Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra kybernetiky

BAKALÁŘSKÁ PRÁCE

Plzeň 2013 **Pavel Bratršovský**

P R O H L Á Š E N Í

Předkládám tímto k posouzení a obhajobě diplomovou/bakalářskou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem bakalářskou/diplomovou práci vypracoval(a) samostatně a výhradně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

V Plzni dne 1.5.2013 .....

Obsah

1. Úvod, anotace

Česky

Tato práce je spojena s firmou Hobl&Pech s.r.o., která se zabývá vývojem softwaru pro řízení a vyhodnocování dopravně-mechanizačního provozu. Cílem firmy bylo vyvinout nové, na trhu zatím neznámé řešení získávání dopravních dat. Řešení by mělo využívat nejnovější technologie, být intuitivní a co nejvíce automatizováno s minimální potřebou lidské interakce. Dále by mělo být maximálně použitelné v praxi, kde se dosud pro záznam dopravních dat velmi často používá formulář ,,Záznam o provozu a výkonu vozidla či mechanizmu", pro který se vžil název ,,stazka". Tato práce se především zaměřuje na automatizační vrstvu takového řešení, která se snaží nahradit lidský člen a dělá tak toto řešení unikátním.

Autor práce více jak rok vyvíjí aplikaci pro automaticky generovaný elektronický záznam o provozu a výkonu vozidla. Vývoj probíhá ve spolupráci s Ing. Jiřím Pechem, který dohlíží na průběh vývoje a navrhuje teorii chování aplikace. Autor pak vytváří teoretický model systému, kde definuje jeho prvky, vazby, logická pravidla a chování. Následně teorii převádí do praxe vytvářením algoritmů a jejich testováním.

English

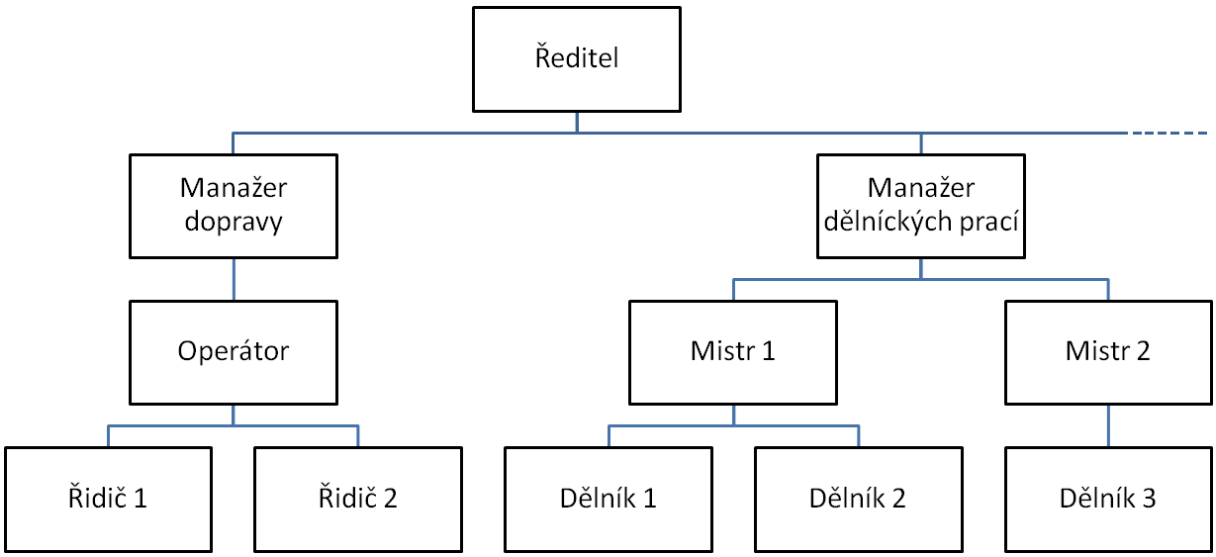
This work is associated with company Hobl & Pech Ltd., a developer of software for the management and evaluation of transport and mechanization operation. Their aim was to develop a new traffic data acquisition solution yet unknown on the market. The solution should use the latest technology to be the most intuitive and automated with minimal human interaction. Furthermore, it should be usable in practice, where is to record traffic data still frequently used form ,,Record of the operation and performance of the vehicle or mechanism". This work is mainly focused on the automation layer of such solution, which tries to replace a human member and makes this solution unique.

Klíčová slova: Automatizace, GPS, Dopravní záznam, Intuitivnost, Android, Java

2. Analýza problematiky

2.1.1. Proč firmy potřebují sledovat vozidla a řidiče

Firmu neboli podnik lze většinou popsat jako soubor lidí, strojů, vozidel a dalších se stejným cílem, kterým je prospěch firmy. Podnik lze také definovat jako jednotný systém skládající se z menších subsystémů, částí či členů podniku. Tento soubor členů lze nejjednodušeji zobrazit jako hierarchický graf, kde na vrcholu grafu je ředitel podniku, pod ním jsou manažeři, pod nimi další členové atd. Každý prvek tohoto grafu má dohled či kontrolu nad prvky pod ním. Na obrázku 1 je zobrazen příklad malé, imaginární firmy, kde je jasně vidět, že dělníci mají kontrolu nad stroji, mistr má dohled nad dělníky, manažer nad mistry a ředitel firmy nad manažery.



Obrázek 1

Z předchozího je tedy jednoduché si uvědomit, že pokud se díváme na podnik jako na systém a chceme, aby tento systém fungoval co nejlépe, je nutné zajistit správnou funkčnost všech prvků systému. Dále lze vyčíst, že pro takovýto systém je nejvíce kritická funkčnost nejnižších prvků, tzn. pokud nebudou stroje, řidiči a dělníci pracovat dle očekávání a potřeb, podnik nepracuje správně i přes správné chování mistrů, manažerů a ředitele.

Pokud se zaměříme na dopravní firmy, bude pro jejich fungování kritické zajistit provozuschopnost vozidel a požadované chování řidičů těchto vozidel. Fungování strojů lze

zajistit vcelku jednoduše, jelikož stroj funguje dle očekávání, dokud se neprojeví mechanická či jiná porucha. Člověk si však většinou snaží ulehčit práci a proto je jeho chování často nevyzpytatelné i přesto, že je ve výborném stavu. Nelze se tedy divit, že dopravní firmy chtějí mít přehled nad tím, jak vykonávají řidiči jejich vozidel svou práci, jak jsou efektivní a zda se nesnaží přilepšit si na úkor samotného podniku.

2.1.2. Co je potřeba sledovat

Nejobecnější úlohou dopravní firmy je přeprava nákladu z bodu A do bodu B, o kterém chceme získat informace pro výpočet fakturace zákazníkovi, mzdy pracovníka a vyhodnocení efektivity a správnosti provedené práce. Tím je dán nezbytný okruh strukturovaných informací o vozidle a řidiči. Lze navrhnout řešení typu získávání zpráv o vozidle v intervalech, například mobilním telefonem, a jejich následné vyhodnocování. Dalším řešením by mohla být instalace kamer do kabiny vozidla. Tato řešení jsou však naprosto nepoužitelná, ať už z důvodu nepraktičnosti, lidských práv, nedostatku použitelných informací, či mnoha dalších. Potřebujeme tedy nalézt přijatelné řešení, které poskytne relevantní informace o vozidle a řidiči použitelné pro další zpracování.

U vozidla jsou důležité informace o jeho stavu, například tedy, kde se nachází, zda stojí či jede, zda provádí nějakou činnost a další. U řidiče jsou nejdůležitější osobní informace a informace o činnostech s vozidlem spojené. Tyto informace nazveme dopravní data. Jak vypadá záznam těchto dat, je uvedeno v následující kapitole.

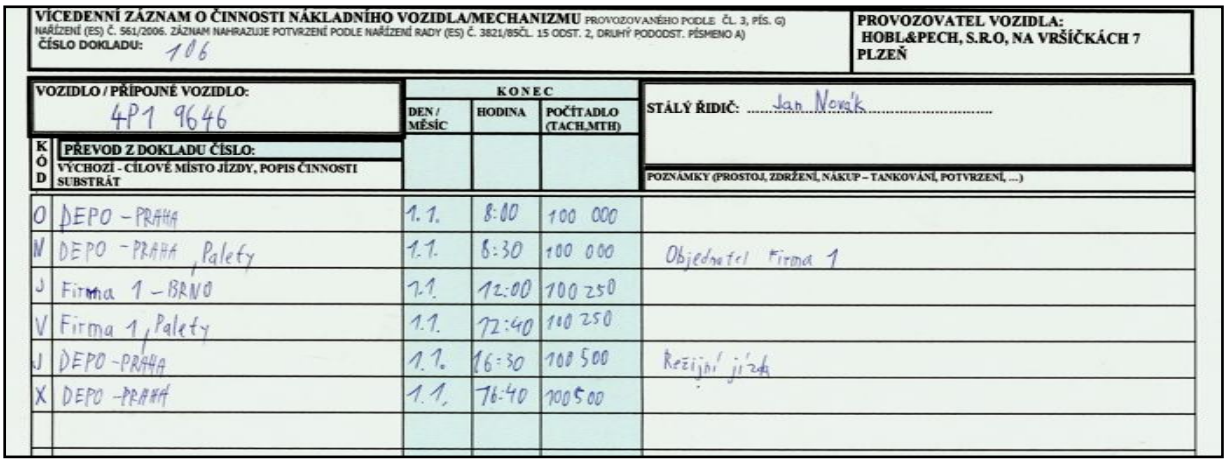
Ideální situace by byla, kdyby dohlížející pracovník měl přístup k dopravním datům v reálném čase, to by však znamenalo, že by musel v reálném čase kontrolovat, zda vozidlo či řidič jedná správně. Je tedy jasné, že pro dohlížejícího pracovníka je důležitý záznam těchto dopravních dat v časové ose celé úlohy přepravy nákladu z bodu A do bodu B. Možnost zjištění dat v reálném čase je však důležité ponechat pro možnost okamžité kontroly, tím se však dostáváme již mimo naši oblast řešené problematiky.

Výše uvedené tedy shrneme na následujícím příkladu. Operátor pověří řidiče převozem zboží z Prahy do Brna. Řidič vyjede s nákladem z Prahy a během jízdy zaznamenává svou činnost do záznamu. Operátor může kdykoli zavolat řidiči a zjistit, kde se

právě nachází a zdali zakázku stíhá splnit. Řidič vyloží náklad v Brně, vrátí se zpět do Prahy a odevzdá operátorovi záznam dopravních dat. Manažer pak ze záznamu může provést fakturaci a ověřit efektivitu a správné chování řidiče. Tím je myšleno, že pokud by řidič jel do Brna oklikou a ujel tak o 100 Km více, bylo jeho chování nesprávné a manažer by byl schopen tuto chybu rozpoznat. Dále ze záznamu může zjistit, zda se řidič nesnažil obohatit se na firmě, například odčerpáním pohonných hmot z vozidla.

2.1.3. Jak vypadají nynější záznamy

I přes to, jaký je v dnešní době rozmach informačních technologií, stále se kvůli absenci digitalizované podoby dopravních dat nejčastěji používají řidičem ručně psané záznamy dopravních dat. Pro takovýto dokument, obsahující záznamy dopravních dat s vozidlem spojené, se zažil název „stazka". Jedna „stazka" obvykle ohraničuje časový interval od příchodu obsluhy k vozidlu až po jeho odstavení po splnění zakázek zpět do depa. Tento časový interval je v praxi většinou jedna pracovní směna. Jak vypadá takto psaná „stazka" k příkladu z předchozí kapitoly, je ukázáno na obrázku 2.



VÍCEDENNÍ ZÁZNAM O ČINNOSTI NÁKLADNÍHO VOZIDLA/MECHANIZMU PROVOZOVANÉHO PODLE ČL. 3, PÍS. G) NAŘÍZENÍ (ES) Č. 561/2006. ZÁZNAM NAHRAZUJE POTVRZENÍ PODLE NAŘÍZENÍ RADY (ES) Č. 3821/85ČL. 15 ODST. 2, DRUHÝ PODODST. PÍSMENO A)

ČÍSLO DOKLADU: 106

PROVOZOVATEL VOZIDLA: HOBL&PECH, S.R.O, NA VRŠÍČKÁCH 7 PLZEŇ

VOZIDLO / PŘÍPOJNÉ VOZIDLO: 4P1 9646

KONEC

STÁLÝ ŘIDIČ: Jan Novák

KÓD	PŘEVOD Z DOKLADU ČÍSLO: VÝCHOZÍ - CÍLOVÉ MÍSTO JÍZDY, POPIS ČINNOSTI SUBSTRÁT	DEN / MĚSÍC	HODINA	POČÍTADLO (TACH,MTH)	POZNÁMKY (PROSTOJ, ZDRŽENÍ, NÁKUP – TANKOVÁNÍ, POTVRZENÍ, ...)
O	DEPO - PRAHA	1.1.	8:00	100 000	
N	DEPO - PRAHA, Palety	1.1.	8:30	100 000	Objednatel Firma 1
J	Firma 1 - BRNO	1.1.	12:00	100 250	
V	Firma 1, Palety	1.1.	12:40	100 250	
J	DEPO - PRAHA	1.1.	16:30	100 500	Režijní jízda
X	DEPO - PRAHA	1.1.	16:40	100 500	

Obrázek 2

2.1.4. Možnosti použití technologií dnes již používaných

Ručně psané záznamy dopravních dat jsou nepraktické, zastaralé a je potřeba je ručně digitalizovat pro strojové zpracování, což je pro podnik zdrojem dalších nákladů.

Pro nejjednodušší a nejlevnější nasazení do praxe je vhodné využít již dnes dostupných a hojně využívaných technologií. Pro nás důležitou technologií je globální družicový polohový systém se známou zkratkou GPS [1]. Díky této technologii je dnes možné v mnoha zařízeních zjistit svou aktuální polohu a čas kdekoli na Zemi s přesností na jednotky metrů. Tato technologie je dnes využívána ve sledovacích zařízeních, GPS navigacích, chytrých telefonech a dalších.

Mnoho firem nabízí jednotky montovatelné do vozidla, které zaznamenávají polohu vozidla a dokážou ji poslat například na mobilní telefon. Pouze polohové informace jsou jako dopravní data nedostačující a proto je potřeba získávat ostatní data i jiným způsobem. Existuje možnost montáže jednotky do vozidla, která zaznamenává všechny zjistitelné informace o vozidle, od aktuální rychlosti až po informaci, zda jsou otevřené dveře. Bohužel ani tyto informace nenaplní množinu potřebných dopravních dat a proto je v nynější době potřeba alespoň minimální interakce s řidičem vozidla pomocí terminálu.

Samozřejmě jsou na trhu firmy, které nabízí jednotky do vozidel i s terminály, avšak toto řešení pouze digitalizuje ručně psané záznamy, tudíž není automatizováno a nesnaží se tak eliminovat lidské chyby při obsluze zařízení ani řidiči usnadnit obsluhu. Lze uvažovat nad možností montáže podobné jednotky a terminálu do vozidla a instalace vlastního automatizačního softwaru. To by však nebylo nejlevnější ani nejjednodušší řešení, a to z důvodu nutnosti omezení se na jeden typ zařízení.

Nejlepším řešením je využít technologie chytrých telefonů či tabletů, které mají v dnešní době obrovský rozmach. Obecně lze říci, že téměř každý chytrý telefon či tablet obsahuje GPS technologii pro zjišťování polohy, GPRS a WiFi technologii pro bezdrátovou výměnu informací a dotykovou obrazovku pro interakci s uživatelem. Máme tedy zařízení schopné zjistit velkou část dopravních dat a terminál pro doplnění dat v jednom. Navíc je dost pravděpodobné, že uživatel takovéto zařízení již vlastní z osobních důvodů, a tudíž je toto řešení vcelku levné.

Nalezli jsme tedy nejlepší řešení pro automatizování získávání dopravních dat, kdy stačí pouze vyvinout software pro chytré telefony či tablety.

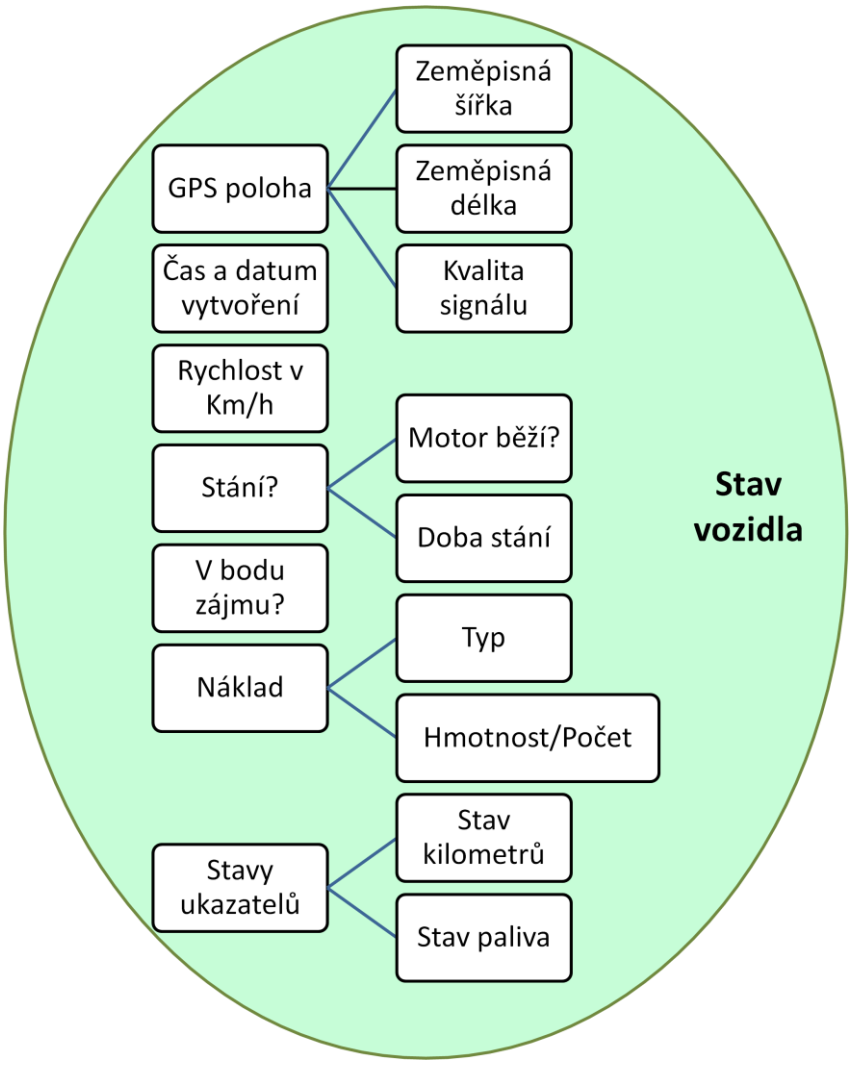
2.2. Definice datové základny

Před samotným vývojem softwaru je nutné si definovat teorii logických pravidel požadovaného chování a datovou základnu. Datovou základnou je myšlena definice potřebných informací, jejich vzájemné vazby, definice objektů, se kterými bude software pracovat, popřípadě základní nezbytné funkce. Dále do datové základny patří veškeré číselné, textové či pravdivostní proměnné, které bude software využívat.

2.2.1. Okamžitý stav vozidla

Stavem vozidla je myšlena množina veškerých proměnných vztahujících se ke konkrétnímu stavu vozidla. Je nutné si uvědomit, že stav vozidla je jedinečný a neopakovatelný, protože se váže k času vytvoření. Pokud tedy budou v nějakém časovém intervalu, například jedné sekundy, zaznamenány dva stavy stojícího vozidla, nebudou totožné i přes to, že se s vozidlem nic nedělo, jelikož budou pořízené v různých časových okamžicích.

Dále je dobré si definovat přístup k této množině. Přístupů je samozřejmě více, ale z programátorského hlediska budeme na tuto množinu pohlížet jako na objekt. Pokaždé, když bude zaznamenán stav vozidla, bude vytvořen nový objekt, ke kterému se budou vázat hodnoty jednotlivých proměnných či další objekty. Ilustrace takového, námi nadefinovaného objektu je znázorněna na obrázku 3.



Obrázek 3

Objekt stav vozidla obsahuje:

GPS poloha - objekt obsahující číslo zeměpisné šířky, délky a informaci o kvalitě signálu.

Čas a datum vytvoření - čas a datum vytvoření stavu vozidla.

Rychlost v km/h - rychlost vozidla při pořízení stavu.

Stání? - pravdivostní hodnota vázající se k rychlosti. K této hodnotě se dále váže, zda je nastartováno a jak dlouho již vozidlo stojí.

V bodu zájmu - Ukazatel na objekt. Pokud se vozidlo nachází v bodu zájmu, ukazuje na tento bod zájmu. Více v kapitole definice bodu zájmu.

Náklad - seznam objektů obsahující informace o typ a množství každého nákladu.

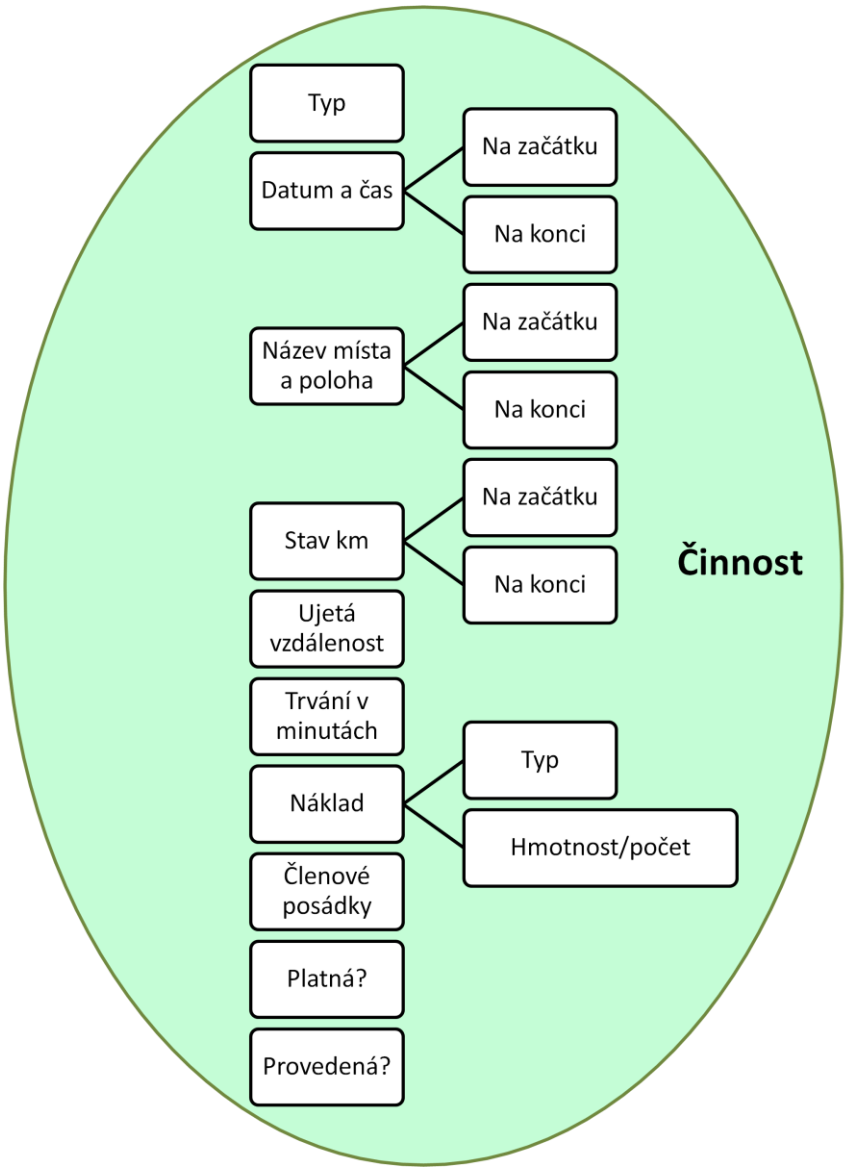
Stavy ukazatelů - číselné údaje o stavu ujetých kilometrů a množství litrů paliva v nádrži.

2.2.2. Definice činnosti

Již víme, jak si představit stav vozidla. Nyní definujeme činnost s vozidlem spojenou. U dopravních podniků mohou být činnosti s vozidlem různé, ale pro naše účely bude stačit následujících 12 typů:

- **Příprava před jízdou**
- **Jízda**
- **Nakládka**
- **Vykládka**
- **Mechanizační činnost/Práce**
- **Činnost mimo vozidlo**
- **Prostoj** - lze dále dělit na prostoj dopravce, přepravce, bezpečnostní přestávku a spánek.
- **Tankování**
- **Oprava**
- **Soukromá jízda**
- **Odstavení vozidla**
- **Výměna řidičů**
- **Neznámá činnost** - akce, u které zatím nebyl zjištěn její typ.

Každá činnost bude tedy jednoznačně určena jejím typem. Tento údaj ovšem nestačí, proto stejně jako u stavu vozidla, definujeme činnost jako objekt obsahující další objekty či proměnné. Na rozdíl od stavu vozidla se činnost váže k časovému intervalu, a proto většina proměnných jsou vlastně objekty obsahující údaje o stavu proměnné na začátku a konci činnosti. Lze také předpokládat, že činnost může být plánovaná a objekty činnost mohou být vytvořeny ještě před jejich vykonáním. Znázornění je na obrázku 4.



Obrázek 4

Obsah objektu činnost:

Typ - celočíselné číslo typu činnosti.

Datum a čas - objekt obsahující hodnoty na začátku a konci činnosti.

Název místa a poloha - objekt obsahující informace o poloze a názvu místa. Více o názvu místa v následující kapitole.

Stav km - objekt s informacemi o stavu ukazatele ujetých kilometrů.

Ujetá vzdálenost - počítána ze stavu ujetých kilometrů.

Trvání v minutách - počítáno z času začátku a konce.

Náklad - stejné jako u stavu vozidla.

Členové posádky - seznam členů posádky vozidla v době činnosti. Potřeba vědět kvůli účtování.

Platná? - pravdivostní proměnná, zda je plánovaná činnost platná. Váže se k maximálnímu datu a času splnění.

Provedená - pravdivostní proměnná, zda je činnost již provedená, či stále probíhá, nebo ještě nezačala.

2.2.3. Definice bodu zájmu

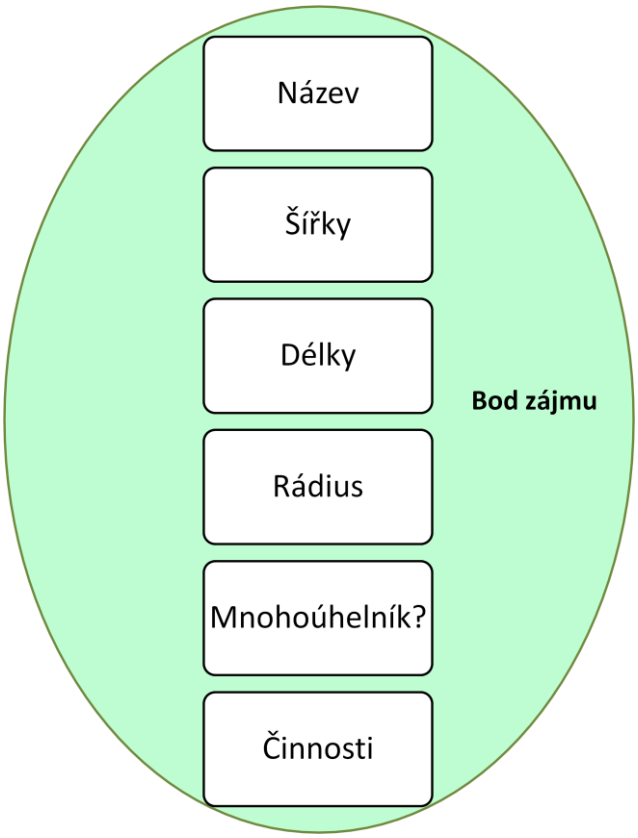
Bod zájmu definujme jako známé místo jednoznačně určené svou polohou, názvem a obvyklými činnostmi v místě prováděných.

Již jsme definovali, jak vypadá základní datová základna záznamů „stazek". Pro potřebu plného přiblížení se těmto záznamům a pro komfort uživatele nestačí u jednotlivých činností pouze GPS záznam polohy, ale je také nutné znát název místa spojeného s polohou. Pokud tedy řidič naloží náklad u nějaké firmy, v záznamu by měl být název firmy či ulice. V digitálním záznamu pak bude zaznamenán jak název místa, tak GPS souřadnice.

Pro potřeby automatizace je však nutné mít databázi názvů míst spojených s GPS polohou. Bez takovéto databáze by byl řidič či kontrolor „stazek" nucen pokaždé zadávat názvy těchto míst. Nabízí se možnost vytvořit takovou databázi ručně z názvů obcí a firem a přiřadit jim reálné GPS souřadnice. Pokud nebereme v potaz časovou náročnost vytváření takové databáze, narazíme na zásadní problém. V jednom daném místě v okruhu 10 metrů se může vyskytovat více firem, křížit ulice a každá dopravní firma v tomto místě může mít jiný obchodní záměr. Z tohoto důvodu je nutné, aby se pro každou firmu tvořila vlastní databáze míst. Pro co největší automatizaci je nutné zajistit učící se schopnost aplikace a možnost synchronizace více vytvořených databází v rámci podniku. Nyní se zaměříme na schopnost učení se.

Uvažujme případ, kdy řidič provede nějakou činnost na zatím neznámém místě. Automaticky zaznamenáme GPS souřadnice místa a činnost s ním spojenou. Dále nabídneme možnost řidiči, aby zadal název místa. Pro firmu je název od řidiče nejužitečnější, jelikož řidič ví přesně, za jakým účelem na daném místě je. Tímto vytvoříme objekt, který nazveme bod zájmu. Pokud tento bod zájmu uložíme a dokážeme synchronizovat data v rámci celé firmy, jakýkoli zaměstnanec, když přijede do stejného místa, nebude nucen nic zadávat, jelikož aplikace již bude tento bod zájmu znát. Navíc bude mít informaci o nejčastějších činnostech v tomto místě prováděných.

Každý bod zájmu lze na mapě zobrazit buďto jako kruh, kde jeho střed je GPS souřadnice zaznamenaná v době vytvoření, či jako mnohoúhelník přesně označující areál firmy, ulici a další. Na obrázku 5 je znázorněn objekt bod zájmu.



Obrázek 5

Obsah objektu bod zájmu:

Název - řetězec názvu daného místa.

Šířky - pole obsahující zeměpisné šířky bodu zájmu. Pokud je definován jako kruh, pole obsahuje pouze jednu hodnotu.

Délky - pole obsahující zeměpisné délky bodu zájmu. Pokud je definován jako kruh, pole obsahuje pouze jednu hodnotu.

Rádius - pokud je bod definován jako kruh, je to celočíselná hodnota poloměru tohoto kruhu.

Mnohoúhelník - pravdivostní hodnota, zda je bod zájmu definován jako mnohoúhelník.

Činnosti - seznam celočíselných hodnot činností prováděných v tomto místě.

2.2.4. Definice záznamu

Budeme vycházet z praxe a již popsaného ručně psaného záznamu „stazky". Takový záznam obsahuje takzvanou hlavičku a řádky. Hlavička je množina informací konstantních pro celou „stazku", kterou je „stazka" jednoznačně identifikována. Řádky jsou časově řazené činnosti s vozidlem či řidičem spojené. Budeme se dále držet tohoto standardu a pokusíme se plně přiblížit digitální záznam záznamu ručně psanému. Definujeme objekt hlavička, který se vytvoří po přihlášení řidiče do aplikace. Digitální záznam „stazky" pak bude tvořit hlavička a seznam provedených činností, což odpovídá záznamu ručně psanému. Objekt hlavička ilustrujeme na obrázku 6.

Obrázek 6

Obsah objektu hlavička „stazky":

Tažné vozidlo - objekt obsahující informace o SPZ, evidenčním čísle a přírůstku či úbytku u stavů ukazatelů vozidla.

Datum a čas - časový interval ohraničující celou „stazku".

Členové posádky - pracovníci, kteří byli v intervalu „stazky" členy posádky vozidla.

Přípojné vozidla - objekty s informacemi o přípojných vozidel či zařízeních použitých v intervalu „stazky".

Objednatelé - seznam objednatelů hrajících roli v rámci „stazky".

2.2.5. Ostatní proměnné

Aplikace bude samozřejmě využívat mnoho dalších objektů a proměnných. Také lze předpokládat, že v průběhu vývoje se budou proměnné přidávat či odebírat. Následuje seznam nejdůležitějších objektů a proměnných zatím nezmíněných.

- **GPS modul** - objekt pro získání dat z GPS technologie pro aplikaci.
- **Plán** - objekt obsahující seznam naplánovaných činností
- **Substráty** - objekty uchovávající informace o nákladu plánovaném či naloženém
- **Body zájmu** - seznam bodů zájmu nahraný z databáze či souboru.
- **Známí řidiči a vozidla** - seznam známých řidičů a vozidel používající konkrétní zařízení.
- **Nynější a předchozí stavy** - nynější a několik předchozích stavů vozidla.
- **Časy jízdy a bezpečnostních přestávek** - pro kontrolu zákonem daných pravidel.
- **Nejpravděpodobnější činnost** - aproximovaná nejpravděpodobnější činnost právě prováděná.
- **Bluetooth modul** - objekt pracující s jednotkou vozidla.
- **Konstanty** - definované konstanty pro přepočty a přizpůsobení aplikace.

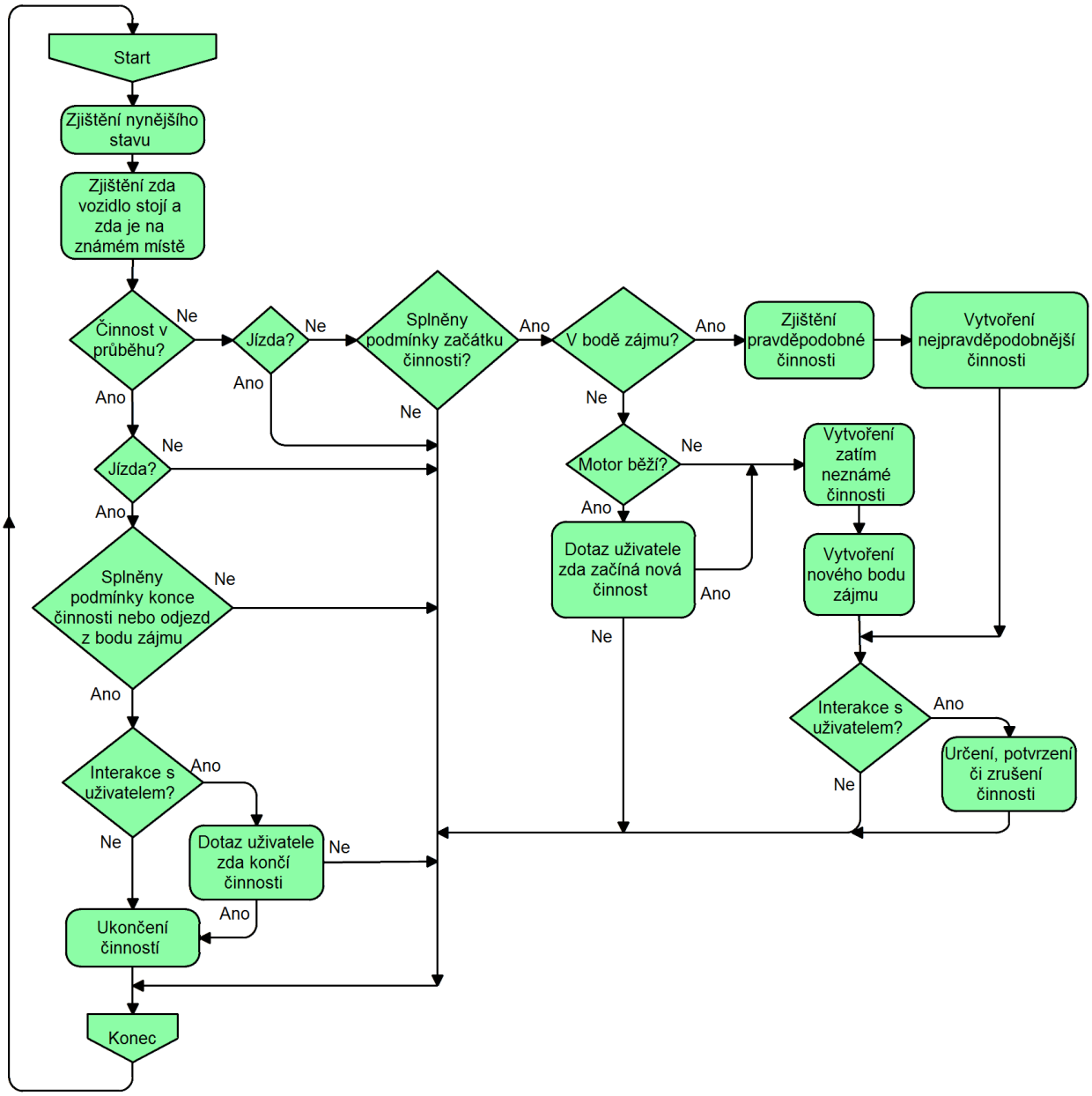
2.3. Definice logických pravidel

Pod pojmem logické pravidlo lze rozumět výběr možností v jisté situaci na základě známé informace. Jednoduchým příkladem je logické pravidlo „pokud mám hlad, najím se". Dalším typem logického pravidla je činnost vždy spojená s jiným pravidlem. Například „pokud jím, je jisté, že pohybuji ústy". Tato logická pravidla se v programátorské logice zobrazují jako vývojové diagramy. Ty popisují jednotlivé kroky algoritmu. Logická pravidla aplikace popíšeme právě těmito diagramy.

Budeme se zabývat pouze automatizovanou vrstvou aplikace, nikoliv její ostatní funkčností. Předpokládejme tedy vytvořenou aplikaci ukládající uživatelem ručně zadávané záznamy. Aplikace nemá žádné známky automatizace a není intuitivní, avšak řidič je schopen ručně vytvořit plnohodnotný digitální záznam „stazky". Zaměříme se tedy na automatizování

aplikace tak, abychom co nejvíce odstranili nutnost interakce s řidičem. Pokud si toto představíme jako vrstvy, nasadíme automatickou vrstvu nad vrstvu manuálního zadávání řidiče.

Nyní převedeme celu automatickou vrstvu do vývojových diagramů. Algoritmus této vrstvy navrhneme tak, aby cyklicky prováděl takzvané životní cykly. To tedy znamená, že například každou sekundu proběhne celý vývojový diagram, zobrazený na obrázku 7.



Obrázek 7

2.4. Zjištění možností aplikace

Mobilních platforem je dnes rozšířeno mnoho, a proto je nutné zanalyzovat, na jakou platformu se zaměříme. Opět se budeme snažit najít jednoduché a co nejvíce rozšířitelné řešení.

2.4.1. Vývoj aplikací na Windows CE, iOS, Android

Následující tři operační systémy jsou na přenosných zařízeních nejznámější a nejrozšířenější.

2.4.1.1. Windows CE a Windows Mobile

Windows CE [2] je operační systém vyvinut firmou Microsoft pro starší typy chytrých telefonů jako jsou PDA či Pocket PC. V současné době je na chytrých telefonech tento systém nahrazen systémy Windows Mobile [3] a Windows Phone, které jsou na Windows CE založeny. Všechny mají však stejné rysy. Uživatelsky se podobají systému Windows pro osobní počítače, uživatelé tedy většinou prostředí znají. Aplikace pro tyto operační systémy jsou vyvíjeny buďto v programovacím jazyku C++, či C# nebo Visual Basic.

2.4.1.2. iOS

Společnost Apple používá na svých zařízeních, jako jsou iPhone, iPad či iPod, vlastní operační systém iOS [4]. Tento systém je na rozdíl od rodiny Windows CE [2] stabilní a velmi vyladěný. Aplikace jsou pro iOS vyvíjeny v jazyku C, či Objective-C. Nevýhodou je, že iOS většinou operuje pouze na zařízeních od firmy Apple a jeho kombinace s chytrým telefon jiného výrobce je vzácná.

2.4.1.3. Android

Android [5] není jen operační systém, ale celá open source platforma pro mobilní zařízení. Je založena na operačním systému Linux. Hlavním vývojářem je firma Google. Tento

systém je použitelný v mnoha zařízeních, s různým hardwarem či velikostí obrazovky. Je uživatelsky a díky své otevřené licenci i programátorsky velmi přívětivý. Také v současné době dominuje mezi používanými operačními systémy na chytrých mobilních zařízeních. Hlavním programovacím jazykem pro Android je Java [6], lze však programovat i v jazyce C. Android nabízí programátorům vývojový balík, který velmi zpřístupňuje vývoj aplikací na tento systém a nabízí i vlastní emulátor mobilního zařízení pro testování.

2.4.2. Proč zvolit Android, rozdíl mezi tabletem a telefonem

Když vezmeme v potaz předchozí informace a také fakt, že autor práce má víceleté zkušenosti s programovacím jazykem Java, je volba dominantního operačního systému Android jako cílového operačního systému jasná. Pro naše účely je velmi výhodná kompatibilita platformy, kdy naše aplikace bude pracovat stejně dobře na chytrém telefonu jako na tabletu jakéhokoli výrobce. Vzhledem k dominaci tohoto systému je také velká pravděpodobnost, že potencionální uživatel již zařízení s tímto systémem vlastní.

Rozdíl mezi tabletem a chytrým telefonem v dnešní době začíná být čím dál menší. Tablet lze definovat jako malý, přenosný počítač bez klávesnice. Chytrý telefon neboli „smartphone" ale v dnešní době disponuje stejnými hardwarovými komponentami jako klasický počítač. Obecně lze říci, že rozdíl mezi těmito zařízeními je hlavně ve velikosti, výkonnosti a možnosti volání. Chytré telefony se díky své velikost vejdou do kapsy, nejsou tak výkonné jako tablety, ale disponují možností volání a psaní SMS. Na druhou stranu lze na trhu najít tablety, které jsou jen o pár centimetrů větší než „smartphone" a možnost volání a SMS mají také. Nicméně pro klasického uživatele stačí vědět fakt, že tablet se více blíží počítači a „smartphone" mobilnímu telefonu.

2.4.3. Možnosti a přesnost geolokačního modulu

Otázka ohledně GPS technologie v jednotlivých chytrých telefonech či tabletech je záludná, jelikož každý výrobce instaluje do svého produktu různé GPS moduly. Ty mají odlišnou přesnost či výkonnost. Zaměříme se tedy na standardní, průměrný typ GPS používaný v pro nás cílových zařízeních.

Důležité je počáteční určení zatím neznámé polohy zařízení, v ideálních podmínkách je pak následná aktualizace známé polohy téměř okamžitá. GPS funguje u mobilních zařízení podobně jako u samotných automobilových navigací, pouze s malými odlišnostmi. Po zapnutí GPS technologie v našem zařízení s operačním systémem Android se systém bude snažit určit aktuální polohu. Nejprve se pokusí najít přibližnou polohu použitím technologií GPRS a WiFi. Pokud jsou tyto technologie aktivní, systém okamžitě ví, přes jaký přístupový bod jsou připojeny, a tak zjistí aktuální polohu v řádu kilometrů. Pokud aktivní nejsou, bude určovat aktuální polohu takzvaně od nuly. GPS modul tedy v obou případech začne hledat dostupné satelity pro určení polohy. Obecně platí, čím více satelitů, tím přesnější určení. Po dosažení určité přesnosti již dává GPS modul systému informace o platné poloze a přesnost se s přibývajícími satelity dále zlepšuje.

Přesnost a rychlost určení polohy závisí na mnoha faktorech. Hlavní z nich je samozřejmě výkonnost hardwaru. Podstatné jsou však proměnné faktory jako je počasí, poslední známá poloha, či přístupnost GPRS nebo WiFi. Je logické, že přístupnost satelitu na oběžné dráze Země přímo závisí s tím, jestli jsme v budově či venku, jestli je jasno či bouřka, zdali máme zařízení v kapse či namířené k obloze. Dalším faktorem je již zmíněná přibližná poloha pomocí ostatních technologií. Bez této informace bude rychlost určení samozřejmě podstatně menší. Následuje seznam několika případů týkajících se přesnosti a rychlosti určení počáteční hodnoty, určený z experimentálního měření provedeného autorem samotným.

Přesnost:

- Venku, jasno - 5 až 10 metrů
- Venku, zataženo - 10 až 15 metrů
- V budově u okna - 20 až 30 metrů
- V budově daleko od okna - 50 a více metrů
- Zařízení v kapse, tašce atd. - zhoršení přesnosti o 5 až 30%

Rychlost prvního určení polohy:

- Platná známá poslední polohy či připojeno GPRS nebo WiFi - 1 až 30 sekund
- Neznámá poslední poloha - 20 sekund až 5 minut

Je jasné, že existuje mnoho kombinací různých případů, a proto je rychlost a přesnost prvotního určení polohy nepředpověditelná. Pokud již známe prvotní polohu, zjištění nové polohy většinou není problém i při slabém signálu, ale s malou přesností. Problém nastává pouze v případech jako je vjezd do tunelu, kdy zařízení ztratí veškerý signál. Po výjezdu z tunelu však opět velmi rychle nalezne novou pozici. Dále bychom se mohli zabývat vlivem této technologie na výdrž baterie zařízení, to však přesahuje rámec této práce.

3. Vytvoření aplikace

Tak jako motor se nedá v pohyb beze zbytku vozidla, tak automatizační jádro nelze vytvořit bez vytvoření aplikace samotné. Autor této práce vytváří aplikaci pro záznam elektronických dopravních záznamů ,,ES-Hobl&Pech" od února roku 2012. Aplikace je stále ve vývoji a v dnešní době je to již její třetí generace. Pod pojmem generace je myšleno sestavení celé aplikace od nuly, nikoliv pouze vylepšení algoritmu. V rámci vývoje již bylo vyvinuto mnoho verzí, ale v průběhu testování a rozšiřování teorie během praxe bylo potřeba aplikaci vylepšovat. Více o nynějším stavu v kapitole 5. Jak již bylo řečeno, předmětem této práce je vývoj jádra aplikace, zaměříme se tedy zejména na toto jádro.

3.1. Jakou funkci zastává jádro na pozadí v aplikaci

V kapitole 2.3 jsme si definovali logické chování automatické vrstvy, říkejme jí z pohledu aplikace jádro. Toto jádro se drží těchto definovaných logických pravidel a spravuje aplikaci jako takovou.

Po spuštění aplikace se vytvoří a inicializuje jádro. Tím se předvytvoří veškeré globální objekty, seznamy a proměnné aplikací používané. Aplikace pak k této datové základně přistupuje právě přes jádro. Některé proměnné či objekty se deklarují již při vytvoření. Dále převezme práva nad některým hardwarem zařízení jako je GPS, externí úložiště, ovládání obrazovky a další. Nakonec nastaví vstupně výstupní souborové operace a nahraje vstupní soubory z úložiště aplikace.

Po vytvoření jádra se vytvoří vlákno jádra. Vláknem je myšlen algoritmus, který se bude cyklicky provádět. Tento algoritmus plně koresponduje s námi nadefinovaným vývojovým diagramem automatické vrstvy. Dále pak kontroluje časy bezpečnostních přestávek, spravuje světlost displeje kvůli výdrži baterie, vypočítává ujeté kilometry, hlídá splnění časově omezených akcí, zapisuje polohy pro pozdější zobrazení jízdy na mapě a obstarává zobrazení aktuálních dat uživateli. Toto vlákno je cyklicky provedeno každou sekundu.

Celý algoritmus jádra je samozřejmě mnohem komplikovanější a je napsán na více jak 800 řádcích programátorského jazyka.

3.2. Výpočty během životního cyklu

Jádro během svého jednoho životního cyklu provádí různé druhy výpočtu a jeho výpočetní doba je proměnlivá v závislosti na podmínkách a provedených výpočtech. To je také jeden z důvodů, proč jsou životní cykly prováděny v časových intervalech a ne spojitě. Dalším důvodem je vytížení hardwaru zařízení. Následuje výčet nejdůležitějších výpočtů v životním cyklu.

V bodě zájmu - rádius

$$BZ = \{GPS\ sirka, GPS\ delka, radius\}$$

$$Nynější\ stav = \{sirka, delka\}$$

$$Konstanta\ prepoctu\ radiusu\ \doteq 0.000014$$

$$V\ BZ? = |stav.sirka - BZ.sirka| + |stav.delka - BZ.delka| < (BZ.radius * prepocet)$$

V bodě zájmu - Mnohoúhelník

-Využívá Cohen-Sutherlandův algoritmus [7]

$$BZ = \{pole\ GPS\ sirkek, pole\ GPS\ delek, počet\ hran\ n\}$$

Pro všechny body mnohoúhelníku:

$$Pokud\ platí: \{(BZ.delky[i] < stab.delka)$$

$$and(BZ.delky[n-1] \geq stav.delka)$$

$$or(\ BZ.delky[n-1] <\ stav.delka)$$

$$and(BZ.delky[i] \geq stav.delka)\}$$

$$and\{\ (BZ.sirky[i] \leq stav.sirka)$$

$$or(BZ.sirky[n-1] \leq stav.sirka\}$$

$$pak\ v\ BZ? = XOR\left\{\frac{(BZ.sirky[i] + (stav.delka - BZ.delky[i])}{(BZ.delky[n-1] - BZ.delky[i])} * (BZ.sirky[n-1] - BZ.sirky[i]) < stav.sirka\right\}$$

Vzdálenost dvou GPS bodů [8]:

$$GPS_i = \{sirka[rad], delka[rad]\}$$

$$Vzdálenost = RadianyNaStupne[cos^{-1}\{\sin(StupneNaRadiany(GPS_1.sirka) * \sin(StupneNaRadiany(GPS_2.sirka) + \cos(StupneNaRadiany(GPS_1.sirka) * \cos(StupneNaRadiany(GPS_2.sirka) * \cos(StupneNaRadiany(GPS_1.delka - GPS_2.delka)\}] * 111{,}18957696$$

3.3. Jak probíhal vývoj, změny a vylepšení

Vývoj jako takový započal učením se programovat pro operační systém Android. Ačkoli pro něj lze programovat v jazyce Java, Android využívá mnoho vlastních knihoven, ovládání hardwaru, pracování s grafickým rozhraním a další. Proto je součástí celého vývoje učení se.

Samotné jádro je část programu, která se v průběhu vývoje nejvíce měnila. Buďto kvůli novým zkušenostem či zlepšení funkčnosti. Jelikož jádro obsahuje veškeré proměnné a objekty, je základ k fungování aplikace, a bylo tedy vyvíjeno od počátku vývoje. Automatické vlákno jádra však bylo možno vyvíjet až po vytvoření většiny funkcí a možností aplikace.

V průběhu vývoje se měnily přístupy k různým objektům či hardwaru, byly testovány různé modely blokových diagramů a také způsoby programátorských pohledů na jádro. Od druhé generace již má jádro programátorský model neměnný a mění se pouze algoritmus jako takový.

Na začátku vývoje obsluhovalo vlákno jádra krom vytváření nynějšího stavu pouze situace, kdy nebylo jasné, jaká je nynější činnost vozidla či řidiče. V průběhu vývoje bylo samozřejmě nasazeno mnoho vylepšení. Tím je například možnost hlídání bezpečnostních

přestávek, možnost připojit se přes technologii Bluetooth k jednotce vozidla, mapování ujeté cesty či kontrola, zda má řidič dostatek času na plánované činnosti.

3.4. Složitost propojení aplikace a problémy při vývoji

Problémy a překážky během vývoje se dají shrnout do čtyř kategorií: neznalost Android Api, neznalost teorie, převedení teorie v praxi, propojení jádra a aplikace.

Neznalostí Android Api je myšlena situace, kdy programátor potřebuje využívat zdroje či funkce operačního systému, avšak zatím s nimi nepracoval a tedy neví, jak s nimi pracovat nebo zda s nimi lze pracovat podle jeho plánu. Vývojář chce například začít používat Bluetooth technologii zařízení. Musí tedy projít dokumentaci operačního systému a nalézt, zda je to možné. Poté musí prostudovat, jak operační systém k tomuto hardwaru přistupuje a jaké nabízí funkce a možnosti. Dále přiloží danou systémovou knihovnu ke své aplikaci a musí vymyslet a naprogramovat propojení své aplikace a funkcí této knihovny tak, aby bylo zajištěno správné používání systémového zdroje a požadované chování a funkčnost aplikace.

Neznalost teorie je vcelku jednoduchý pojem. V našem případě však znamená závažnou překážku. Pokud chceme vytvořit model chování automatického jádra, musíme znát požadované chování. Toto chování je sice možné vymyslet, nicméně v našem případě se potvrdilo, že si nelze dostatečně představit praktickou situaci. Vývojář si tedy může vymyslet teoretický model například na situaci, kdy řidič z neznámého důvodu stojí na místě. Vytvoří si model požadovaného chování, kdy aplikace zjistí, jak dlouho stojí na místě, zda je motor v běhu, zda stojí na známé místě a další. Tento model rozšíří o požadované chování na různé hodnoty proměnných. Problém je v tom, že přibližně ve 20ti % případů nebude mít tento model v praxi správné chování. Během vývoje bylo vytvořeno mnoho modelů chování tzv. ,,za stolem", ale v praxi nebyly použitelné v dostatečně širokém spektru variant situací. Pokud totiž nejsme v konkrétní praktické situaci, je těžké domyslet různé souvislosti a události, které by mohly situaci či model chování ovlivnit.

Převedení teorie v praxi je naprosto opačná situace než v předchozím odstavci. Již máme vytvořený teoretický model chování z praxe. Tento model je správně reprezentován algoritmem v aplikaci. Teoreticky, pokud se dostaneme v praxi do situace, pro kterou jsme

model vytvořili, měla by se aplikace zachovat správně. V situaci naprosto shodné se situací, při které jsme byli v praxi při vytváření modelu, se správně zachová. Problém nastává ve chvíli, kdy se dostaneme do situace, jejíž datový popis odpovídá konkrétnímu modelu chování, avšak požadovaný model chování v dané situaci je jiný. Situací, do kterých se řidič dostane je mnoho a vytvářet pro každou vzorec chování je nepředstavitelné. Je tedy nutné pro mnoho reálných situací vytvořit obecný model správného chování a přizpůsobit ho tak, aby se aplikace ve specifických situacích chovala správně na základě obecného modelu situace.

Poslední kategorie překážek je čistě programátorský problém. Jelikož je aplikace rozsáhlá, už samotný úkol, aby vše fungovalo jak má, je náročný. My se však zaměříme na problém zasazení automatického jádra do aplikace. Jelikož je aplikace interaktivní a automatické jádro pracuje neustále, lze se dostat do situace, kdy uživatel pracuje s proměnnými či objekty, ke kterým přistupuje v daném okamžiku i jádro. Dále v aplikaci existují menší, samostatná vlákna na jádru nezávislá, která také mohou přistupovat k daným proměnným či objektům. Je tedy nutné zabezpečit veškeré práce automatických či manuálních vrstev tak, aby se navzájem neblokovaly či nevytvářely chyby. Dále je nutné zajistit zápis a čtení ze souborů ve správném pořadí, i když tyto úkony provádějí různé, na sobě nezávislé funkce. Také je nutné si uvědomit, že jádro má odkaz na objekty, které mají odkazy na další objekty, které mohou mít odkaz zpět na proměnné jádra. Každý tento objekt může používat stejnou funkci či hardware. Propojení jádra a samotné aplikace je tedy úkol, který je nutný dělat s rozvahou a přemýšlet, jaké souvislosti a závislosti bude nutné ošetřit. V neposlední řadě je také nutné zajistit, že i přes výskyt chyby nezkolabuje celý operační systém, ani aplikace a nejlépe tak, aby byla pořízená dopravní data stále správná a použitelná.

4. Testování

4.1. Jak probíhalo a probíhá testování

Jsme v situaci, kdy jsme vytvořili nový algoritmus jádra a chceme ho otestovat. Ať už je účel algoritmu jakýkoli, je spuštěn při nějaké situaci, a to buď programové či reálné. Pokud tedy chceme správnost algoritmu otestovat, musíme zkušební situaci buďto nasimulovat či reálně vytvořit. Při testování aplikace bylo použito simulace ve virtuálním prostředí, simulace v praxi a testování v reálném provozu.

Simulaci ve virtuálním prostředí situací lze provést již na PC díky emulátoru zařízení s Androidem. Takovou simulací snadno ověříme správnost výpočtů, vnitřních vazeb aplikace či grafického rozhraní. Máme i možnost simulace příjmu GPS souřadnic, která je však omezená. Pokud simulace proběhne bez problémů, je pravděpodobné, že programátorský kód je napsán správně a můžeme tedy přejít na simulaci v praxi.

Simulace v praxi pro nás znamená, že se zařízením, kde je v chodu naše aplikace, budeme reálně pohybovat ve vozidle a provádět napodobení různých činností či situací. Tím lze z části ověřit správnost modelů chování, správnost rozhodování jádra a další. Během testování v této fázi již zaznamenáváme digitální záznam simulované jízdy a na PC tak můžeme dále ověřit správnost zápisu záznamu. Pokud i tato simulace proběhne bez problémů, lze přejít na testování algoritmu v reálném provozu.

Testování v reálném provozu je pojem označující reálnou jízdu s reálnými činnostmi, bez jakékoli simulace, kdy je nutné domluvit reálnou zakázku s dopravním vozidlem a oprávněnou osobou a provádět normální pracovní činnost. Již se nesnažíme dostat aplikaci do konkrétní situace, ale testujeme aplikaci jako celek a kontrolujeme správné chování aplikace i automatizačního jádra. Při tomto testování můžeme simulovat pouze lidské chyby během interakce s aplikací. Získáme tím maximální množství informací o správnosti algoritmu a funkčnosti celé aplikace. Po ukončení reálného testování již máme plnohodnotný digitální záznam „stazky".

Vývojář aplikace využíval všech tří metod při každé aktualizaci algoritmu jádra. I při malé změně bylo potřeba vyzkoušet správnou funkčnost algoritmu nejprve ve virtuálním prostředí. Vzhledem k rozsáhlosti aplikace bylo běžné, že změna algoritmu v jedné části

aplikace se projevila jako chyba či nesrovnalost v jiné části, kde by to člověk běžně nepředpokládal. Poté autor přešel na simulaci v praxi, kdy měl zapnutou aplikaci v osobním automobilu a simuloval činnosti přepravního vozidla. Tyto simulace mají však omezené možnosti přiblížení se realitě. Příkladem může být nakládka na staveništi, kdy nakladač provádí jednu nakládku s několika mírnými přesuny po různých časových intervalech. Tuto činnost bychom v praxi simulovali konstantní nakládkou na jednom místě bez pohybu, což vůbec neodpovídá reálné situaci. To je důvod, proč se rozhodl vývojář přejít k testování při opravdových zakázkách dopravních firem. Příkladem reálného testování v praxi je zakázka: ,,Převoz dřeva 25.3.2013". Majitel lesních pozemků u obce Buková si objednal u firmy Hobl&Pech s.r.o. práci a převoz dřeva z lesa. V této zakázce se jednalo o poražení a nařezání lesních stromů, jejich naložení na vozidlo, odvezení na předem známý pozemek a následné vyložení. Obrázek 8 a 9 byl pořízen z této zakázky při nakládání rozřezaných stromů.

Obrázek 8

Obrázek 9

4.2. Problémy při optimalizaci, nepraktičnost obecné situace

V minulé kapitole jsme naznačili problém obecné situace. Řidič a vozidlo se mohou nacházet v nepřeberném množství variant různých situací a vytvořit model chování pro každou takovou variantu je téměř nepředstavitelné. Je tedy nutné tyto varianty rozdělit do několika skupin obecných situací. Což může být kontraproduktivní, když si uvědomíme, kolik proměnných hraje v takové obecné situaci svou roli.

Zaobírejme se pro ilustraci obecnou situací, kdy vozidlo zastavilo a stojí na neznámém místě. Máme již definovaný model chování, kdy zkoumáme, jak dlouho vozidlo není v pohybu, zda je zapnut motor a další. Dá se tedy předpokládat, že pokud vozidlo není v pohybu dlouhou dobu a nemá motor v chodu, je pravděpodobné, že řidič či vozidlo provádí nějakou činnost. Co když je ale řidič v situaci, kdy zastavil na železničním přejezdu, vypnul motor a čeká na přejezd vlaku? Taková situace přesně sedí do našeho modelu, ale přesto je v praxi brána jako součást jízdy (myšleno z pohledu souvisejících legislativních norem např. Vyhláška 561/2006 sb. [10]). Je zřejmé, že pokud se dotážeme řidiče, v jaké situaci se nachází, tento problém obejdeme. Co když ale řidič nereaguje? Pokud automatické jádro bude pouze čekat na interakci řidiče, může přijít o zásadní údaj, naopak pokud zaznamená činnost i přes to, že řidič žádnou neprovádí, dopravní data také nebudou správná.

Z těchto důvodů trvala optimalizace veškerých konstant a pravidel chování více než půl roku a stále se objevují situace, kdy chování není plně správné. Bylo nutné umožnit jádru samostatné chování a vytváření záznamu, a přesto umožnit, aby řidič mohl dopravní data kdykoli upřesnit či upravit.

4.3. Digitální záznam

Čistý binární záznam aplikace ve formě jedniček a nul není moc použitelný, proto bylo potřeba vytvořit digitální metajazyk pro záznam dat a jejich přenesení na PC. Ten byl vytvořen již při vývoji aplikace jako XML [9], což je obecný značkovací jazyk. Byly tedy definovány vlastní značky, takzvané ,,tagy", které ohraničují informaci pevně spojenou s danou značkou. Tímto způsobem můžeme zaznamenat například název nějakého místa ve formě XML:

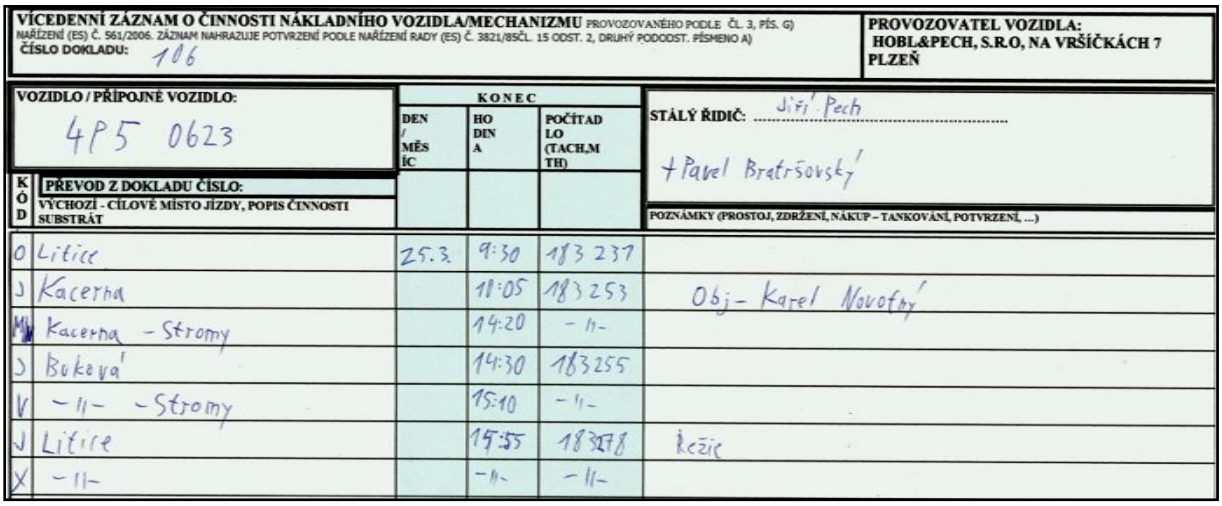
<NAZEV>Praha</NAZEV>

Záznam dopravních dat je tedy aplikací vytvořený textový soubor s příponou „.xml", který obsahuje námi nadefinované značky, které obsahují konkrétní dopravní informace. Tento soubor je následně přenesen do PC, kde může být dále zpracován.

4.4. Porovnání ručně psaného a digitálního záznamu

Nyní porovnáme reálné záznamy z již zmíněné provedené zakázky ,,Převoz dřeva 25.3.2013".

Ručně psaný záznam:



VÍCEDENNÍ ZÁZNAM O ČINNOSTI NÁKLADNÍHO VOZIDLA/MECHANIZMU PROVOZOVANÉHO PODLE ČL. 3, PÍS. G) NAŘÍZENÍ (ES) Č. 561/2006. ZÁZNAM NAHRAZUJE POTVRZENÍ PODLE NAŘÍZENÍ RADY (ES) Č. 3821/85ČL. 15 ODST. 2, DRUHÝ PODODST. PÍSMENO A)

ČÍSLO DOKLADU: 106

PROVOZOVATEL VOZIDLA: HOBL&PECH, S.R.O, NA VRŠÍČKÁCH 7 PLZEŇ

VOZIDLO / PŘÍPOJNÉ VOZIDLO: 4P5 0623

STÁLÝ ŘIDIČ: Jiří Pech, † Pavel Bratršovský

KÓD	PŘEVOD Z DOKLADU ČÍSLO: VÝCHOZÍ - CÍLOVÉ MÍSTO JÍZDY, POPIS ČINNOSTI SUBSTRÁT	KONEC DEN / MĚSÍC	KONEC HODINA	KONEC POČÍTADLO (TACH,MTH)	POZNÁMKY (PROSTOJ, ZDRŽENÍ, NÁKUP – TANKOVÁNÍ, POTVRZENÍ, ...)
O	Litice	25.3	9:30	183 237	
J	Kacerna		11:05	183 253	Obj - Karel Novotný
M	Kacerna - Stromy		14:20	- ||-	
J	Bukeva		14:30	183255	
V	- ||- - Stromy		15:10	- ||-	
J	Litice		15:55	183278	Řežic
X	- ||-		- ||-	- ||-	

Obrázek 10

Digitální záznam v jazyce XML:

```
<STAZKA>
 <STH>
  <SPZ>4P50623</SPZ>
  <DATUMPOCATKU>25.03.2013</DATUMPOCATKU>
  <CASPOCATKU>09:28:12</CASPOCATKU>
  <DATUMKONCE>26.04.2013</DATUMKONCE>
  <CASKONCE>15:57:53</CASKONCE>
  <CLENOVEOSADKY>
   <SOS>
    <OSC>1</OSC>
    <JMENO>Pech Jiří</JMENO>
   </SOS>
   <SOS>
    <OSC>9</OSC>
    <JMENO>Bratršovský pavel</JMENO>
   </SOS>
  </CLENOVEOSADKY>
  <UKAZATELE>
   <SSU>
    <DRUH>1</DRUH>
    <POCATECNISTAV>183231.7</POCATECNISTAV>
    <KONECNYSTAV>183278.1</KONECNYSTAV>
   </SSU>
   <SSU>
    <DRUH>2</DRUH>
    <POCATECNISTAV>57.2</POCATECNISTAV>
    <KONECNYSTAV>51.9</KONECNYSTAV>
   </SSU>
  </UKAZATELE>
  <OBJEDNATELE>
   <FEH>
    <PORADI>1</PORADI>
    <OBJEDNATEL>
     <OBJEDMASKA>6547</OBJEDMASKA>
     <OBJEDMASO>968</OBJEDMASO>
    </OBJEDNATEL>
    <IDRELACE>8735469</IDRELACE>
   </FEH>
  </OBJEDNATELE>
 </STH>
 <RADKY>
  <STR>
   <DRUHKOD>1</DRUHKOD>
   <MISTO>Litice</MISTO>
   <DATUMPOCATKU>25.03.2013</DATUMPOCATKU>
   <CASPOCATKU>09:28:12</CASPOCATKU>
```

```
 <DATUMKONCE>25.03.2013</DATUMKONCE>
 <CASKONCE>09:31:54</CASKONCE>
 <VZD>0.0</VZD>
 <DVY>3</DVY>
 <CLENOVEOSADKY>
  <SOS>
   <OSC>1</OSC>
   <JMENO>Pech Jiří</JMENO>
  </SOS>
 </CLENOVEOSADKY>
 <GPSZACATEK>49.69853;13.359692</GPSZACATEK>
 <GPSKONEC>49.69853;13.359692</GPSKONEC>
</STR>
<STR>
 <DRUHKOD>2</DRUHKOD>
 <MISTO>Kacerna</MISTO>
 <DATUMPOCATKU>25.03.2013</DATUMPOCATKU>
 <CASPOCATKU>09:31:54</CASPOCATKU>
 <DATUMKONCE>25.03.2013</DATUMKONCE>
 <CASKONCE>10:04:08</CASKONCE>
 <VZD>21.5</VZD>
 <DVY>32</DVY>
 <CLENOVEOSADKY>
  <SOS>
   <OSC>1</OSC>
   <JMENO>Pech Jiří</JMENO>
  </SOS>
  <SOS>
   <OSC>9</OSC>
   <JMENO>Bratršovský pavel</JMENO>
  </SOS>
 </CLENOVEOSADKY>
 <OBJEDNATELE>
  <FEH>
   <PORADI>1</PORADI>
   <OBJEDNATEL>
    <OBJEDMASKA>6547</OBJEDMASKA>
    <OBJEDMASO>968</OBJEDMASO>
   </OBJEDNATEL>
   <IDRELACE>8735469</IDRELACE>
  </FEH>
 </OBJEDNATELE>
 <GPSZACATEK>49.69853;13.359692</GPSZACATEK>
 <GPSKONEC>49.355625;15.4886298</GPSKONEC>
</STR>
<STR>
 <DRUHKOD>3</DRUHKOD>
 <MISTO>Kacerna</MISTO>
 <DATUMPOCATKU>25.03.2013</DATUMPOCATKU>
 <CASPOCATKU>10:04:08</CASPOCATKU>
 <DATUMKONCE>25.03.2013</DATUMKONCE>
 <CASKONCE>14:23:57</CASKONCE>
 <VZD>0.0</VZD>
 <DVY>259</DVY>
 <SUBSTRATY>
  <STS>
   <NAZEV>Dřevo</NAZEV>
   <PJD>1</PJD>
   <JED>TUNA</JED>
  </STS>
 </SUBSTRATY>
 <CLENOVEOSADKY>
  <SOS>
   <OSC>1</OSC>
   <JMENO>Pech Jiří</JMENO>
  </SOS>
  <SOS>
   <OSC>9</OSC>
   <JMENO>Bratršovský pavel</JMENO>
  </SOS>
 </CLENOVEOSADKY>
 <OBJEDNATELE>
  <FEH>
   <PORADI>1</PORADI>
   <OBJEDNATEL>
    <OBJEDMASKA>6547</OBJEDMASKA>
    <OBJEDMASO>968</OBJEDMASO>
   </OBJEDNATEL>
   <IDRELACE>8735469</IDRELACE>
  </FEH>
 </OBJEDNATELE>
 <GPSZACATEK>49.355625;15.488629</GPSZACATEK>
 <GPSKONEC>49.355985;15.486851</GPSKONEC>
</STR>
<STR>
 <DRUHKOD>2</DRUHKOD>
 <MISTO>Buková</MISTO>
 <DATUMPOCATKU>25.03.2013</DATUMPOCATKU>
 <CASPOCATKU>14:23:57</CASPOCATKU>
 <DATUMKONCE>25.03.2013</DATUMKONCE>
 <CASKONCE>14:29:45</CASKONCE>
 <VZD>2.3</VZD>
 <DVY>5</DVY>
 <SUBSTRATY>
  <STS>
   <NAZEV>Dřevo</NAZEV>
   <PJD>1</PJD>
   <JED>TUNA</JED>
  </STS>
 </SUBSTRATY>
 <CLENOVEOSADKY>
  <SOS>
   <OSC>1</OSC>
   <JMENO>Pech Jiří</JMENO>
  </SOS>
  <SOS>
   <OSC>9</OSC>
   <JMENO>Bratršovský pavel</JMENO>
  </SOS>
 </CLENOVEOSADKY>
```

```
<OBJEDNATELE>
 <FEH>
  <PORADI>1</PORADI>
  <OBJEDNATEL>
   <OBJEDMASKA>6547</OBJEDMASKA>
   <OBJEDMASO>968</OBJEDMASO>
  </OBJEDNATEL>
  <IDRELACE>8735469</IDRELACE>
 </FEH>
</OBJEDNATELE>
<GPSZACATEK>49.355985;15.486851</GPSZACATEK>
<GPSKONEC>49.60394;13.33973</GPSKONEC>
</STR>
<STR>
 <DRUHKOD>4</DRUHKOD>
 <MISTO>Buková</MISTO>
 <DATUMPOCATKU>25.03.2013</DATUMPOCATKU>
 <CASPOCATKU>14:29:45</CASPOCATKU>
 <DATUMKONCE>25.03.2013</DATUMKONCE>
 <CASKONCE>15:10:32</CASKONCE>
 <VZD>0.0</VZD>
 <DVY>41</DVY>
 <SUBSTRATY>
  <STS>
   <NAZEV>Dřevo</NAZEV>
   <PJD>1</PJD>
   <JED>TUNA</JED>
  </STS>
 </SUBSTRATY>
 <CLENOVEOSADKY>
  <SOS>
   <OSC>1</OSC>
   <JMENO>Pech Jiří</JMENO>
  </SOS>
  <SOS>
   <OSC>9</OSC>
   <JMENO>Bratršovský pavel</JMENO>
  </SOS>
 </CLENOVEOSADKY>
 <OBJEDNATELE>
  <FEH>
   <PORADI>1</PORADI>
   <OBJEDNATEL>
    <OBJEDMASKA>6547</OBJEDMASKA>
    <OBJEDMASO>968</OBJEDMASO>
   </OBJEDNATEL>
   <IDRELACE>8735469</IDRELACE>
  </FEH>
 </OBJEDNATELE>
 <GPSZACATEK>49.60394;13.33973</GPSZACATEK>
 <GPSKONEC>49.60394;13.33973</GPSKONEC>
</STR>
<STR>
 <DRUHKOD>2</DRUHKOD>
 <MISTO>Litice</MISTO>
 <DATUMPOCATKU>25.03.2013</DATUMPOCATKU>
 <CASPOCATKU>15:10:32</CASPOCATKU>
 <DATUMKONCE>25.03.2013</DATUMKONCE>
 <CASKONCE>15:57:53</CASKONCE>
 <VZD>23.4</VZD>
 <DVY>47</DVY>
 <CLENOVEOSADKY>
  <SOS>
   <OSC>1</OSC>
   <JMENO>Pech Jiří</JMENO>
  </SOS>
 </CLENOVEOSADKY>
 <GPSZACATEK>49.60394;13.33973</GPSZACATEK>
 <GPSKONEC>0.0;0.0</GPSKONEC>
</STR>
<STR>
 <DRUHKOD>13</DRUHKOD>
 <MISTO>Litice</MISTO>
 <DATUMPOCATKU>25.03.2013</DATUMPOCATKU>
 <CASPOCATKU>15:57:53</CASPOCATKU>
 <DATUMKONCE>25.03.2013</DATUMKONCE>
 <CASKONCE>15:57:53</CASKONCE>
 <VZD>0.0</VZD>
 <DVY>0</DVY>
 <CLENOVEOSADKY>
  <SOS>
   <OSC>1</OSC>
   <JMENO>Pech Jiří</JMENO>
  </SOS>
 </CLENOVEOSADKY>
 <GPSZACATEK>49.69853;13.359692</GPSZACATEK>
 <GPSKONEC>49.69853;13.359692</GPSKONEC>
</STR>
</RADKY>
</STAZKA>
```

Porovnání záznamů:

Na první pohled je viditelné, že ručně psaný záznam je pro člověka mnohem lépe čitelný. Na druhý pohled však zjistíme, že digitální záznam obsahuje mnohem více informací a po strojovém zpracování na PC nám poskytne mnohem lepší záznam dopravních dat.

Z ručně psaného záznamu víme jednoduché informace, avšak z digitálního záznamu víme komplexní informace o celé „stazce". Z hlavičky „stazky" víme informace o vozidle, řidiči, ukazatelích vozidla, posádce vozidla a objednatelů v rámci celé ,,stazky". U každé činnosti je pak zaznamenána informace o typu činnosti, místu, intervalu provádění, ujeté vzdálenosti, délce činnosti v minutách, členech posádky a objednatelích během činnosti, substrátu a GPS souřadnicích při začátku a konci činnosti. Při správném zpracování těchto informací tak dispečer dostane přesné informace, co se kdy s vozidlem dělo.

5. Závěr

V nynější fázi je třetí generace elektronické „stazky" plně funkční a stále se testuje. Naším cílem je vyladit aplikaci do maximální možné dokonalosti, bez výskytu chyb a s nejlepším možným celkovým chováním a funkčností. Dále je aplikace ve fázi, kdy se začíná s ostrým nasazením do provozu, a to u několika reálných dopravních firem, které jsou zákazníky firmy Hobl&Pech. I za optimistického předpokladu, že by nebylo potřeba upravovat aplikaci na základě zkušeností s reálným nasazením, počítá se s dalším vývojem a několika dalšími generacemi aplikace. Do budoucna je v plánu propojit aplikaci s GPS navigací a pomocí GPRS technologie umožnit komunikaci s dispečerem v reálném čase. Ideou je tedy vytvořit jednotné řešení pro dopravní firmy, pokrývající veškeré možnosti a požadavky v oblasti sledování činností vozidel, pomáhání řidičům a vzájemném propojení s dispečinkem.

6. Literatura a reference

Během celé práce bylo čerpáno ze zkušeností z praxe, internetu a lidských nápadů či myšlenek. Internetové zdroje sloužily především k čerpání technologických či programátorských informací, následuje výčet nejdůležitějších referencí.

[Hobl&Pech s.r.o.] - http://www.hoblapech.cz/

[Android developers]: http://developer.android.com/index.html

[1] GPS: http://cs.wikipedia.org/wiki/GPS

[2] Windows CE: http://cs.wikipedia.org/wiki/Windows_CE

[3] Windows Mobile: http://cs.wikipedia.org/wiki/Windows_Mobile

[4] iOS: http://cs.wikipedia.org/wiki/IOS_(Apple)

[5] Android: http://cs.wikipedia.org/wiki/Android_(operační_systém)

[6] Java: http://cs.wikipedia.org/wiki/Java_(programovací_jazyk)

[7] Cohen-Sutherlandův algoritmus: http://en.wikipedia.org/wiki/Cohe Sutherland_algorithm

[8] Vzdálenost dvou GPS bodů: http://cs.wikipedia.org/wiki/Elipsoid#Vzd.C3.A1lenost_dvou_bod.C5.AF_na_povrchu_elipsoidu

[9] XML: http://cs.wikipedia.org/wiki/Xml

[10] Vyhláška 561/2006 sb.: http://www.mpsv.cz/ppropo.php?ID=nv589_2006