pouze jednu proměnou – `pocetPrujezdu`.

`Smer` – Využívána třídou `VjezdovyPruh`. Zde jsou uloženy počty průjezdů pro jeden směr v rámci pruhu.

Hlavní aktivita

Jak bylo popsáno v kapitole 4.4.1, základem každé aplikace pro Android je aktivita, v našem případě se jedná o třídu `KrizovatkyActivity`.

Ta má, jako každá aktivita, deklarovanou metodu `onCreate()`, která je volána při spuštění aplikace. V této metodě se především nastaví grafické uživatelské rozhraní, což se provede připojením souboru s layoutem (k tomu se opět použije třída `R`):

```
setContentView(R.layout.main).
```

Důležité proměnné

`VjezdovyPruh[] vjezdPruhy` – pole s vjezdovými pruhy. Jednotlivé vjezdové pruhy jsou do pole namapovány podle svého značení na indexy od 0 do 11.

`VyjezdovyPruh[] vyjezdPruhy` – pole s výjezdovými pruhy. Ty jsou v poli na indexech odpovídajících jejich označení (sníženém o jedna, neboť indexy v poli začínají od 0, zatímco označení pruhu od 1).

Zpracování XML

Po stisku tlačítka „Začít“ se zobrazí dialog se všemi soubory ze složky[6] aplikace. Po výběru se zavolá metoda `nacteniDat()`, ve které se provádí zpracování vstupního XML souboru. K tomu je použit parser DOM, neboť je v našem případě vhodné mít celý soubor s daty uložen v paměti ve stromové struktuře. To umožní modifikaci dat (započtených průjezdů) a následné uložení změněného XML souboru na externí úložiště v mobilním telefonu.

Nastavení widgetů

Po zpracování XML souboru se volá metoda `inicializace()`. V té se nastavují widgety na základě vstupních dat. To obnáší především „zneviditelnění“ tlačítek pro

[6] Aplikace na OS Android mají vlastní externí úložiště ve složce: /Android/data/<jméno_balíku>/files/

pruhy, které v křižovatce nejsou (má-li křižovatka v 1. rameni dva pruhy, dvěma tlačítkům se nastaví viditelnost na „true“, poslednímu na „false“) a dále je nutné nastavit na tlačítka obrázky se směrovými šipkami. Vzhledem k tomu, že veškeré widgety jsou nadefinované v XML souborech s layoutem, je nutné pro přístup k nim využít opět třídu `R` a metodu `findViewById()`. Z `R` třídy se získá *id* a metoda `findViewById()` vrátí odpovídající widget. Tomu již lze měnit vlastnosti klasickým způsobem. Například „zneviditelnění“ prostředního pruhu z 3. ramena vypadá následovně:

```
ImageButton IBpruh = (ImageButton) findViewById(R.id.IBpruh32)
IBpruh.setVisibility(NEVIDITELNE);
```

U nastavení obrázků směrových šipek na tlačítka je situace složitější, neboť název obrázku je vytvářen dynamicky spojením údajů o rameni a směrových šipkách pro konkrétní pruh (např. „sipka_3rp“ – směrové šipky rovně a vpravo pro pruh z 3. ramene). K tomu je použita metoda `getResources().getIdentifier()`, která na základě názvu vrátí *id* obrázku, jenž se konstrukcí `IBpruh.setImageResource(ID)` nastaví na tlačítko.

Zaznamenávání průjezdů

Po stisku každého tlačítka reprezentujícího pruh se volá jemu odpovídající metoda `stisknutoPruh<oznaceni>`. Například pro prostřední pruh z 3. ramena vypadá metoda následovně:

```
public void stisknutoPruh32(View v) {
      zapocteniVjezdu(PRUH32);
      urceniVyjezdovehoPruhu(PRUH32);
}
```

Z ukázky je vidět, že jsou volány další dvě metody. První metoda započte průjezd vjezdovému pruhu, avšak pouze v případě, kdy je možný jen jeden směr. V druhé metodě se zjišťuje výjezdový pruh. Pokud je pouze jeden, započte se automaticky průjezd i odpovídajícímu výjezdovému pruhu a tím výpočet skončí.

Složitější situace nastane, když je možných směrů a tím pádem i výjezdových pruhů více. V takovém případě je nutné zadat, kterým pruhem vozidlo křižovatku opustilo. To je umožněno tím, že se po zadání vjezdového pruhu nabídnou tlačítka pro zadávání výjezdových pruhů. Po zadání výjezdového pruhu se započte průjezd jak pro vjezdový pruh (v zadaném směru), tak pro výjezdový pruh. Místo tlačítek pro výjezdové pruhy se opět nabídnout tlačítka pro zadávání vjezdových pruhů a tím je možné zadat další průjezd.

Časovač

Časovač má v aplikaci dvě funkce:

- informovat uživatele o délce doby měření,
- ukládat průběžně data do souboru.

Časovač je implementován pomocí vlákna a handleru, který zajišťuje komunikaci s hlavním vláknem aplikace. Vlákno časovače každých 60 sekund pošle zprávu handleru a ten následně volá metody hlavního vlákna pro uložení dat a nastavení času měření.

7.2 Desktopová aplikace

V této kapitole je popsána implementace desktopové aplikace. Připomeňme, že její hlavní funkce jsou návrh křižovatky a zobrazení statistik.

7.2.1 Datová reprezentace křižovatky

V kapitole 6.1 byly vyjmenovány všechny nezbytné atributy popisující křižovatku. Pro uchování výše uvedených informací slouží třídy `Krizovatka`, `Rameno` a `Pruh`.

Při návrhu křižovatky jsou v instancích těchto tříd uloženy již zadané parametry, na jejichž základě je vygenerován XML soubor.

Při zobrazení statistik jsou nejprve data z XML souboru načtena do zmíněných tříd, které se následně využívají při výpočtu statistik.

Krizovatka

Kromě atributů pro název křižovatky, počet ramen a čas měření, obsahuje ještě seznam ramen. Pro implementaci seznamu je použita kolekce – `ArrayList`. Celá třída je zděděna od `Observable` – metodami `setChanged()` a `notifyObservers()` upozorňuje zaregistrované posluchače o změně klíčových atributů. To se využívá při vykreslení křižovatky (viz kapitola 7.2.6).

Rameno

Atributy jsou jméno ulice, číslo ramena, počty vjezdových a výjezdových pruhů a seznam pruhů (opět implementován jako `ArrayList`).

Pruh

Obsahuje především informace o označení, směrových šipkách pruhu a počtech průjezdů vozidel. Počty průjezdů jsou implementovány jako kolekce `HashMap`, kde klíčem je směr a hodnotou počet průjezdů. To umožňuje jednoduché vkládání a vybírání hodnot podle zadaného směru s využitím metod `put` a `get` (ukázka na následujících řádcích).

Vložení počtu průjezdů `pocet` ve směru `smer`:

```
poctyPrujezdu.put(smer, pocet);
```

Výběr počtu průjezdů ve směru `smer`:

```
poctyPrujezdu.get(smer);
```

7.2.2 Generování XML souborů

Po dokončení návrhu křižovatky se exportují data z datových tříd do XML souboru. O to se stará třída `GenerovaniXML` (z balíku `bp.krizovatka`), které se předá název souboru a instance třídy `Krizovatka`.

Pro samotné vytvoření XML struktury je použita technologie StAX. Ta umožňuje jednoduchý zápis dat do výstupního proudu ve formátu XML.

Využívá se především metod `writeStartElement()`, `writeEndElement()`, `writeAttribute()` a `writeCharacters()` třídy `XMLStreamWriter`.

Například element `cas`:

```
<cas casZacatku="0" casKonce="0"/>,
```

je zapsán takto:

```
w.writeStartElement("cas");
w.writeAttribute("casZacatku", "" + 0);
w.writeAttribute("casKonce", "" + 0);
w.writeEndElement();
```

Zapisuje se do proudu bajtů. Pro vlastní zápis do souboru se využije třída `Transformer` – to zajistí správné odřádkování a odsazení[7].

7.2.3 Načítání XML souborů

Před zobrazením statistik je nutné načíst a zpracovat XML soubor s naměřenými daty. K tomuto účelu je použit parser DOM a třída `NacteniDat` (balík `bp.krizovatka`) Hlavním důvodem použití parseru DOM je možnost využít některých konstrukcí z mobilní aplikace, jenž tuto technologii pro načítání souborů používá.

Vlastní parser představuje instance třídy `DocumentBuilder`. Ta nabízí metodu `parse`, která vytvoří stromovou reprezentaci (`document`) XML souboru v paměti. Konkrétní uzly stromu se získají voláním metody `getElementsByTagName()` – vrátí seznam uzlů podle zadaného jména elementu. Následně se voláním dalších metod získají atributy a hodnoty elementů.

[7] Při přímém zápisu do souboru by bylo vše na jedné řádce, což by sice nevadilo aplikacím (které soubor načítají), ale pro člověka by to bylo nečitelné.

7.2.4 Výpočet statistik

Po načtení souboru (souborů) se provede výpočet statistik. Ten má na starosti třída `StatistikyKrizovatka` (balík `bp.statistiky`). Vzhledem k tomu, že je možné načíst více souborů[8] najednou a následně si za běhu přepínat mezi soubory, pro které mají být statistiky zobrazené, probíhá přepočet statistik i při každé změně výběru.

Objektu této třídy je při vytvoření předáno, kromě pole s načtenými křižovatkami, i odpovídající pole s „boolean“ hodnotami nastavenými podle toho, jaké křižovatky jsou vybrány[9]. Následně se volají metody pro výpočet všech statistik popsaných v kapitole 6.4.2 – v cyklu přes všechny vybrané křižovatky se sumarizují hodnoty a vypočítají procenta v daném kontextu (např. pro jeden směr ve srovnání s ostatními směry v konkrétním ramenu).

7.2.5 GUI

Grafické uživatelské rozhraní zajišťuje interakci s uživatelem. Skládá se ze dvou oken:

- okno pro zobrazení statistik,
- okno pro návrh křižovatky.

GUIStatistiky

Po spuštění celé aplikace se vytvoří právě instance této třídy odděděné od `JFrame`, což způsobí vykreslení hlavního okna. To je rozděleno na tři části, čehož je docíleno použitím sofistikovaného layout manažeru `GridBagLayout`. Se sofistikovaností souvisí také relativní komplikovanost tohoto manažeru. To je dáno především tím, že se před vkládáním musí každá komponenta předpřipravit – musí se vytvořit objekt `GridBagConstraints` a nastavit jeho proměnné (podrobněji viz [HER09]). Aby nebylo nutné psát několikařádkový kód před vkládáním každé komponenty, stará se o nastavení metoda `nastavGBL()`, kam se předají všechny potřebné parametry. Jednotlivé části okna jsou popsány na následujících řádcích. Výsledný vzhled okna pro zobrazení statistik je na obrázku 7.2.

Levý panel

V horní části obsahuje komponenty `JCheckBox`. Ty jsou dynamicky vytvořeny na základě vybraných souborů, přičemž každý soubor je reprezentován jedním „checkboxem“. Při změně vybraných souborů dojde k překreslení zbylých dvou částí se statistikami a grafy. V dolní části panelu je nákres křižovatky.

[8] Ale jen soubory pro stejnou křižovatku.

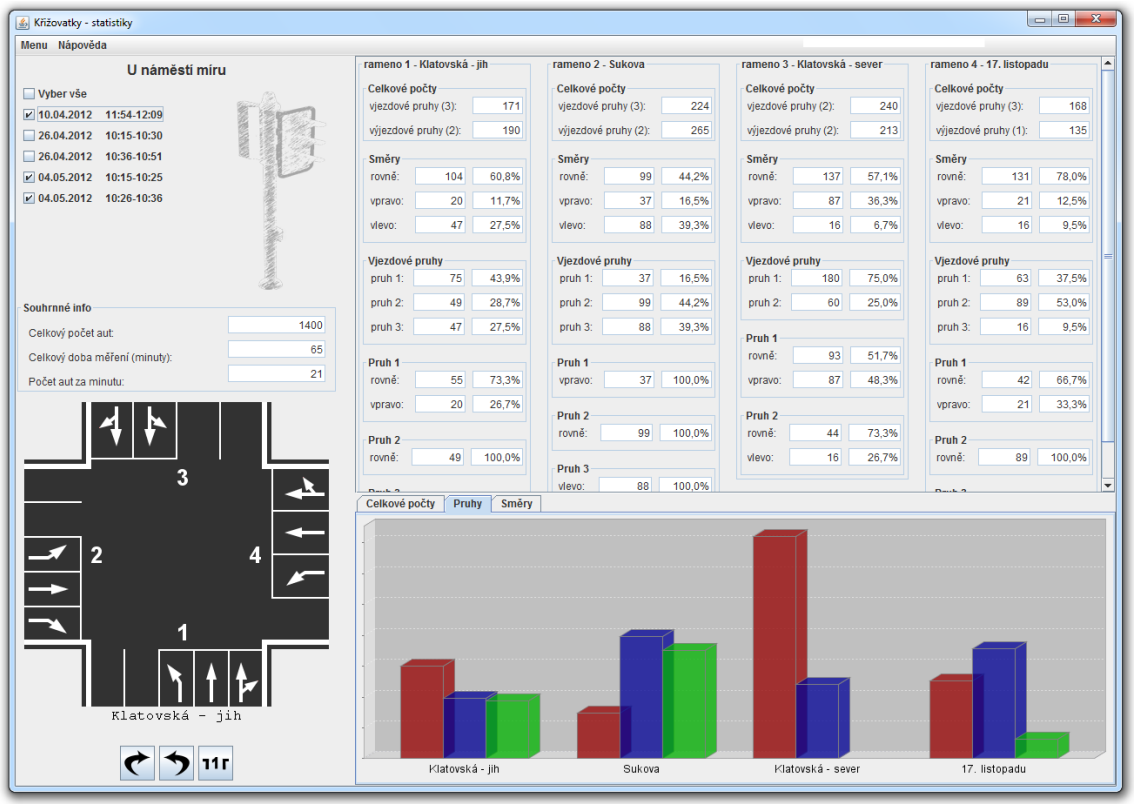
[9] Výběr křižovatek se provádí pomocí zaškrtávacích políček v GUI.

Panel se statistikami

V této části jsou textově zobrazeny statistiky pro jednotlivá ramena. Pro tento účel jsou vytvořeny nové komponenty[10] `JLabelPlain` a `MujTextField` (umístěné v balíku `bp.komponenty`). Ty v konstruktoru mění určité parametry, aby nebylo například nutné nastavovat pro každé textové pole zvlášť editovatelnost na „false“. Rozmístění zajišťuje opět `GridBagLayout` a metoda `nastavGBL()`. Komponenty představující jednotlivé pruhy mají přidán `MouseListener` s implementovanými metodami `mouseEntered()` a `mouseExited()`. Ty volají metodu nákresu křižovatky, která zajistí zvýraznění pruhu, na jehož popisek ve statistikách se najelo myší.

Panel s grafy

Jedná se o `JTabbedPane` se třemi záložkami, do kterých jsou umístěny panely – instance třídy `PanelGraf` (viz kapitola 7.2.7).



Obr. 7.2: Vzhled desktopové aplikace – zobrazení statistik

GUINavrhKrizovatky

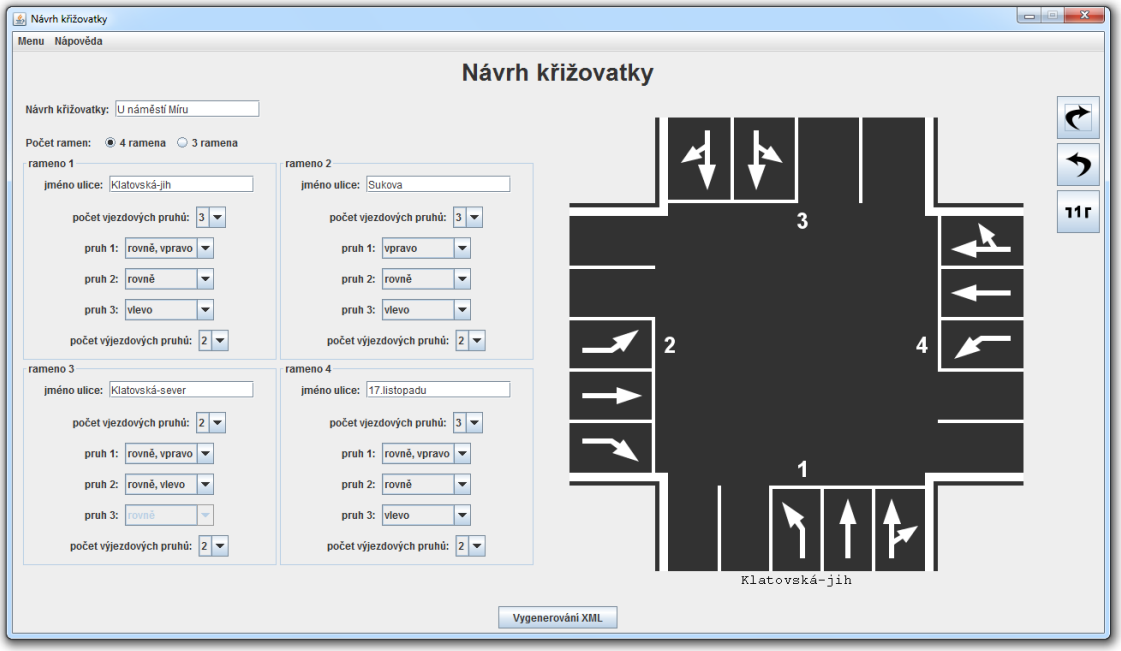
Třída reprezentuje okno pro návrh křižovatky – je odděděna od `JFrame`. V konstruktoru se vytvoří vnitřek okna složený především z komponent `JLabel`, `JTextField`,

[10] Oddděné od původních JLabel a JtextField.

`JComboBox` a `JRadioButton`. Dále jsou nastaveny obsluhy událostí pro jednotlivé ovládací prvky. Základní rozmístění komponent zajišťuje layout manažer `BorderLayout`, panely pro jednotlivá ramena jsou rozmístěny do mřížky s využitím `GridLayout`.

V levé části okna jsou ovládací prvky pro nastavení atributů křižovatky – ty jsou hned ukládány do objektu `Krizovatka`. Tento objekt má zaregistrovaného posluchače – `NakresKrizovatky`, což znamená, že nákres v pravé části okna reaguje překreslením na každou změnu klíčového atributu křižovatky.

Vzhled okna pro návrh křižovatky je na obrázku 7.3.



Obr. 7.3: Vzhled desktopové aplikace – zobrazení návrhu křižovatky

7.2.6 Grafika

Aby měl uživatel lepší představu o tom, jak vypadá navržená křižovatka, vykresluje se na základě již zadaných parametrů její nákres. Nákres křižovatky je pro usnadnění orientace i v okně se zobrazenými statistikami. Vykreslení obrázku křižovatky má na starosti třída `NakresKrizovatky`, která je potomkem třídy `Canvas` (plátno), přičemž samotné kreslení využívá možností třídy `Graphics2D`. Nejdůležitější metodou je `paint()` – ta zajišťuje vlastní kreslení. Je volána jak při prvotním vykreslení, tak při změně parametrů křižovatky, aby byl nákres aktuální. O změně parametrů se třída provádějící nákres dozví díky tomu, že implementuje rozhraní `Observer` a je zaregistrována jako posluchač objektu `Krizovatka`.

7.2.7 Vykreslování grafů

Pro vykreslování grafů je použita volně dostupná knihovna *JFreeChart*. Pomocí ní lze vykreslovat grafy na základě vytvořených „datasetů“[11].

Je použit 3D sloupcový graf s vertikální orientací a kategoriemi[12]. Do hlavního okna se statistikami se grafy přidají jakožto panely (instance třídy `PanelGraf` odděděné od `JPanel`) pro komponentu `JTabbedPane`.

Před vykreslením grafu se musí nejprve vytvořit dataset. Ten je typu `DefaultCategoryDataset`, přičemž nová data se přidávají voláním metody `addValue()` s třemi parametry – hodnota, sloupce, kategorie.

Samotný graf je typu `JFreeChart` a vytvoří se příkazem:

```
ChartFactory.createBarChart3D(<parametry>)
```

Parametry slouží k nastavení grafu (např. titulek, orientace) a předání „datasetu“.

[11] Sada vstupních dat pro graf.

[12] Jedna kategorie představuje jedno rameno.

8 TESTOVÁNÍ APLIKACÍ

Testování systému spočívalo v:

- navržení křižovatky desktopovou aplikací,
- přenesení vygenerovaného XML souboru na mobilní zařízení,
- zaznamenávání průjezdů vozidel křižovatkou v terénu,
- přenesení souboru s naměřenými daty zpět do desktopové aplikace,
- zobrazení statistik na základě naměřených dat,
- export statistik do XML souboru.

Měření probíhalo na dvou křižovatkách – v Plzni a Přešticích. Vlivem testování docházelo k drobným změnám (vylepšením) mobilní aplikace.

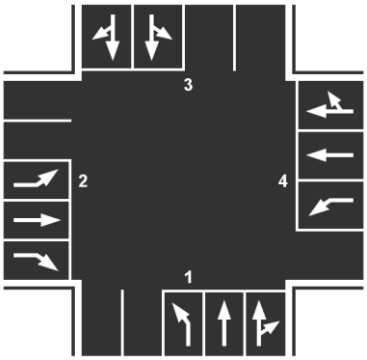
8.1 Křižovatka u náměstí Míru

Tato křižovatka se nachází v Plzni a má 4 ramena – tvoří ji ulice Klatovská, Sukova a 17. listopadu.

Na obrázku 8.1 je nákres křižovatky.

Popis ramen (názvy ulic):

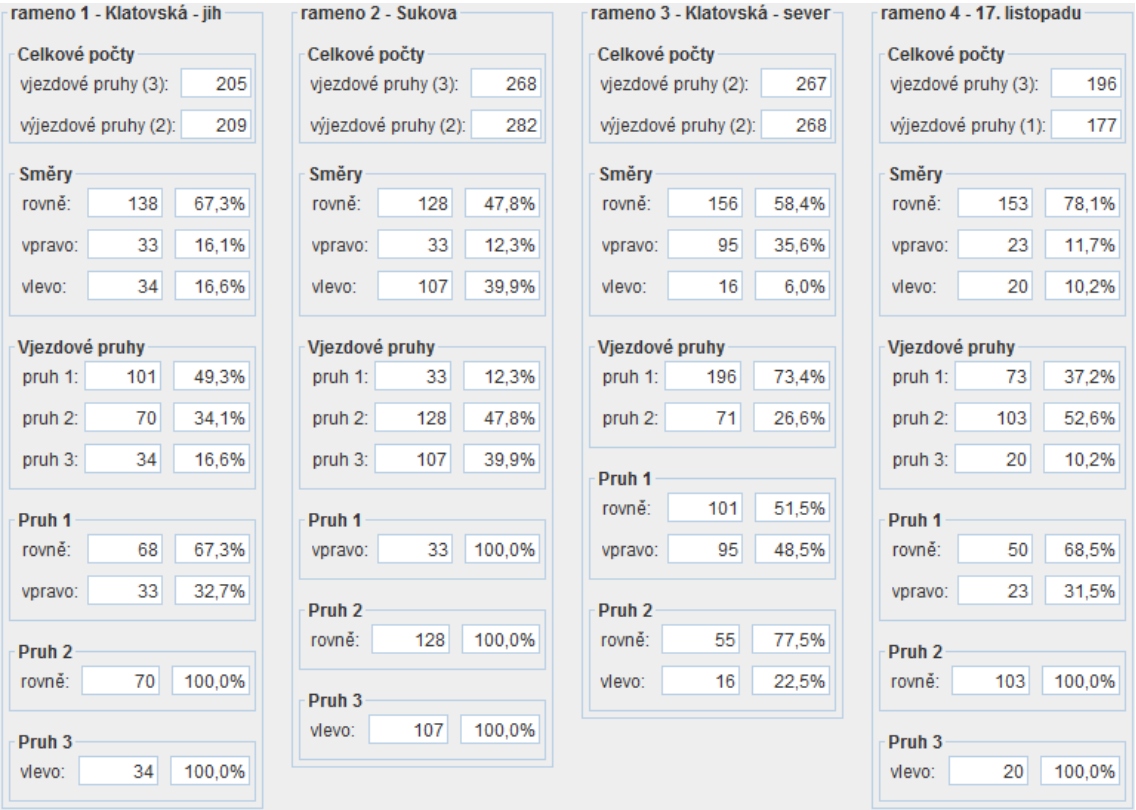
1 – Klatovská-jih
2 – Sukova
3 – Klatovská-sever
4 – 17.listopadu



Obr. 8.1: Nákres křižovatky

Zaznamenávání průjezdů na této křižovatce probíhalo ve dvou dnech vždy v čase mezi 10. a 11. hodinou dopoledne, přičemž interval měření byl 15 minut. Vzhledem k velikosti křižovatky se průjezdy zaznamenávaly nadvakrát (nejprve 2 ramena, následně zbylá 2 ramena).

Podrobné statistiky z obou měření, zpracované desktopovou aplikací, jsou na obrázku 8.2. Z naměřených dat lze například vyčíst, že vozidlo, přijíždějící ulicí Sukova, pojede s pravděpodobností 47,8% rovně, s pravděpodobností 39,9% zabočí doleva a s malou pravděpodobností 12,3% zabočí doprava. Zaměříme-li se na jednotlivé pruhy, můžeme například vyčíst, že vozidlo, jedoucí druhým pruhem ze severu ulicí Klatovská, bude s pravděpodobností 77,5% pokračovat rovně a s pravděpodobností 22,5% zabočí doleva do ulice 17. listopadu.



Obr. 8.2: Statistiky zobrazené desktopovou aplikací

Po prvním testu došlo k drobnému upravení mobilní aplikace. Byla změněna velikost a umístění tlačítek výjezdových pruhů, aby bylo zadávání snadnější a intuitivní bez přímého pohledu na displej.

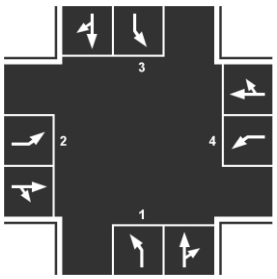
8.2 Křižovatka v Přešticích

Další testování probíhalo na křižovatce v Přešticích a oproti prvnímu měření byla testována i přesnost naměřených dat.

Na obrázku 8.3 je nákres křižovatky.

Popis ramen (názvy ulic):

1 – tř. 1.máje-východ
2 – Husova
3 – tř. 1.máje-západ
4 – Palackého



Obr. 8.3: Nákres křižovatky

Vzhledem k menší velikosti byly tentokrát zaznamenávány průjezdy pro celou křižovatku najednou. Zároveň bylo vše nahráváno na záznamové zařízení. Video záznam pak bylo možné pustit zpomaleně, případně zastavovat a provést tak přesné bezchybné měření.

Nakonec byl záznam použit (tentokrát bez zpomalení a zastavování) ke klasickému zaznamenávání s využitím tužky a papíru. Porovnání počtu zaznamenaných průjezdů je vidět v tabulce 8.4.

Z tabulky je vidět, že při měření v reálném čase vznikají drobné nepřesnosti. Ty se ovšem vyskytují i při použití klasického zaznamenávání „na papír“. Důvodem je příliš mnoho vozidel, které projíždějí křižovatkou najednou, a může se stát, že uživatel nestihne všechny zaznamenat, případně se může ve spěchu překlepnout. V tomto případě se naměřené pravděpodobnosti odbočení lišily od skutečnosti maximálně o 2%. Je ale nutno připomenout, že byly zaznamenávány průjezdy celou křižovatkou najednou. Přesnějšího měření by bylo docíleno rozdělením křižovatky a měřením menšího počtu ramen naráz, což platí obecně.

	naměřeno - mobil			skutečnost			naměřeno - papír,tužka		
	vlevo	rovně	vpravo	vlevo	rovně	vpravo	vlevo	rovně	vpravo
tř.1.máje-východ	8	74	12	9	71	14	9	72	14
Husova	9	12	5	10	12	5	9	11	5
tř.1.máje-západ	9	70	14	8	73	16	7	76	12
Palackého	14	10	6	14	12	6	13	11	7

Tab. 8.4: Porovnání naměřených dat se skutečností

Po tomto testu byla do mobilní aplikace implementována možnost zapnutí krátkého zavibrování při klepnutí na tlačítko představující pruh. Díky tomu má uživatel při intuitivním zadávání (bez neustálého pohledu na displej) zpětnou vazbu, která ho informuje o tom, zda došlo k zaznamenání průjezdu.

9 ZÁVĚR

Cílem práce bylo seznámit se se základy dopravních simulací, dále s vývojem aplikací pro mobilní zařízení a se současným stavem na trhu mobilních platforem a na základě získaných znalostí vyvinout systém umožňující uživateli získat údaje o pravděpodobnostech odbočení pro vybranou křižovatku.

Výsledkem jsou dvě aplikace (jedna pro mobil, druhá pro desktop), mezi kterými se přenášejí data prostřednictvím XML souborů. S ohledem na velikost displeje mobilního telefonu bylo nutné učinit omezení týkající se velikosti křižovatky, nicméně nutno podotknout, že i přesto systém postačuje pro drtivou většinou křižovatek v Plzni a s vysokou pravděpodobností i v ostatních městech

Samotná mobilní aplikace umožňuje, s ohledem na velikost displeje, snadné zadávání vjezdových a výjezdových pruhů, které nezanáší do naměřených dat větší chybu, než klasické „čárkování na papír“. Na rozdíl od záznamu na papír však umožňuje snazší přenos naměřených dat do počítače k dalšímu zpracování.

Desktopová aplikace uživateli umožňuje velmi rychlé navržení vybrané křižovatky a přehledné zobrazení statistik (a přepínání mezi nimi) včetně grafického výstupu. Oproti původnímu zadání navíc umožňuje export statistik i grafů do souboru. Výsledná práce tak zcela splňuje zadání.

Z hlediska přínosu autorovi jsou důležité především zkušenosti získané při práci s velkým množstvím rozličných technologií. Od mobilních platforem, přes programování pro Android, zpracování XML souborů různými technologiemi až po vytváření netriviálního, dynamicky se měnícího GUI.

Do budoucna by bylo možné upravit mobilní aplikaci pro tablety s Androidem. Ta by mohla využívat mnohem většího displeje, což by umožňovalo práci s většími křižovatky a hlavně by to přinášelo větší možnosti, jak zlepšit ovládání aplikace. Avšak v tuto chvíli (1. polovina roku 2012) u nás zatím nejsou, zejména z cenových důvodů, tablety příliš rozšířeny.

PŘEHLED ZKRATEK

ADT	Android Development Tools
API	Application Programming Interface
AVD	Android Virtual Device
DDMS	Dalvik Debug Monitor Server
DOM	Document Object Model
GUI	Graphical User Interface – grafické uživatelské rozhraní
J2ME	Java 2 Micro Edition
Java SE	Java Standard Edition
JAXB	Java Architecture for XML Binding
OS	Operační Systém
PNG	Portable Network Graphics
SAX	Simple API for XML
SDK	Software Development Kit
StAX	Streaming API for XML
XML	Extensible Markup Language

LITERATURA A ZDROJE

[HPN96] HAMILTON, J. A., POOCH, U. V., NASH, D. A.: ***Distributed Simulation***, CRC Press, New York, 1996.

[POT09] POTUŽÁK, T.: ***Methods for Reduction of Inter-Process Communication in Distributed Simulation of Road Traffic***, dizertační práce ZČU-KIV, 2009.

[FMT00] FUJIMOTO, R. M.: ***Parallel and Distributed Simulation Systems***, John Wiley & Sons, New York, 2000.

[HAR03] HARTMAN, D.: ***Modelování dopravní situace systémy hromadné obsluhy***, diplomová práce ZČU-KIV, 2003.

[MIK07] MIKKONEN, T.: ***Programming mobile devices: an introduction for practitioners***, Chichester: John Wiley and Sons, 2007.

[MUR10] MURPHY, M. L.: ***Android 2: Průvodce programováním mobilních aplikací***. 1. vydání, Brno: Computer Press, 2011.

[HER00] HEROUT, P.: ***Učebnice jazyka Java***. 3. rozšířené vydání, České Budějovice: Kopp, 2007.

[HER07] HEROUT, P.: ***Java a XML***. 1. vydání, České Budějovice: Kopp, 2007.

[HER09] HEROUT, P.: ***Java – grafické uživatelské prostředí a čeština***. dotisk 2. vydání, České Budějovice: Kopp, 2009.

[WEB1] Android Developers.
URL < http://developer.android.com > [cit. 12.dubna 2012]

[WEB2] Článek se základními informacemi o OS Android.
URL < http://www.zdrojak.cz/clanky/vyvoj-pro-android-i/ >
[cit. 12. dubna 2012]

[WEB3] Informace o projektu JUTS.
URL < http://www.juts.zcu.cz/ > [cit. 21. dubna 2012]

PŘÍLOHA A – UŽIVATELSKÁ PŘÍRUČKA K MOBILNÍ APLIKACI

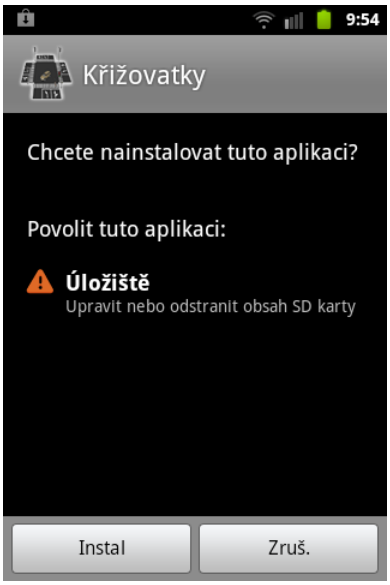
Minimální požadavky na rozlišení displeje: 480 x 320.

Instalace

1. Aplikace je distribuována jako balík **Krizovatky.apk**[13]. Tento balík si nakopírujeme do vhodné složky (např. „download“) na paměťové kartě mobilního telefonu (např. pomocí USB kabelu).

2. V dalším kroku potřebujeme nějaký souborový manažer. Většina telefonů má v sobě souborový manažer nainstalovaný, v opačném případě je možné ho stáhnout například z **Google Play**[14]. V něm se přesuneme do složky, kam jsme balík nakopírovali a spustíme jej. Objeví se instalační dialog. Klepnutím na „Instal“ aplikaci nainstalujeme (obr. A.1).

3. Nakonec aplikaci otevřeme, aby došlo k vytvoření složky na paměťové kartě, kde budou umístěny XML soubory, se kterými aplikace pracuje.



Obr. A.1: Instalace

[13] .apk – Android Package – instalační balík do kterého jsou kompilovány všechny aplikace určené pro Android.

[14] Oficiální market s aplikacemi pro Android.

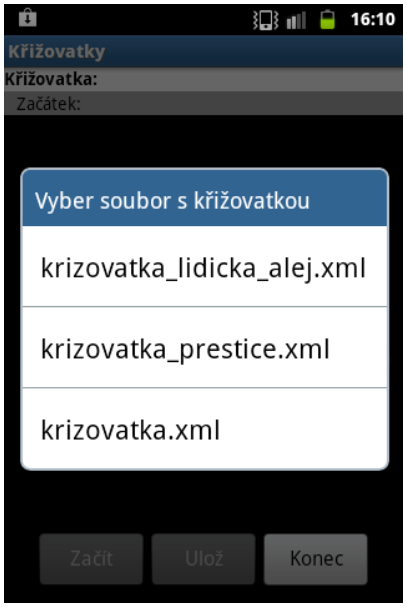
Výběr vstupního souboru

1. XML soubor s popisem křižovatky, který jsme vygenerovali pomocí desktopové aplikace, nakopírujeme na paměťovou kartu do složky:

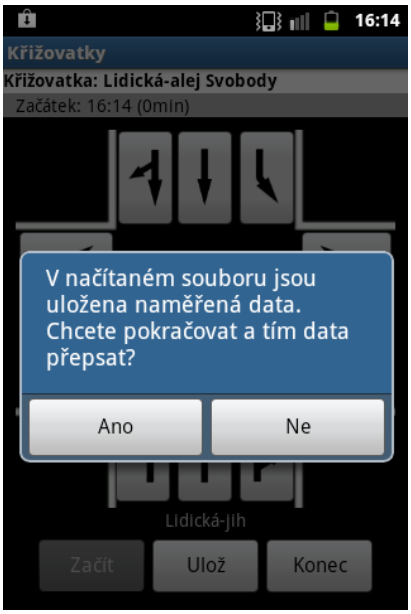
```
Android/data/cz.kozisek.krizovatky/files
```

2. Spustíme aplikaci a stiskneme tlačítko „Začít“. Objeví se dialog, v němž můžeme vybrat jeden ze souborů, které jsou umístěné ve výše uvedené složce (obr. A.2).

Pozn.: Obsahuje-li vstupní soubor již naměřená data, upozorní nás na to dialog (obr. A.3). V případě, že chceme pokračovat (klepnutí na „Ano“), dojde k přepsání starých dat v souboru!



Obr. A.2: Výběr souboru



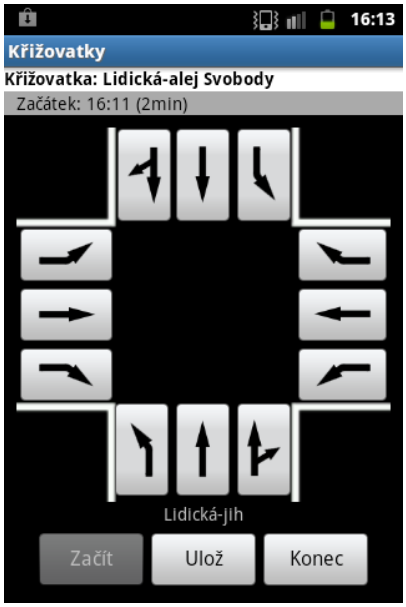
Obr. A.3: Přepsání souboru

Měření

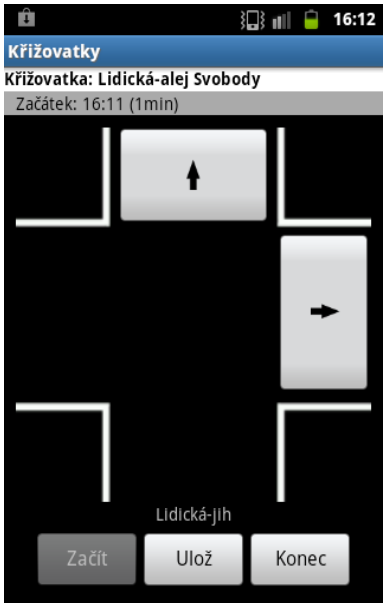
Po výběru vstupního souboru se vykreslí křižovatka a je možné zadávat průjezdy vozidel.

Zadávání průjezdů je intuitivní – pokud pruh, kterým vozidlo přijelo, umožňuje pouze jeden směr, stačí klepnout na dané tlačítko (obr. A.4).

V případě, že je možných směrů více, je nutné po zadání vjezdového pruhu označit pruh (rameno), kterým vozidlo křižovatku opustilo (obr. A.5).



Obr. A.4: Záznam průjezdů



Obr. A.5: Zadání výjezdového pruhu

Stisknutím tlačítka „Uložit“ se uloží naměřená data do XML souboru (automaticky jsou data ukládána po minutě).

Prostřednictvím Menu lze zapnout nebo vypnout vibrace při zadávání vjezdových a výjezdových pruhů.

Pro ukončení aplikace je nutno stisknout „Konec“ – tlačítko BACK[15], které používají telefony s Androidem pro ukončování aplikací je **deaktivováno**, aby nedošlo k nechtěnému ukončení aplikace.

[15] Nachází se většinou ve spodní části telefonu.

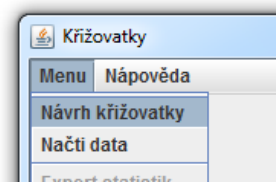
PŘÍLOHA B – UŽIVATELSKÁ PŘÍRUČKA K DESKTOPOVÉ APLIKACI

Spuštění

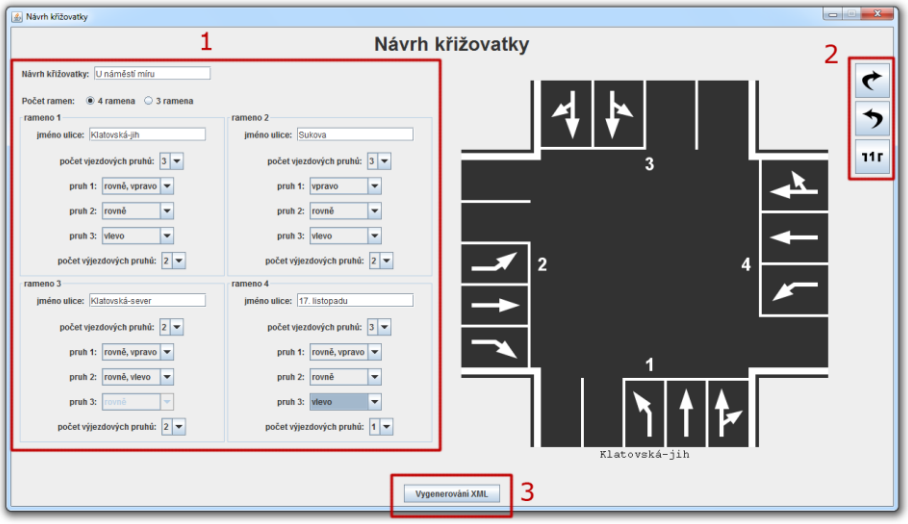
Pro spuštění je nutné mít na počítači nainstalovanou Javu[16]. Aplikace se neinstaluje, stačí rozbalit soubor `KrizovatkyDesktop.zip` a spustit `Krizovatky.jar`.

Návrh křižovatky

V menu vybereme položku „Návrh křižovatky“ (obr. B.1). Nyní lze zadáváním parametrů navrhnout vlastní křižovatku (obr. B.2 – 1). Vjezdové pruhy jsou číslovány zprava doleva po směru jízdy. Tlačítky v pravé části můžeme nákres natáčet (obr. B.2 – 2). Po dokončení návrhu stiskem tlačítka „Vygenerování XML“ vytvoříme soubor s popisem křižovatky, který bude sloužit jako vstup pro mobilní aplikaci (obr. B.2 – 3).



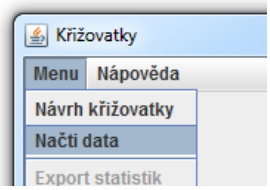
Obr. B.1: Menu



Obr. B.2: Návrh křižovatky

Zobrazení statistik

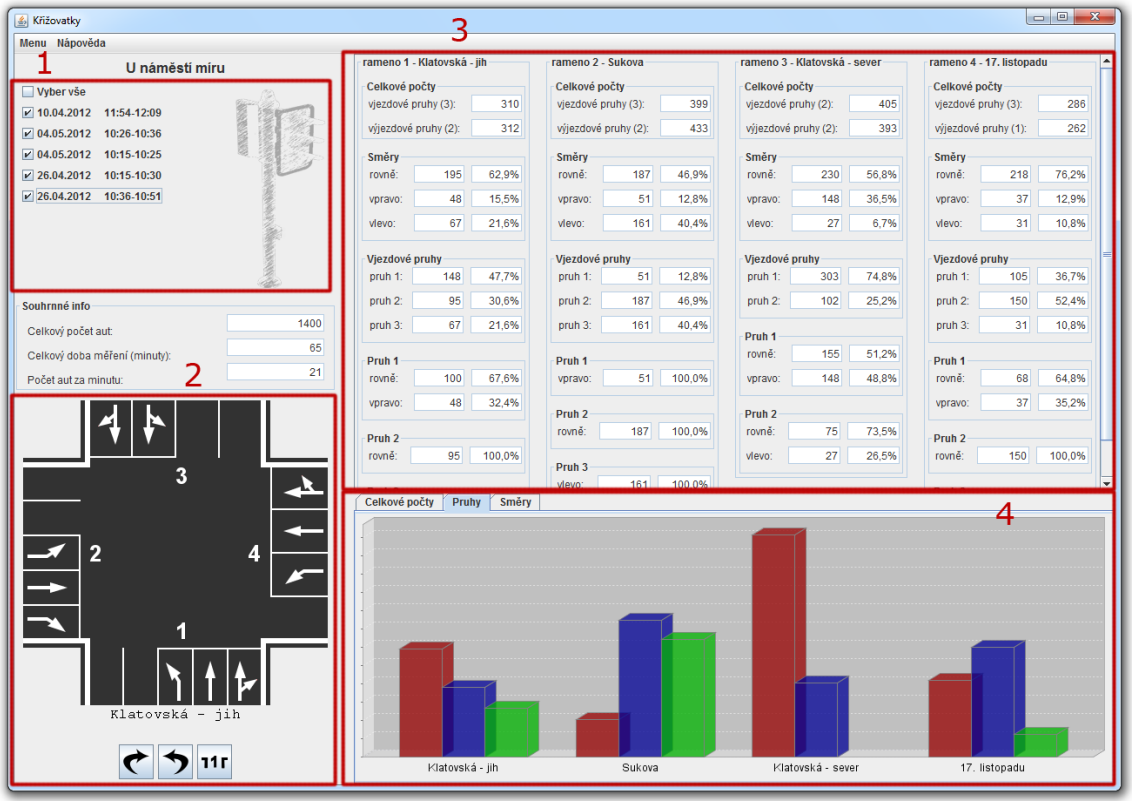
V menu vybereme položku „Načti data“ (obr. B.3). Přidržením klávesy Ctrl a klepnutím myši vybereme všechny soubory, pro které chceme zobrazit statistiky. Po načtení lze pomocí zaškrtávacích políček v levé části (obr. B.4 – 1) vybrat, ze kterých souborů se mají počítat statistiky (obr. B.4 – 3) a zobrazovat grafy (obr. B.4 –



Obr. B.3: Menu

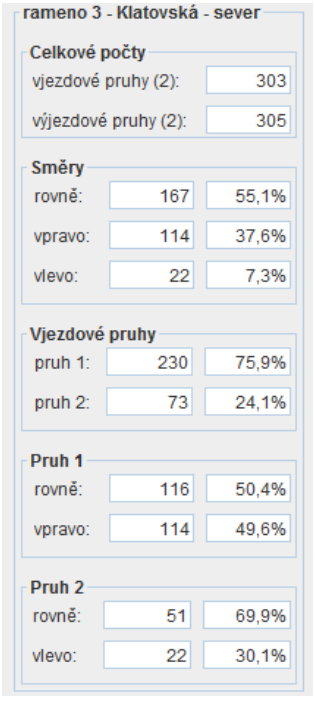
[16] Ke stažení na: http://www.oracle.com/technetwork/java/javase/downloads/index.html

4). Pomocí záložek na horní části panelu s grafy je možné mezi grafy přepínat. V levé dolní části je nákres křižovatky. Při najetí myší na popisek konkrétního pruhu v zobrazených statistikách, se tento pruh zvýrazní na nákresu.



Obr. B.4: Okno se statistikami

Vysvětlivky k zobrazeným statistikám



Celkové počty – celkový počet aut, která projela všemi vjezdovými pruhy a celkový počet aut, která projela výjezdovými pruhy (v závorce vždy uveden počet pruhů).

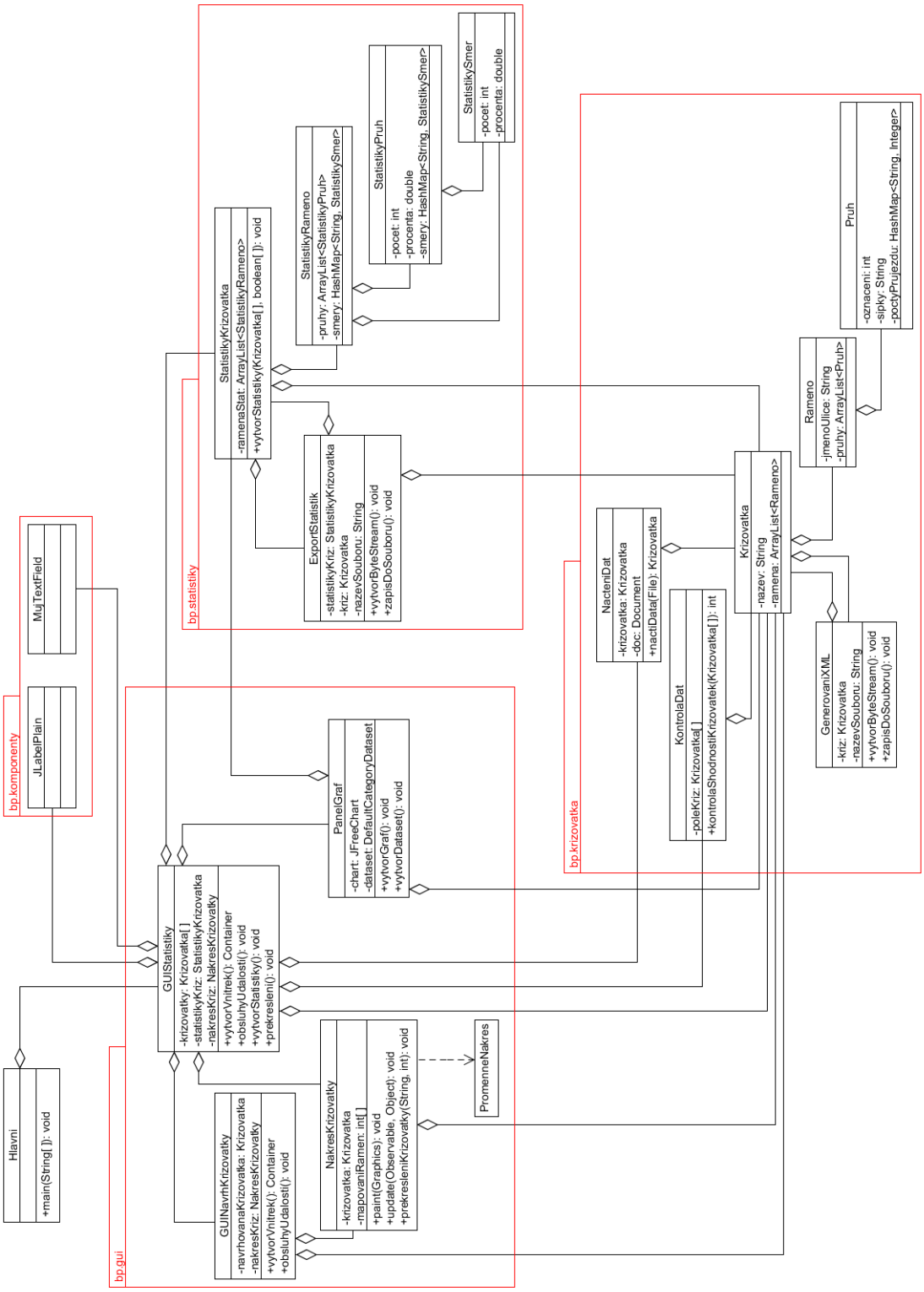
Směry – počty a procenta aut pro jednotlivé směry v rámci celého ramena, bez ohledu na pruhy.

Vjezdové pruhy – počty a procenta projíždějících aut jednotlivými vjezdovými pruhy.

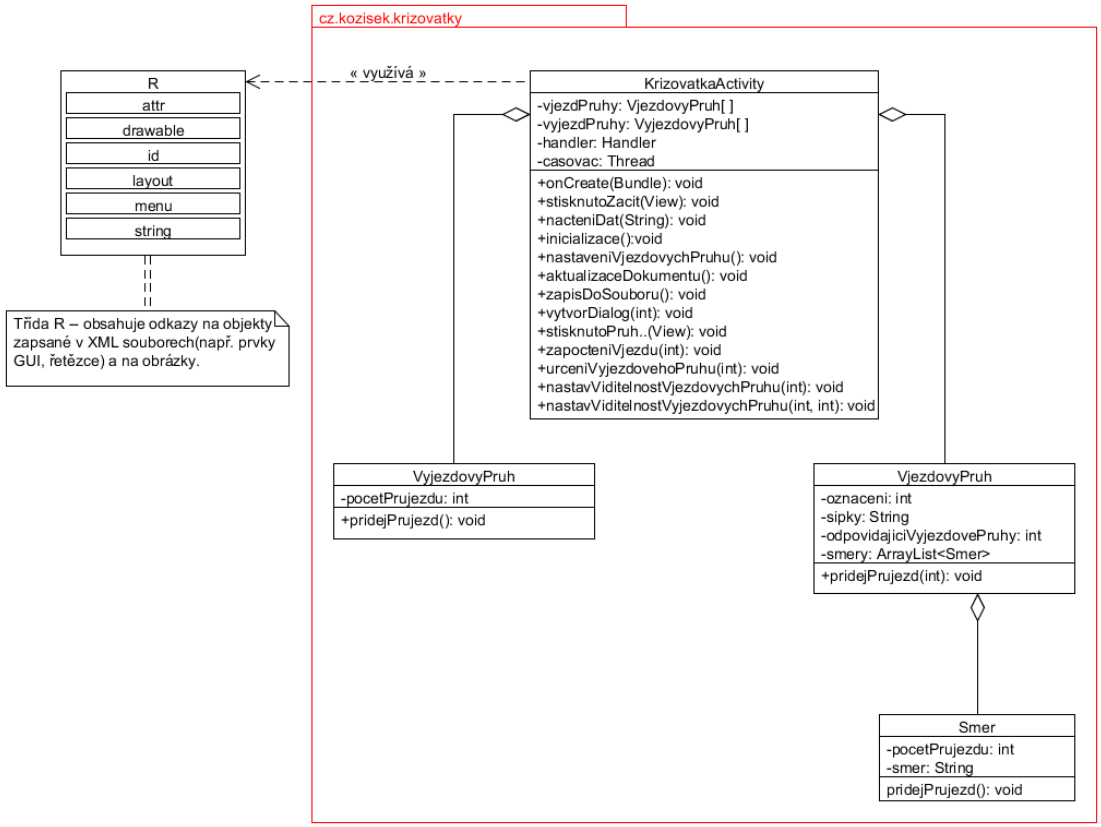
Pruh *n* – počty a procenta aut pro jednotlivé směry v rámci jednoho konkrétního pruhu.

Obr. B.5: Statistiky pro 1 rameno

PŘÍLOHA C – UML DIAGRAMY



Obr. C.1: Desktopová aplikace – zjednodušený diagram tříd (jen nejdůležitější metody a proměnné)



Obr. C.2: Mobilní aplikace – zjednodušený diagram tříd (jen nejdůležitější metody a proměnné)

PŘÍLOHA D – UKÁZKA XML SOUBORU S KŘIŽOVATKOU

```
<?xml version="1.0" encoding="UTF-8"?>
<krizovatka pocetRamen="4" nazev="U náměstí míru">
    <cas casKonce="2012-04-10 12:09" casZacatku="2012-04-10 11:54"/>
    <rameno jmenoUlice="Klatovská - jih" cislo="1">
        <pruhVjezd sipky="RP" oznaceni="11">
            <pocetPrujezdu smer="R">32</pocetPrujezdu>
            <pocetPrujezdu smer="P">15</pocetPrujezdu>
        </pruhVjezd>
        <pruhVjezd sipky="R" oznaceni="12">
            <pocetPrujezdu smer="R">25</pocetPrujezdu>
        </pruhVjezd>
        <pruhVjezd sipky="L" oznaceni="13">
            <pocetPrujezdu smer="L">33</pocetPrujezdu>
        </pruhVjezd>
        <pruhVyjezd oznaceni="1" pocet="2">
            <pocetPrujezdu>103</pocetPrujezdu>
        </pruhVyjezd>
    </rameno>
    <rameno jmenoUlice="Sukova" cislo="2">
        <pruhVjezd sipky="P" oznaceni="21">
            <pocetPrujezdu smer="P">18</pocetPrujezdu>
        </pruhVjezd>
        <pruhVjezd sipky="R" oznaceni="22">
            <pocetPrujezdu smer="R">59</pocetPrujezdu>
        </pruhVjezd>
        <pruhVjezd sipky="L" oznaceni="23">
            <pocetPrujezdu smer="L">54</pocetPrujezdu>
        </pruhVjezd>
        <pruhVyjezd oznaceni="2" pocet="2">
            <pocetPrujezdu>151</pocetPrujezdu>
        </pruhVyjezd>
    </rameno>
    <rameno jmenoUlice="Klatovská - sever" cislo="3">
        <pruhVjezd sipky="RP" oznaceni="31">
            <pocetPrujezdu smer="R">54</pocetPrujezdu>
            <pocetPrujezdu smer="P">53</pocetPrujezdu>
        </pruhVjezd>
        <pruhVjezd sipky="RL" oznaceni="32">
            <pocetPrujezdu smer="R">20</pocetPrujezdu>
            <pocetPrujezdu smer="L">11</pocetPrujezdu>
        </pruhVjezd>
        <pruhVyjezd oznaceni="3" pocet="2">
            <pocetPrujezdu>125</pocetPrujezdu>
        </pruhVyjezd>
    </rameno>
```

```
    <rameno jmenoUlice="17. listopadu" cislo="4">
        <pruhVjezd sipky="RP" oznaceni="41">
            <pocetPrujezdu smer="R">18</pocetPrujezdu>
            <pocetPrujezdu smer="P">14</pocetPrujezdu>
        </pruhVjezd>
        <pruhVjezd sipky="R" oznaceni="42">
            <pocetPrujezdu smer="R">47</pocetPrujezdu>
        </pruhVjezd>
        <pruhVjezd sipky="L" oznaceni="43">
            <pocetPrujezdu smer="L">11</pocetPrujezdu>
        </pruhVjezd>
        <pruhVyjezd oznaceni="4" pocet="1">
            <pocetPrujezdu>85</pocetPrujezdu>
        </pruhVyjezd>
    </rameno>
</krizovatka>
```