VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

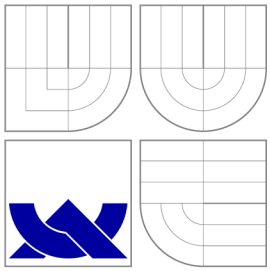
# ŘÍDÍCI APLIKACE PRO KAMEROVÝ DOHLEDOVÝ SYSTÉM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

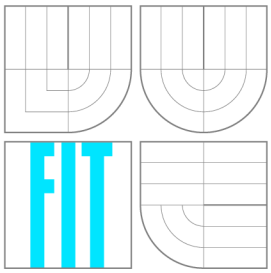
AUTOR PRÁCE
AUTHOR

TOMÁŠ MATULA

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

# ŘÍDÍCI APLIKACE PRO KAMEROVÝ DOHLEDOVÝ SYSTÉM

CONTROL APPLICATION OF A CAMERA SURVEILLANCE SYSTEM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — TOMÁŠ MATULA
AUTHOR

VEDOUCÍ PRÁCE — Ing. FILIP ORSÁG, Ph.D.
SUPERVISOR

BRNO 2011

# Abstrakt

Práce se zabývá návrhem a implementací řídící aplikace pro sledovací kamerový systém. Velká část práce je věnována tvorbě uživatelského rozhraní s využitím knihovny Qt a návrhu struktury modulární aplikace. Výsledný produkt je otestován a zhodnocen.



# Abstract

This work describes design and implementation of control application of a camera surveillance system. Major part of this work is dedicated to GUI creation using Qt library and modular application design. The final product is tested and evaluated.



# Klíčová slova

Qt, grafické uživatelské rozhraní, GUI, dohledový systém, plugin systém, dynamicky linkovaná knihovna, dll

# Keywords

Qt, graphical user interface, GUI, surveillance system, plugin system , dynamic linking library, dll


# Citace

Tomáš Matula: Řídíci aplikace pro kamerový dohledový systém, bakalářská práce, Brno, FIT VUT v Brně, 2011

# Řídíci aplikace pro kamerový dohledový systém

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Filipa Orsága, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Tomáš Matula
16. mája 2011

## Poděkování

Děkuji svému vedoucímu Ing. Filipovi Orságovi, Ph.D. za odbornou pomoc při tvorbě této práce. Chtěl bych poděkovat také kamarádovi Borisovi Švancarovi za poskytnutí materiálů, své rodině a ostatním, kteří mě podporovali.

© Tomáš Matula, 2011.
*Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.*

# Obsah

# Kapitola 1

# Úvod

Človek sa snaží, aby mohol prácu vykonávať ľahšie, jednoduchšie a rýchlejšie už od počiatku vekov. Z toho dôvodu sa ľudské poznanie rozvýjalo hlavne v odboroch vedy a techniky. Boli vynájdené tisícky vynálezov a v 20. storočí sa medzi ne zaradil aj počítač. Na začiatku bol využívaný iba v určitých odvetviach, ale postupne sa vyvýjal a dostával do ďalších a ďalších. Dnes je zastúpený skoro v každom bežne dostupnom zariadení (práčka, kávovar, mobilný telefón, auto...). Každé takéto zariadenie by malo mať vlastné užívateľské rozhranie na komunikáciu človeka s počítačom, inak by ho bolo zložité obsluhovať.

V tejto práci sa budeme venovať vytvoreniu riadiacej aplikácie pre sledovací kamerový systém. Podobné aplikácie nie sú na trhu veľmi rozšírené, preto sa bude jednať o ojedinelý produkt. Zameriame sa na tvorbu užívateľského rozhrania. Rozhranie môže byť samotný príkazový riadok, textové, grafické, hlasové a pod. My budeme vytvárať grafické užívateľské rozhranie tiež GUI (angl. Graphical User Interface), pretože je vhodné a práca s ním je jednoduchšia. Kladie sa dôraz na to aby bola aplikácia modulárna, čo znamená, že umožní pridávanie samostatných modulov, ktoré budú rozširovať funkčnosť aplikácie. Táto vlastnosť zabezpečí, že môžu byť vytvárané oddelene bez nutnosti zoznámenia sa s celou aplikáciou. Ďalej ponúka možnosť náhrady zastaralého modulu za novší, a tým predĺženie jej životnosti. Cieľom je tak analýza, návrh a implementácia základného riešenia danej aplikácie, ktorá možno nájde svoje využite v praxi.

Postupne sa v práci zameriam na základné požiadavky sledovacieho systému, jeho popis a porovnanie s existujúcimi aplikáciami podobného druhu. V ďalšej časti určím teoretické predpoklady potrebné na vytvorenie aplikácie. Nasledujúce kapitoly sa budú venovať návrhu spomenutého užívateľského rozhrania a následnej implementácií celého programu. Na koniec vykonám na aplikácii potrebné testy a zhodnotím funkčnosť celého systému.

# Kapitola 2

# Sledovacie kamerové systémy

V posledných rokov začal narastať záujem o sledovacie kamerové systémy. Dochádza k tomu hlavne kvôli potrebe neustále zvyšovať bezpečnosť (banky, sklady, domácnosti,...), efektivitu práce (monitorovanie prevádzok) atď. Kamerové systémy, od ktorých sa požaduje vyhodnocovanie udalostí (napr. nečakaný pohyb) a následná odpoveď v reálnom čase, sa musia vysporiadať s veľkým problémom. Tým je neustále sledovanie. Vo väčšne prípadov sa to vykonáva pomocou ľudského operátora, ktorý musí celý čas dávať pozor. Takáto prevádzka je značne neefektívna, pretože aj dobre trénovaní a skúsení ľudia majú problém udržať pozornosť na dlhšie časové obdobie a zamestnávať ľudí na sledovanie videa je pomerne drahé. Preto sa dostáva do popredia vývoj inteligentných poloautomatických a automatických systémov, ktoré zabezpečujú efektívne rozoznávanie a sledovanie rôznych objektov, ako napríklad vozidiel, osôb, atď. alebo detekovanie pohybu a upozornenie operátora alebo priamo vykonanie reakcie (napr. spustenie alarmu). Takéto systémy sú riadené aplikáciou, ktorá zabezpečuje logickú funkčnosť celého systému a jeho komunikáciu s užívateľom. Práve návrh a vytvorenie riadiacej aplikácie je predmetom tejto práce. Informácie v tejto kapitole som čerpal z práce[6].

## 2.1 Vlastnosti sledovacieho systému

Sledovacie systémy by mali byť stavané na prevádzku 24 hodín denne 7 dní v týždni. Pri neustálej prevádzke je vysoká pravdepodobnosť výskytu chyby, preto by mal byť spoľahlivý, bez nečakaného správania a s minimálnymi zdržaniami. Je potrebné, aby sa dokázal prispôsobovať svetelným podmienkam, či už automaticky alebo pomocou zásahu užívateľa nastavením potrebných parametrov. Nemenej dôležité je spracovanie a zobrazenie obrazu v reálnom čase. Trhaný alebo oneskorený obraz môžu v systémoch zameraných na určenie polohy v danom okamžiku skresliť a znehodnotiť výsledky. Toto boli vlastnosti, ktoré musí mať kamerový systém určený na nepretržité sledovanie.

## 2.2 Základné požiadavky na aplikáciu

Požiadavky na sledovací systém sa môžu líšiť v závislosti od uhla pohľadu a oblasti, kde sa bude používať. Keďže vytvárame aplikáciu a jej užívateľské rozhranie nebudeme sa zaoberať požiadavkami na celý systém, obzvlášť na hardvér. My sa pozrieme na najbežnejšie nároky užívateľov na takéto aplikácie.

Vlastnosťou dobrej aplikácie (nielen sledovacieho systému) je užívateľská prívetivosť. Program by malo byť intuitívne, rýchlo a jednoducho ovládateľný. Práca s ním by mala byť po krátkom zaškolení veľmi efektívna.

Sledovacie systémy sa väčšinou využívajú na kontrolu bezpečnosti, preto je potrebné zaistiť aj bezpečnosť aplikácie pred neželaným vstupom neoprávnených osôb. To znamená, že musí obsahovať funkčnosť na prihlasovanie, odhlasovnie a zároveň aj pridávanie nových užívateľov. Ale nie každý musí mať rovnaké práva na prácu so systémom, preto by malo byť zabezpečné nastavenie a rozdelenie práv na jednotlivé činnosti, minimálne treba rozlišovať 2 typy (administrátor a klasický užívateľ). Je dosť pravdepodobné, že do kontaktu s aplikácou sa dostane viac ľudí (striedanie zamestnancov) a každý má iný spôsob používania, tak by mala umožnovať uloženie a samozrejme aj načítanie nastavnení.

Keďže sa jedná o kamerový systém, jedným s najdôležitejších požiadavkov je zobrazenie výstupu kamier. Užívateľ by mal vedieť na obraze rozoznať aspoň základné typy objektov, preto by sa snímky z kamery mali zobrazovať na dostatočne veľkej ploche a tým tvoriť hlavnú časť vizuálnej stránky aplikácie. Pri viackamerových systémoch musí obsahovať výber, z ktorej kamery budeme sledovať obraz, poprípade aj pridávanie nových kamier do systému. Ďalej pri systémoch s pohyblivými kamerami je manipulácia s nimi tiež sučasťou aplikácie. Zároveň by mala obsahovať informačný panel, kde bude užívateľovi oznamovať, čo program v danej chvíli vykonáva a upozorňovať ho na prípadné chyby a získané údaje (počet, poloha objektov, ai.). Pre potreby dodatočného preskúmania sledovaného miesta, by mal program umožňovať ukladanie videozáznamov z kamery. Ich manuálne sledovanie je, ako už bolo povedané, dosť náročné, tak je tiež požadované prehrávanie záznamov a tým samozrejme využívanie možností a funkcií aplikácie.

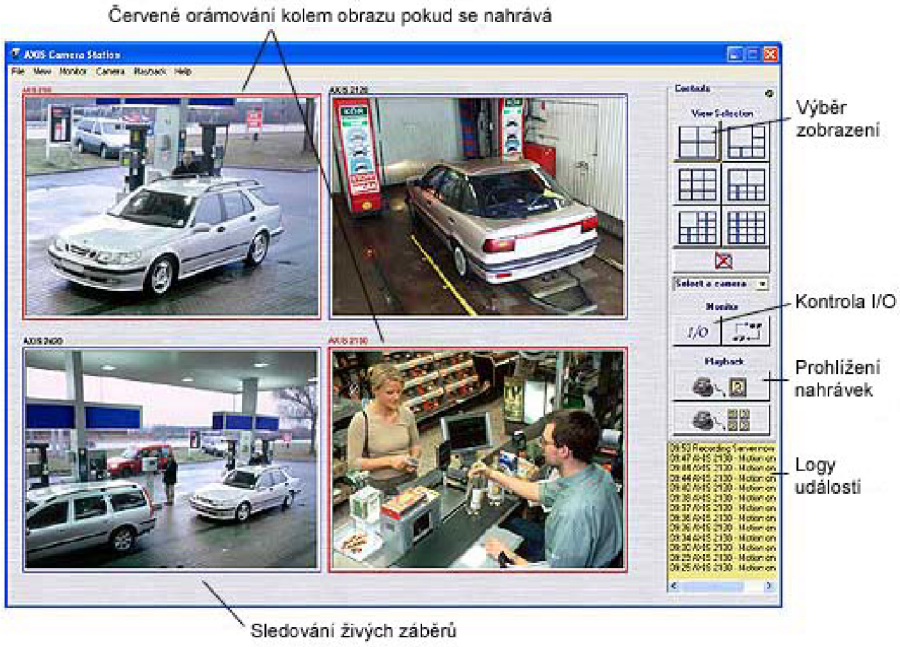
Aplikácia inteligentného sledovacieho systému musí ďalej sprostredkovať vykonávanie množstva rôznych činností. Medzi základné sa radí rozoznávanie objektov v obraze, detekovanie pohybu alebo sledovanie objektov (angl. tracking). S týmito funkciami sú spojené ďalšie činnosti ako odstránenie šumu alebo použitie filtrov pre rôzne prostredia a intenzity osvetlenia. Tieto funkcie by mali byť do programu pridávané osobitne ako moduly, aby bolo zaručené ich jednoduché vylepšenie a prípadné odstránenie. Ich tvorba však nie je náplňou tejto práce.

## 2.3 Aplikácie na trhu

Na trhu sa nachádza niekoľko spoločností, ktoré sa zaoberajú výrobou, inštaláciou a podporou kamerových systémov. Každá z nich ponúka aj softvér, ktorý je kompatibilný s nimi vyrábaným alebo dodávaným systémom (napr. kamerami). Dané aplikácie sa rozdeľujú do dvoch skupín. Do prvej skupiny patria aplikácie pre správu a zobrazenie obrazu z kamier a do druhej pre analýzu obrazu, ako je napríklad vyhľadávanie a sledovanie objektov. Niektoré funkcie sa medzi programami z jednotlivých skupín prelínajú, hlavne tie z druhej menovanej skupiny obsahujú alebo využívajú veľké množstvo fukncií, ktoré sú typické pre správu obrazu a videa.

### 2.3.1 Aplikácie pre správu kamerových systémov

Teraz si predstavíme niekoľko aplikácií, ktoré radíme do prvej skupiny, čiže medzi aplikácie pre správu a zobrazenie videa. Jednoduchý a vhodný pre použitie s malými až stredne veľkými kamerovými systémami je softvér *Axis Camera Station*[3], ktorý pracuje so sieťovými kamerami a video servermi. Dokáže náhrávať video vo vysokej kvalite a zároveň umožňuje

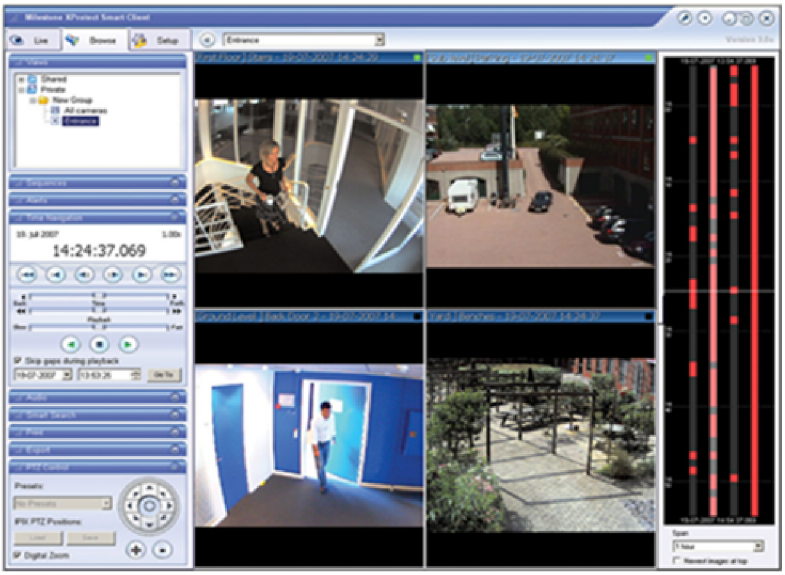



Obrázok 2.1: Užívateľské rozhranie ACS[3]

sledovanie reálneho výstupu z kamery. Ponúka tiež zobrazenie viacerých udalosti z kamier súčasne a tým poskytne užívateľovi úplný prehľad o dianí v sledovanej oblasti. Videozáznam môže byť spustený externým snímačom alebo na základe udalosti detekovanej pohybom. Pomocou *Axis Camera Station* klienta alebo cez internetový prehliadač je dostuné vzdialené sledovanie a administrácia v prípade, keď sa nemôžme dostaviť ku hlavnému počítaču. Ukážka užívateľského rozhrania aplikácie je na obrázku 2.1.

Ďalšia predstavená aplikácia je od spoločnosti *Milestone Systems*[2], ktorá poskytuje jedny z najlepších aplikácií na správu kamerových systémov založených na sieťovýh kamerách. *XProtect Corporate* je najvyspelejšia platforma pre správu sieťového videa. Je určená pre veľké organizácie. Medzi základné funkcie sa radí neobmedzený počet kamier, užívateľov a serverov. Z dôvodu využívania veľkým množstvom osôb je zabezpečená kontrola prístupových práv. Všetky nastavenia zariadení a užívateľov sú vykonávané pomocou hlavnej konzoly pre správu, takže je centralizovaná. Dokáže naplánovať spustenie akcií na vopred určený čas. Umožňuje ovládanie otáčania a zoomu kamier. Disponuje funkciou nahrávania videa a ukladania snímok. Ďalej pomocou klientského modulu[2.2] umožní vzdialený prístup, ktorý je charakterizovaný slovami užívateľská prívetivosť, ovládanie a integrácia. K aplikácií je možné doinštalovať niekoľko voliteľných prídavných modulov (napr. analýza poznávacích značiek automobilov), ktoré rozširujú jej funkčnosť a možnosti, a tým sa právom radí medzi špičku na trhu.

Balík *BOSCH Video-over-IP Lite Suite*[4] poskytuje intuitívne ovládateľnú pracovnú stanicu a ďalšie potrebné aplikácie a prvky prostredia (*Viewer, Archive Player, Configuration Manager*). Aplikáciu je možné používať samostatne alebo tiež ako klienta serveru *BOSCH VIDOS*. Je vhodná skôr pre menšie systémy, z dôvodu podpory obmedzeného počtu kamier. Užívateľ tohoto systému môže nahrávať, prehľadávať a prehrávať záznamy a zhotovovať fotografie, sledovať živý obraz súčasne z viacerých kamier a kompletne ovládať kamery. Na obrázku 2.3 je ukážka pracovanej stanice.




Obrázok 2.2: Milestone Xprotect Smart Client[2]

### 2.3.2 Aplikácie na analýzu obrazu

Softvér, ktorý vykonáva analýzu obrazu a videozáznamu je dôležitý hlavne pri inteligentných sledovacích kamerových systémoch. Po zhliadnutí a vyskúšaní niektorých aplikácií[3] sa vo väčšine vykonávaných funkciách zhodujú. Líšia sa najmä v kvalite spracovania a množstvom automaticky vykonávaných činností. Za zmienku stojí softvér *BOSCH IVA 4.0*, ktorý zabezpečuje kompletnú analýzu obrazu. Dokáže detekovať pohyb, dlhšiu dobu nečinné objekty alebo prekročenie vopred stanovených hraníc, pričom eliminuje rušivé elementy (napr. pohyb stromov vo vetre). Zobrazuuje trajektórie, smer a rýchlosť pohybu. Prispôsobovanie sa rôznym svetelným podmienkam mu tiež nerobí problém, kde automaticky zvolí vhodné nastavenie kamery. Podporuje zobrazenie a spracovanie obrazu získaného z termálnych kamier. Pri tom všetkom je relatívne jednoduchý na ovládanie. Vďaka tomuto všetkému sa radí medzi najlepšie produkty v danej oblasti.

Zaujímavý je softvér *Cognimatics TrueView*, ktorý má niekoľko podproduktov. Jednotlivé podprodukty sú určené na rôzne činnosti (počítanie osôb, vozidiel, zisťovanie dĺžky radov pri pokladniach). Tento produkt je určený predovšetkým na štatistický zber informácií z kamerových záberov a ich následné využitie (napr. indikovanie počtu voľných parkovacích miest v závisloti od obojstranne prejdených vozidiel). Použitím týchto produktov získame niečo ako senzor, ktorý spoľahlivo a v reálnom čase detekuje, rozpoznáva, počíta a sleduje objekty.

Posledným spomenutým softvérom na analýzu obrazu bude *AgentVi Vi-System*. Detekuje množstvo definovaných udalostí a objektov. Umožňuje k týmto udalostiam priradiť spustenie určeného druhu poplachu. Automaticky zistí prekračovanie hraníc, chýbajúce objekty, zhomažďovanie osôb a potulovanie sa v sledovanej oblasti. Dokáže rozlišovať osoby, vozidlá a spočítať ich výskyt v zábere, alebo za daný čas. Obsahuje podporu pre všetky typy prostedia v rôznych poveternostných a svetlných podmienkach. Pre veľký počet funkcií nachádza svoje použitie v rôznych oblastiach, ako napríklad pri zabezpečení pozemkov, sledovaní a riadení dopravy alebo v marketingovej analýze.

Podľa vyššie uvedených informácií musí vedieť softvér pre kamerové sledovacie systémy


Obrázok 2.3: Pracovná stanica BOSCH Video-over-IP[4]

vykonávať rôzne činnosti a funkcie. Všetky tieto funkcie bude zastrešovať riadiaca aplikácia, ktorá by mala umožniť ich jednoduché pridávanie a odoberanie. Samozrejme užívateľsky prívetivú manipuláciu s nimi a s celou aplikáciou.

# Kapitola 3

# Teoretické základy

Aplikácia bude vytvorená v jazyku C++ s využitím knižnice Qt, preto je potrebné zoznámiť sa s ňou. Ďalej musíme poznať pojem modularita a ako ju dosiahnuť. Z toho dôvodu bude táto kapitola rozdelená na dve časti. Prvá sa bude venovať knižnici Qt a popisu niektorých prvkov, ktoré poskytuje. Pri získavaní potrebných informácií som vychádzal z [1], [7], [5]. Druhá v skratke vysvetlí, čo je to modularita a ďalšie s ňou spojené pojmy.

## 3.1 Knižnica Qt

Qt je mulitiplatformná knižnica vytvorená spoločnosťou *Trolltech*, ktorej vlastníkom je od roku 2008 firma *Nokia*. V roku 1995 bola na trh uvedená jej prvá verzia. Odvtedy prešla značným vývojom a momentálne je jej aktuálna verzia 4.7.1 (od 9. Novembra 2010). Qt má podporu pre jazyk C++, ale existujú verzie aj pre Python, Perl, atď. Je vhodná na tvorbu programov s grafický užívateľským rozhraním. Obsahuje tiež nástroje na spracovanie XML súborov, mutimédií, prácu v počítačovej sieti a množstvo ďalších. V súčasnej dobe je pomocou nej implementovaných veľa programov ako napríklad internetový prehliadač Opera, VoIP komunikátor Skype, VirtualBox a iné. Je na nej postavené aj unixové prostredie KDE.

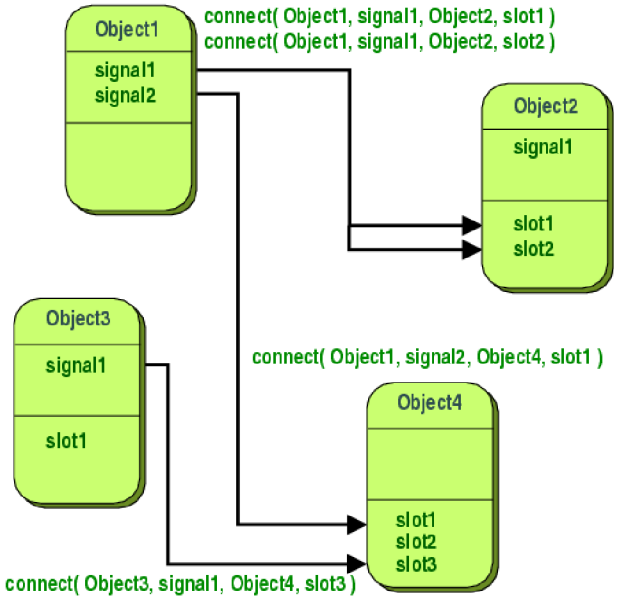
### 3.1.1 Licencie

Knižnica Qt má dostupné 3 druhy licencií. Pre voľne šíriteľné programy sú to Qt GNU LGPL v 2.1 a Qt GNU GPL v 3.0. Pre komerčné využitie, kde nechceme zdielať zdrojový kód, existuje Komerčná licencia Qt.

### 3.1.2 Signály a sloty

Jedna z najväčších odlišností Qt od iných knižníc na tvorbu aplikácií s grafickým užívateľským rozhraním je komunikácia medzi objektami. Qt používa signály a sloty namiesto kedysi používaného spätného volania (angl. callback), ktorý mal dve nevýhody. Pri volaní nebola vykonaná typová kontrola a metóda, ktorá bola volaná, musela poznať ukazovateľ na volajúcu metódu. Použitým signálov a slotov sa tieto problémy odstránili.

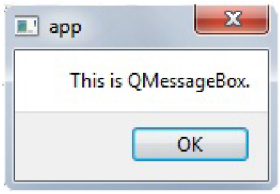
Signál je špeciálna metóda bez návratovej hodnoty, ktorá má iba prototyp (nemá telo). Signály sú emitované objektami pri zmene ich stavu (napr. stlačenie tlačítka). Objekty nevedia o tom, že ich signály príjmajú nejaké iné objekty. Slot je metóda, ktorá je volaná ako odpoveď na prijatý signál, ale môže byť použitá aj ako klasická metóda triedy. V Qt je




Obrázok 3.1: Prepojenie signálov a slotov[1]

množstvo preddefinovaných signálov a slotov a je tu aj možnosť vytvoriť si vlastné a priradiť ich na reakcie k daným udalostiam.

Triedy, v ktorých sa využívajú signály a sloty, musia byť triedy zdedené z triedy `QObject`, v ktorej je definovaná metóda connect. Táto metóda zabezpečuje spojenie signálu a slotu. Je volaná so štyrmi parametrami. Prvý je objekt, od ktorého očakávame signál a druhý je emitovaný signál. Ďalšie sú objekt, ktorý reaguje na daný signál a jeho slot. Príklad komunikácie objektov pomocou signálov a slotov je zobrazený na obrázku 3.1.

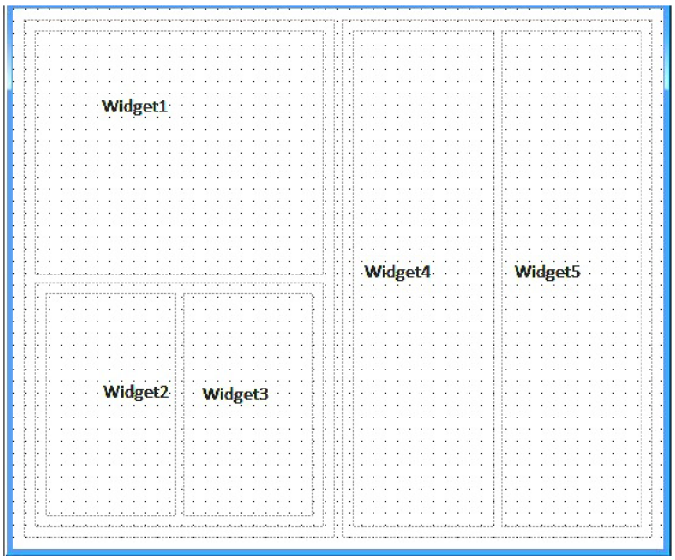



Obrázok 3.2: QMessageBox

### 3.1.3 Prvky GUI

V nasledujúcej časti budú popísané niektoré používané prvky knižnice Qt.

**QMainWindow** je trieda, ktorá sa používa na vytvorenie hlavného okna aplikácie. Jej výhoda je v tom, že umožňuje do okna pridať menu (QMenuBar), nástrojovú (QToolBar) a stavovú (QStatusBar) lištu. Ďalej má miesto, ktoré môže obsahovať ľubovolné widgety (QLabel, QPushButton...).
**QWidget** je asi najpoužívanejšia trieda, slúži ako vrstva medzi prvkom a oknom. Dedí sa z nej veľké množstvo ďalších tried (QMainWindow, QDialog, QPushButton...).
**QLineEdit a QTextEdit** vytvoria políčka na vloženie vstupných údajov. QLineEdit na



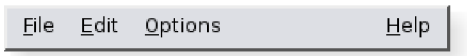

Obrázok 3.3: Priklad vnoreného layoutu


Obrázok 3.4: QPushButton[1]

jednoriadkový[3.7] a QTextEdit pre viac riadkový text.

**QDialog** slúži na tvorbu dialógových okien. Na rozdiel od okien vytvorených pomocou triedy QWidget alebo QMainWindow sa nedajú minimalizovať (maximalizovať) a ani nie je možné meniť ich veľkosť. Od Qdialog-u sú odvodené triedy QMessageBox (zobrazenie textovej správy[3.2], QFileDialog (výber súboru alebo priečinka) a iné.
**QHLayut a QVLayout** sa používajú na rozloženie objektov v rámci okna, nastavujú pozície a rozdiely medzi nimi. Zaručia, že sa prvky prispôsobia pri zmene veľkosti okna. QHLayout rozmiestňuje prvky horizontálne a QVLayout vertikálne. Layouty sa dajú vnorovať do seba[3.3], tým je možné získať zložitejšie rozloženie prvkov v okne.
**QLabel** umožňuje zobrazenie popiskov, textu alebo obrázkov.
**QPushButton a QToolButton**, objekty daných tried slúžia ako tlačítka, ich použitím sa spustí príslušná činnosť. QPushButton je klasicky používané tlačítko popísané textom[3.4]. QToolButton[3.5] nachádza svoje použitie hlavne v nástrojových lištách, obsahuje miesto textu obrázok.
**QToolBar** je trieda slúžiaca na vytvorenie nástrojovej lišty[3.5] v hlavnom okne vytvorenom pomocou QMainWindow.
**QMenuBar**[3.6] vytvorí lištu s menu podobne ako QToolBar, poznáme ju z bežne používaných aplikácií.
**QSpinBox** je to trieda pre číselníkový widget[3.8], je určený na spracovanie celých čísel

Obrázok 3.5: QToolButton a QToolBar[1]




Obrázok 3.6: QMenuBar[1]


Obrázok 3.7: QLineEdit[1]

alebo diskrétnych hodnôt. Umožňuje užívateľovi kliknutím alebo ručne zmeniť zobrazenú hodnotu. Často sa používa v kombinácií s objektom triedy QSlider alebo QScrollBar.
**QScrollBar a QSlider** slúžia na vytvorenie horizontálneho alebo vertikálneho posuvníka. Môžu byť použité na posun a indikáciu pozície v dokumente, ktorý presahuje veľkosť widgetu. Uplatnenie nachádzajú tiež v prípade, kedy posuvník slúži na získanie hodnoty z daného rozsahu[3.8].
Triedy Qt sú väčšinou vytvárané dedením z iných tried. Príklad odvodzovania tried je na obrázku 3.9. Toto je len zlomok tried, ktoré obsahuje knižnica Qt. Ich počet a a druh sa mení v závislosti od verzie, ale býva zaručená spätná kompatibilita. Kompletný prehľad tried a ich metód je možné nájsť v dobre spracovanej dokumentácií na oficiálnej stránke Qt[1].

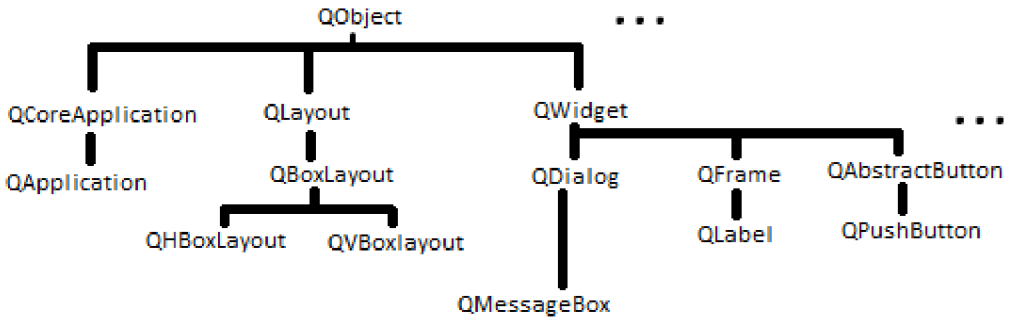
## 3.2 Modularita aplikácií

Ako už bolo povedané modularita znamená rozšíriteľnosť aplikácie. Táto vlastnosť je výhodná hlavne pri väčších projektoch, kde je pri zmene a pridaní novej funkcie systému nutné meniť celý zdrojový kód. Zásah do kódu je zložitá a časovo náročná činnosť, preto sa program rozdeľuje do samostatných a nezávislých častí (modulov), ktoré vykonávajú jednotlivé funkcie. Takto vytvorené moduly je možné pridávať alebo odoberať zo systému, ale tiež použiť aj v iných aplikáciách. Modularita nesie so sebou radu ďalších výhod: prehľadnosť programu, jednoduché rozdelenie úloh pri práci v tíme a možnosť prispôsobenia aplikácie priamo na mieru pre rôznych zákazníkov (tie funkcie potrebujeme, iné nie). Asi jedinou nevýhodou je väčšia náročnosť návrhu aplikácie, kde treba vyriešiť rozhranie modulov a hlavného programu. Moduly používané ako knižnice sa delia do dvoch skupín na statické a dynamické.

### 3.2.1 Statické knižnice

Statické knižnice je pripajané k hlavnému programu už v dobe jeho zostavovania linkerom. Do výsledného programu sa vloží celý kód z knižnice. V prípade využitia danej knižnice viacerými programami, kód knižnice je v operačnej pamäti pre každý osobitne. Uľahčí to prenositeľnosť aplikácie, pretože stačí stačí preniesť iba jeden spustiteľný súbor, ktorý ale má väčšiu veľkosť. Program je horšie modifikovateľný, pretože pre pridanie novej funkcie


Obrázok 3.8: QSpinBox a QSlider




Obrázok 3.9: Hierarchia tried

je nutné všetky knižnice a hlavný program opätovne zlinkovať dokopy. Prípony statických knižníc závisia od operačného systému a použitého linkeru, najčastejšie sa používajú `.a` a `.lib`.

### 3.2.2 Dynamické knižnice

Dynamické knižne sa narozdiel od statických spájajú s programom po jeho spustení alebo až za behu. V operačnej pamäti sa kód nachádza iba raz a môže byť zdielaný aj viacerými programami. Pre beh aplikácie je potrebné mať v počítači uložené všetky využívané knižnice. Z tohto dôvodu musíme pri prenesení programu na iný počítač, nakopírovať spustiteľný súbor a tiež knižnice. Spustiteľný súbor je v tomto prípade oveľa menší ako u statických knižníc. Pre prostredie Windows majú dynamické knižnice príponu `.dll` a pre Unix `.so`. Práve pre vlastnosť linkovania za behu programu, využijeme tieto knižnice v práci na vytvorenie plugin systému. A keďže budeme využívať operačný systém Windows, pôjde o dll knižnice.

# Kapitola 4

# Návrh aplikácie

V tejto kapitole predstavím návrh aplikácie, ktorý bude rozdelený do štyroch častí. V prvej časti uvediem základný popis aplikácie. Druhá časť sa bude zaoberať návrhom grafického užívateľského rozhrania. Nasleduje časť, v ktorej sa oboznámime s rozdelením užívateľov a ich právami na vykonávanie jednotlivých činností. Na záver rozoberiem celkovú štruktúru aplikácie.

## 4.1 Základný popis aplikácie

Vytvárať budem riadiacu aplikáciu sledovacieho kamerového systému. Riadiaca aplikácia je program, ktorý zapuzdruje všetku činnosti systému. Musí obsahovať základné nastavenia systému, výber kamery a sprostredkovať užívateľovi požadovanú funkčnosť. Cieľom je tiež navrhnúť a implementovať užívateľské rozhranie, ktoré bude zabezpečovať komunikáciu s užívateľmi. Sledovací systém bude môcť vykonávať množstvo funkcií. Funkčnosť systému môže byť rozširovaná samostatne implemenovanými modulmi (pluginy), ktoré bude možné do aplikácie pridávať alebo ich z nej odoberať. Modularita je preto dôležitou vlastnosťou aplikácie a jej návrhu treba venovať dostatočnú pozornosť.

Ďalšou úlohou je zabezpečiť aplikáciu pred neželaným prístupom. To znamená, že užívateľ sa musí skôr, ako bude môcť pracovať, prihlásiť. Ruka v ruke s tým ide rozdelenie práv užívateľov, keďže niektorí môžu mať obmedzenú prácu so systémom. S tým súvisí aj ukladanie nastavení do projektov. Aplikácia bude tiež umožnovať výber kamery alebo video záznamu, ktorého obraz budeme chcieť spracovať a zanalyzovať. Je určená pre jednokamerový systém, takže môže byť vybraná iba jedna kamera. Ďalej bude užívateľovi oznamovať dôležité udalosti, ktoré nastanú pri behu programu alebo informácie, ktoré získal modul z obrazu.

Je od nej požadované, ako platí pre väčšinu aplikácií, čo najjednoduchšie a najintuitívnejšie ovládanie. Pri sledovacích systémoch je to obzvlášť dôležité, pretože je potrebné okamžite vykonať reakciu na danú spozorovanú udalosť a pri neprehľadnom a zložitom rozhraní by to mohlo byť problém. Z toho vyplýva, že budem venovať veľkú pozornosť aj návrhu grafického užívateľského rozhrania, ktorý sa často podceňuje. Takto vytvorená aplikácia by mala slúžiť ako prototyp pre vytváraný inteligentný sledovací kamerový systém.

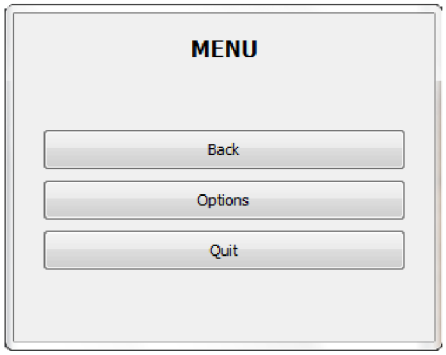
## 4.2 Návrh grafického užívateľského rozhrania

Pred návrhom musím naskôr zostaviť požiadavky na užívateľské rozhranie. Pri práci so systémom je predpokladané, že nie je nutné súčasne využívať aj iné aplikácie, preto je vhodné, aby bola spustená na celú obrazovku (angl. fullscreen application). Týmto aj zaistím väčšiu zobrazovaciu plochu pre obraz z kamery a aj väčší prehľad o dianí v sledovanej oblasti. Z toho je jasné, že hlavnú časť vizuálnej stránky aplikácie bude tvoriť miesto, kde bude zobrazovaný obraz. Aplikácia byť nemala otvárať zbytočne množstvá rôznych okien (napr. upozornenia, varovné hlásenia, nová činnosť modulu v novom okne), a tým vyrušovať užívateľa a sťažovať mu prácu. Ak je to možné, je praktickejšie využívať jedno okno (hlavné okno) pre všetku vykonávanú činnosť. Ďalej je účelné, aby boli základné nastavenia aplikácie rýchlo dostupné. Zároveň, by nemali prekážať pri hlavnej činnosti programu, čo môžem zabezpečiť ich vložením do menu aplikácie. Pri návrhu užívateľského rozhrania som sa zameral najmä na jednoduchosť, pretože zložité ovládanie znižuje kvalitu celej aplikácie. Celkový vzhľad aplikácie musí byť prehľadný. Aby som vytvoril najvhodnejšie grafické rozhranie pre danú aplikáciu, navrhol som postupne 3 rôzne užívateľské rozhrania. Všetky 3 návrhy splňujú vyššie uvedené požiadavky a odlišujú sa viac-menej iba v spôsobe ovládania. Každé z nich malo svoje výhody a nevýhody, ktoré budú popísané nižšie. Nakoniec ich zhodnotím a vyberiem riešenie, ktoré bude najviac vyhovovať zadaným požiadavkám.


Obrázok 4.1: Grafický návrh číslo 1

### 4.2.1 Návrh číslo 1

Prvé užívateľské rozhranie som navrhol tak, aby čo najprehľadnejšie. Ako vidíme na obrázku 4.1, v ľavej časti okna je zoznam pridaných modulov (pluginov) a pri každom z nich sú uvedené dve možnosti. Sú to spustenie a zastavenie vybraného pluginu. Nad zoznamom je tlačítko pre vstup do menu aplikácie. Po stlačení sa zobrazí okno s nastaveniami sa zobrazí okno s možnosťami aplikácie. Ukážka takéhoto okna s možnosťou návratu do aplikácie,




Obrázok 4.2: Ukážka menu z návrhu číslo 1

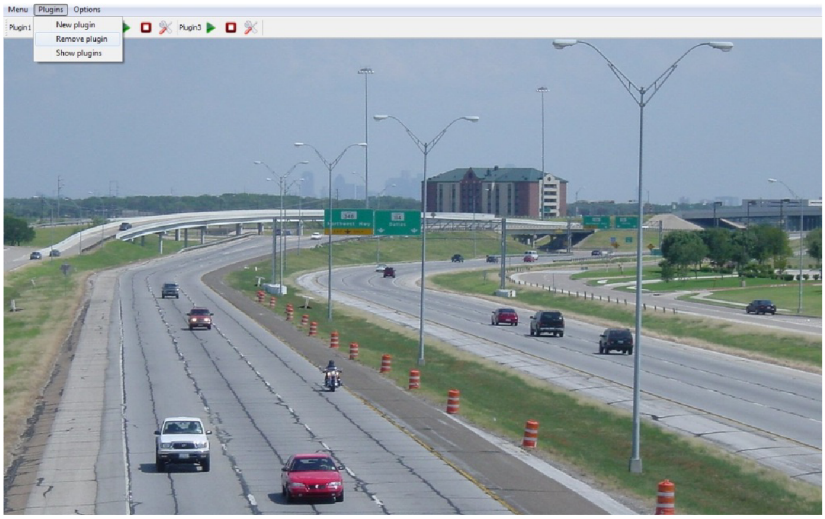
jednou možnosťou nastavení a tlačítkom na ukončenie je na obrázku 4.2. Zvyšná časť okna je venovaná modulom. Do tejto časti bude spustený modul zobrazovať všetko, čo potrebuje (vlastné nastavenia, obraz,...).

**Výhody:**
Tento návrh v sebe zahrňuje niekoľko užitočných prvkov, ktoré uľahčujú prehľad v aplikácií. Asi najväčším pozitívom je neustále zobrazený zoznam pluginov. Máme tak celý čas pod kontrolou, ktoré pluginy sú pridané, ktorý je práve spustený. Veľmi rýchlo dokáže vybrať a zapnúť požadovaný plugin aj menej skúsený užívateľ. Všetký nastavenia aplikácie sú skryté a zobrazia sa až po klinutí na tlačítko menu, čím nezavadzajú pri sledovaní a zároveň sú združené na jednom mieste, kde sú v prípade potreby ihneď k dispozícií.

**Nevýhody:**
Samozerejme skoro každá výhoda prináša so sebou aj nejakú tú nevýhodu. Jednou z nich je vyriešenie prístupu do menu. Aj keď je tlačítko viditeľné a dostatočne veľké, jeho riešenie je iné ako pri bežne zaužívaných aplikáciách (panel s položkami - tzv. menubar), čo môže nových užívateľov odradiť alebo aspoň im v začiatkoch trochu sťažiť prácu. Ďalej pri vstupe do menu sa otvorí nové okno[4.2], ktoré by sa pri rozdelení jednotlivých položiek do klasického panelu s menu nemuselo zobrazovať a súčasne by sa tak ušetrilo jedno stlačenie,čo môže pri sledovaní v reálnom čase zohrať podstatnú úlohu. Napriek dostatočnej veľkosti priestoru, ktorý je prenechaný modulom, ho považujem nevhodne riešený. Moduly budú zobrazovať predovšetkým obraz z kamery a v praxi by sa mali využívať skôr širokouhlé kamery s vysokým rozlíšením. Z toho vyplíva, že môžeme v dôsledku zobraznia zoznamu pluginov na ľavom okraji, stratiť podstatnú časť sledovanej oblasti. Preto by bolo vhodnejšie, aby som ovládacie prvky umiestňoval skôr v hornej alebo spodnej časti okna ako po bočných okrajoch.

### 4.2.2 Návrh číslo 2

V druhom návrhu som odstránil niektoré nevýhody, ktoré som odhalil v predchádzajúcom riešení. Ako na prvý pohľad vidieť[4.3] vo vrchnej časti okna je známy panel s nastaveniami aplikácie. V ňom sú jednotlivé položky združené do logických skupín, a tým som odstránil zobrazenie nového okna s ponukou. Pod ním sa nachádza panel nástrojov (angl. tollbox), kde sú umiestnené pluginy pridané do aplikácie. Pri každom plugine sú 3 tlačítka. Jedno je na spustenie činnosti zvoleného modulu, ďalšie na zastavenie a posledné na zobrazenie




Obrázok 4.3: Grafický návrh číslo 2

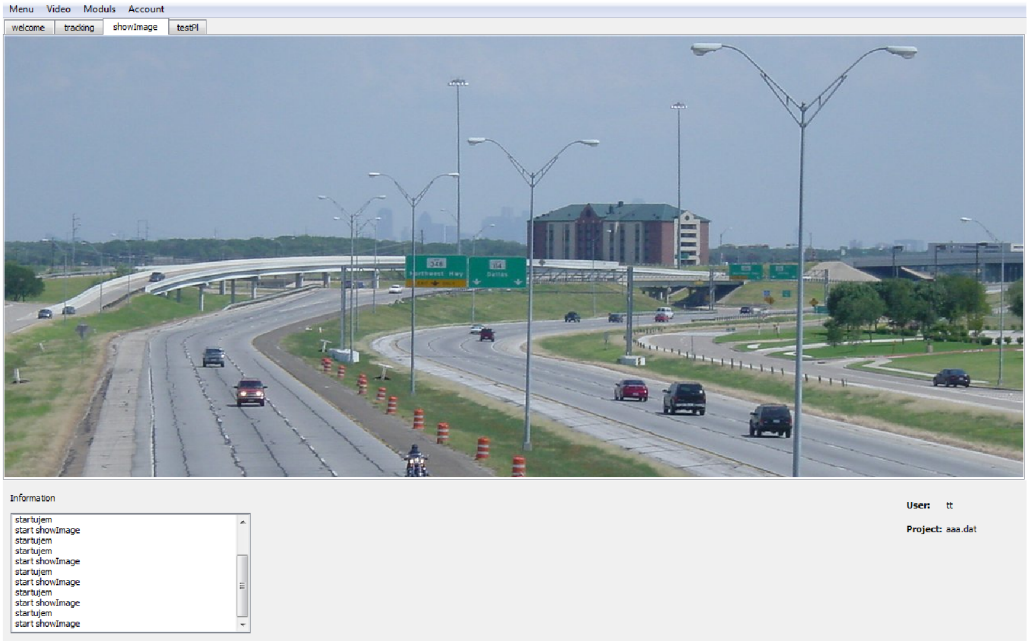
možnosti nastavení, ktoré sú potrebné na správnu funkčnosť pluginu. Ďalej sa nachádza už iba obrovský priestor, ktorý bude plne v réžii spusteného modulu.

**Výhody:**
Hlavnou výhodou tohto riešenia je značná veľkosť oblasti, v ktorej bude zobrazovaný výstup z kamery. Tým bude zabezpečený väčší prehľad a aj väčšia efektivita pri sledovaní oblasti. V predošlom návrhu som ako nevýhodu uviedol umiestnenie a riešenie menu, ktoré je v tomto návrhu prerobené do klasického panelu, ktorý poznáme z bežne používaných aplikácií. Skrytie položiek, zbytočné nerozptylovanie a zasahovanie do práce užívateľa sú vlastnosti menu, ktoré som predtým vyzdvihol, takže ostali zachované aj v tomto návrhu. Teraz by už prístup do nastavení aplikácie samotnej nemal zkomplikovať prácu bežnému užívateľovi.

**Nevýhody:**
Cieľom tohto návrhu bolo odstrániť hlavné nevýhody prvého riešenia, čo sa podľa môjho názoru podarilo. Všetky predtým vytýkané mínusy boli vyriešené, ale vyskytli sa ďalšie zápory. Získanie miesta na zobraznie záberov zo širokouhlých kamier, som musel vykompenzovať zmenením zobrazenia zoznamu pluginov. Navrhnuté zobrazenie je síce dobre viditeľné a zrozumiteľné, ale už nie je také prehľadné a na prvý pohľad jasné ako predtým implementovaný zoznam modulov. Pri väčšom počte pridaných pluginov môže byť problém ihneď sa zorientovať a nájsť modul, ktorý potrebujeme spustiť. Napriek tomu, že ani toto riešenie nie je úplne bez chýb, nemožno ho odpísať, pretože výhody zjavne prevyšujú počet nedostatkov.

### 4.2.3 Návrh číslo 3

V poslednom návrhu, ktorý som vytvoril, som vychádzal z druhého riešenia. Kladne hodnotené menu ostalo implementované rovnako, takže na samotnom vrchu okna ostal umiestený panel sa ponukami. Po rozbalení obsahujú všetky nastavenia a možnosti aplikácie. Pod tým sa nachádza okno so záložkami, kde každá záložka bude reprezentovať jeden samostatný plugin. Jednotlivé záložky si možeme ľubovolne usporiadať, aby sme mali po ruke práve tie,




Obrázok 4.4: Grafický návrh číslo 3

ktoré budeme najčastejšie používať. Po zvolení požadovanej záložky sa spustí priradený plugin. Súčasné spustenie viacerých modulov je možné, ale neustále vykreslovanie obrazu do rôznych záložiek je hardverovo náročné. Preto je vhodné ponechať spustený naraz iba jeden plugin. Vykoná sa to tak, že by sa zastaví vždy posledný spustený modul (posledná aktívna záložka). Zastavenie modulu sa ale prenecháva jeho vlastnej réžii a ak je nutné, aby plugin bežal stále môže. Na obrázku 4.4 je ukážka vzhľadu a vidíme na nej, že k záložkám je miesto na vykreslovanie, ktoré síce nie je také veľké ako v predchádzajúcom prípade, ale vyhovuje požiadavke širokouhlej kamery. Pod touto plochou som umiestnil priestor, kde sa zobrazujú všetky upozornenia a informácie získané z obrazu pomocou modulov. Je dostatočne veľký, tak v ňom môžu byť umiestnené tiež ovládacie prvky kamery (zoom, otáčanie) a ostatné potrebné nástroje.

**Výhody:**
Tento návrh v sebe zahrňuje hneď niekoľko výhodných prvkov. Jedným z nich je panel s menu, ktorý už bol považovaný za dobré riešenie aj v druhom návrhu. Medzi ďalšiu výhodu zaraďujem aj záložkové riešenie pluginov. Aplikácia teraz pripomína internetové prehliadače, ktoré patria medzi každodenne používané programy pre milióny ľudí, preto aj menej zdatní užívatelia sú schopní zvládnuť ovládanie aplikácie bez zaškolenia a veľmi rýchlo. Výhodou je aj automatické zastavanie modulu po spustení iného, kedy sa ušetrí čas a kliknutie potrebné na vyhľadanie a stlačenie daného tlačítka. Výpisovanie varovaní a aktuálne vykonávaných činností v spodnej časti okna je lepšie riešenie ako otváranie nových dialógových okien, preto aj túto môžnosť hodnotím kladne. Celkovo sa javí tento návrh ako veľmi vhodný.

**Nevýhody:**
Ťažko tomuto riešeniu vytknúť veľa vecí, keďže je prehľadné a ľahko ovládateľné. Nedostatkom je len menšia plocha pre zobrazenie obrazu. Zvolil som ju ale tak, že je vhodná

pre použitie kamier, ktorých výstup má širokouhlé rozlíšenie. A pri väčšom množstve pridaných pluginov je trochu menšia prehľadnosť ako v zozname (prvé riešenie), ale to je vykompenzované intuitívnym ovládaním a možnosťou zoradiť ich podľa vlastných potrieb.

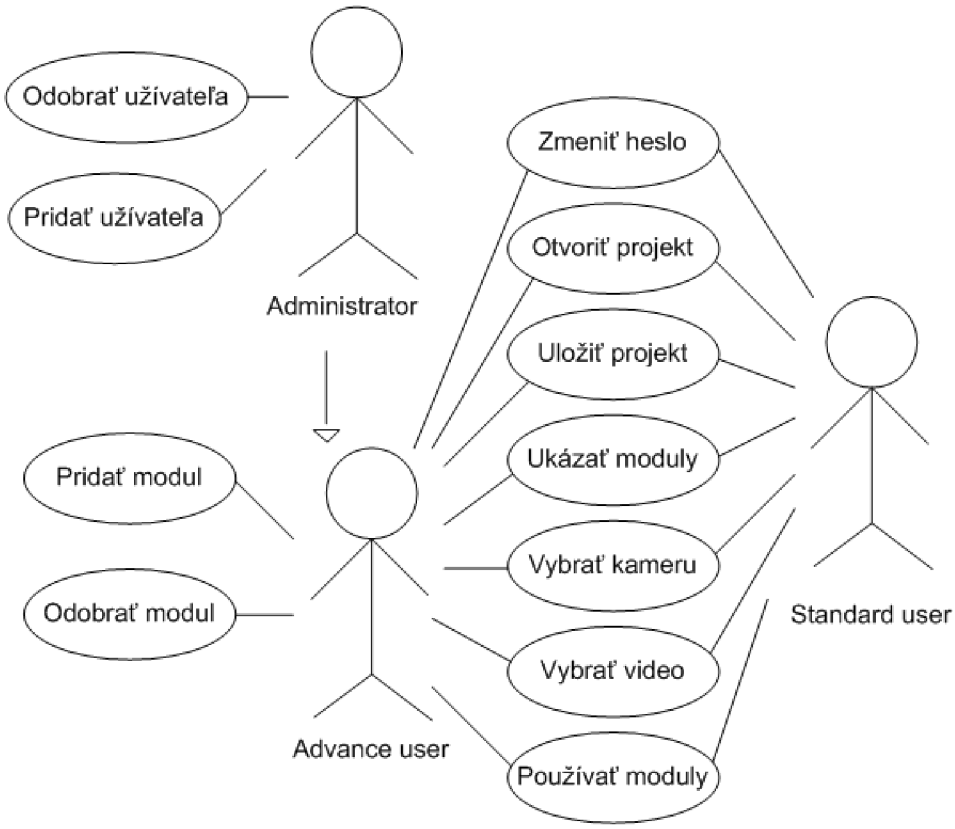
### 4.2.4 Zhodnotenie a výber návrhu užívateľského rozhrania

Myslím si, že nie je možné vytvoriť úplne ideálny návrh vhodný pre všetky typy užívateľov, ale mojou snahou bolo, návrhnúť optimálne riešenie vzhľadom na počet a kvalitu výhod a závažnosť nevýhod. Každé z navrhnutých riešení sa odlišujú hlavne v zobrazení a ovládaní (spustení a zastavení) pluginov. Najdôležitejšou požiadavkou na aplikáciu je zabezpečiť jej modularitu, a preto kladiem dôraz predovšetkým na pohodlnnú prácu s modulmi. Z tohto dôvodu mi najmenej prijateľne vychádza druhý návrh, pretože v prvom je veľmi kladne hodnotený zoznam, kde sú pluginy zobrazené a v poslednom zase veľmi intuitívne a známe ovládanie. V druhom sa riešenie pluginov považuje skôr za zápornú stránku aplikácie. Ďalším kritériom porovnania je priestor prenechaný modulu na zobrazenie obrazu. Zo zvyšných 2 návrhov je určite lepší posledný. Aj keď veľkosťou sú porovnateľné, je situovaný skôr na šírku, čo je pokladané ako výhodnejšie. Za posledný návrh hovorí aj umiestnenie a vyriešenie možností a nastavení aplikácie v panely s menu. Preto som sa rozhodol, že implementovaná aplikácia bude využívať grafické užívateľské rozhranie z tretieho, čiže posledného návrhu.

K ovládaniu aplikácie je potrebné dodať ešte pár detailov. Predpokladá sa, že bude ovládaná prioritne myšou alebo pomocou dotykového displeja. Z dôvodu urýchlenia niektorých činností, sú k najpoužívanejším možnostiam aplikácie priradené klávesové skratky, čo tiež vyzdvihuje zameranie sa na efektivitu práce. Otvárania a pridávania súborov (projekty, video záznamy, pluginy) budú riešené pomocou klasickeho dialógového okna. Takisto sa budú samostatné dialógové okná zobrazovať pri ďalších potrebných možnostiach a nastaveniach aplikácie (informácie o moduloch, zmena hesla,...). Ďalšie podrobnosti ohľadom ovládania budú postupne vysvetlené v nasledujúcej kapitole.

## 4.3 Rozdelenie užívateľov

So systémom bude pracovať niekoľko užívateľov a je možné, že nie každý bude môcť využívať všetky funkcie systému rovanko. Úroveň ovládania aplikácie sa bude líšiť v závislosti na skúsenosti užívateľa s prácou s podobnými systémami a aj s celkovými skúsenosťami s počítačovými aplikáciami. Preto som ich rozdelil na dve skupiny. Sú to klasický (standard user) a pokročilý užívateľ (advance user). Ďalej je potrebné, aby týchto užívateľov niekto pridával, keďže nie vhodné, aby sa užívatelia mohli sami pridavať. Tým by sa znehodnotila funkcia prihlasovania a mohla by sa do aplikácia prihlásiť aj neželaná osoba. Pridávanie užívateľov bude vykonávať administrátor (administrator), ktorý je rozšírením pokročilého užívateľa. Klasický užívateľ bude môcť plnohodnotne využívať aplikáciu, len stratí možnosť ľubovolne pridávať a odoberať moduly, načo je potrebná skúsenosť s aplikáciou a znalosť funkčnosti jednotlivých modulov. Ďalej môžu aj jednotlivé pluginy určovať, ktoré činnosti budú mať možnosť vykonávať jednotlivé typy užívateľov. To ale nemôžem teraz zistiť, keďže pluginy budú vytvarané neskôr a oddelene od aplikácie. Z týchto požadovaných vlastností som vytvoril diagram prípadov užitia[4.5], na ktorom vidíme základné rozdelenie užívateľov a ich možnosti v ovládaní aplikácie.




Obrázok 4.5: Diagram prípadov užitia (Use case)

## 4.4 Návrh štruktúry aplikácie

Návrh vnútornej štruktúry aplikácie som rozdelil do niekoľkých častí. V prvej časti sa budem zaoberať analýzou a návrhom riešenia modularity (plugin systému). V ďalšej rozoberiem postup prihlasovania užívateľa, načítania projektu (relácie) a celkovo spustenie aplikácie. Posledná sa bude venovať návrhu ukážkových pluginov.

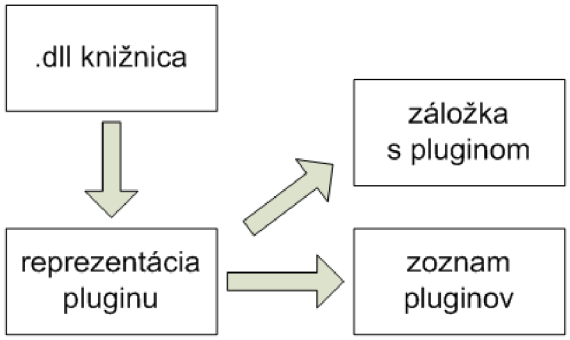
### 4.4.1 Plugin systém

Medzi hlavné požiadavky na aplikáciu patrí zabezpečenie modularity. To znamená, že do už hotovej aplikácie bude možné voľne pridávať samostatné moduly. Aby mohol byť modul pridaný a plne funkčný musí splňovať určené podmienky a samozrejme musí mu to samotná aplikácia dovoliť. Keďže pluginy môžu vykonávať rôzne činnosti, ktoré nevieme dopredu presne určiť, mali by sme im prenechať určitú oblasť. Táto oblasť bude plne v ich réžii. Môže obsahovať nastavenie ich vlastností. Vzhľadom k tomu som sa rozhodol, že každý plugin bude implementovať vlastné grafické užívateľské rozhranie, ktoré bude potom do predaného miesta vložené.

**Rozhranie**
Každý plugin musí byť zdedený od triedy, špecifikujúcej rozhranie na spoluprácu s riadiacou aplikáciou. Táto trieda bude zdedená od triedy `QWidget`, aby mohla implementovať vlastné užívateľské rozhranie. Pri spracovavaní rozhrania je potrebné sa ďalej zamyslieť,

ktoré funkcie musí obsahovať. Na začiatok požadujem získanie názvu pluginu. Ten bude použitý hlavne na pomenovanie záložky. Ďalej je samozrejmé že, keď sa má modul začať alebo skončiť prácu po zvolení záložky, tieto akcie sa budú vyvolávať z hlavnej aplikácie. Preto bude plugin implementovať ďalšie 2 funkcie (spustenie a zastavenie). Jednou z požadaviek bolo tiež ukladanie projektov. Ukladať iba nastavenia aplikácie nemá veľmi zmysel, takže ukladať sa budú aj vlastné nastavenia modulov. Pribudnú mi tak do rozhrania opäť 2 funkcie (uloženie a načítanie). Pre užívateľov, ktorí začínajú s aplikáciou pracovať je vhodné, aby získali od každého pluginu aspoň základný popis. To môže zabezpečiť jedna funkcia. Ako posledná vec, ku ktorej som sa dopracoval je predanie získaných informácií riadiacej aplikácii. Tá potom bude môcť dané informácie zobraziť alebo predať ďalej tretím osobám (napr. po sieti). Nie je ale v našich silách zistiť, kedy nám presne môže plugin informácie ponúknuť, preto je užitočné použiť silný nástroj knižnice Qt. Tým je spojenie signálov a slotov. Modul bude obsahovať signál, cez ktorý bude posielať údaj a bude ho emitovať vždy, keď bude treba. v riadiacej aplikácii bude iba jednoducho tento signál a spojený s požadovaným slotom a problém je vyriešený. Celkovo som tak prišiel na 6 funkcií a 1 signál, ktorý musí rozhranie zahŕňať. Bližšiu špecifikáciu funkcií s názvami a potrebnými parametrami uvediem v kapitole s implementáciou.

**Pridávanie modulov**

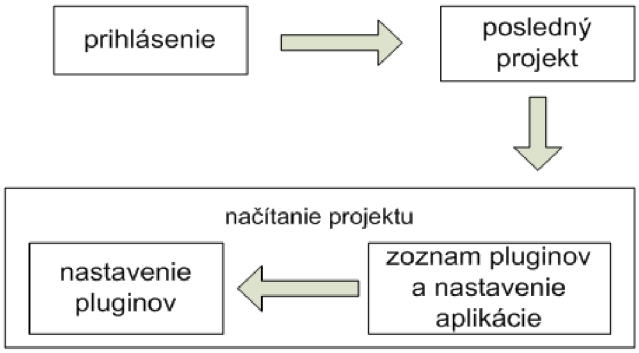



Obrázok 4.6: Postup pridania pluginu

Jednotlivé moduly budú implmentované samostatne ako dll knižnice. Aplikácia preto bude musieť vyriešiť ich pridávanie. Užívateľ bude mať možnosť výberu požadovanej dll knižnice. Po jej zvolení sa vytvorí inštancia triedy (rozhranie). Už síce môžem používať metódy z daného modulu, ale ešte stále nie je prepojený s užívateľským rozhraním. Preto sa pridá nová záložka s názvom pluginu a k nej sa potom priradí inštancia. Takto už môže užívateľ využívať celú funkčnosť, ktorá bude k dispozícii prostredníctvom jednotlivých už spomenutých záložiek. Každá inštancia pridaného pluginu musí byť tiež vložená do určitého zoznamu, aby bolo možné vykonávanie operácií s modulom aj mimo záložiek (napr. odoberanie pluginov). Tento postup je zakreslený na obrázku 4.6.

### 4.4.2 Priebeh aplikácie

V tejto časti popišem, ktoré činnosti musia byť vykonané pri spustení aplikácie. Hneď ako prvé sa zobrazí uvodné okno s prihlásením. Pred prihlásením sa najskôr načíta súbor s účtami užívateľov. Prihlasovacie meno a heslo sa skontroluje a nastavia sa práva na používanie aplikácie. Pri chybnom zadaní sa voľba opakuje. Po zamientutí voľby sa spustí aplikácia




Obrázok 4.7: Načítanie aplikácie

bez práv, kde mám iba možnosť opätovného prihlásenia alebo ukončenia. Po úspešnom prihlásení je vhodné, aby sa načítal posledný otvorený projekt. Ušetrí sa čas potrebný na manuálne otvorenie projektu. Názov tohto projektu je uložený v samostatnom súbore. Tento súbor sa vytvára po odhlásení alebo ukončení aplikácie. Názov projektu už máme zistený, tak môžeme pokračovať v jeho otvorení a načítaní. Súbor s projektom obsahuje vlastné nastavenia aplikácie zoznam pridaných pluginov. Nakoniec sa do programu pridajú postupne po jednom všetky moduly s požadovanými nastaveniami, ktoré sú tiež uložené v samostatných súboroch (pre každý plugin jeden súbor). Až teraz je aplikácia pripravená a môžeme začať pracovať. Priebeh činností potrebných na načítanie aplikácie je graficky znázornený na obrázku 4.7. Tento istý postup musí byť vykonaný aj v prípade odhlásenia a opätovného prihlásenia. Vypadne akurát možnosť načítania obsahu súboru s uloženými účtami užívateľov, keďže to už aplikácia vykonala na samom začiatku. Pri manulálnom otvorení zvoleného projektu prebehnú iba posledné 2 kroky z celého postupu[4.7].

Opačné činnosti, ako uloženie projektu sa budú vykonávať ekvivalentne. Najskôr sa vytvorí súbor s konfiguráciou aplikácie a zoznamom použitých pluginov. Potom postupne budú do prislúchajúceho priečinka vložené súbory pre jednotlivé moduly. Pri pridaní nového užívateľa sa iba doplní do súboru s účtami nová položka.

### 4.4.3 Návrh pluginov

Pre ukážku funkčnosti celého systému som sa rozhodol navrhnúť 3 moduly. Dva z nich budú jednoduché a jeden, ktorý predvedie aj nejakú zložitejšiu činnosť, konkrétne sledovanie objektu. Pri tvorbe každého pluginu je dôležité nezabudnúť na to, že musí byť zdedný od triedy reprezentujúcej rozhranie. Tiež musí obsahovať vlastné grafické užívateľské rozhranie.

Najjednoduchší plugin iba zobrazí snímok na celú zobrazovanú oblasť (bude závisieť od veľkosti daného snímku a samozrejme od rozlíšenia monitoru). Pri tomto module nebude nijak špeciálne implmentovaná metóda na zastavenie činnosti, keďže sa vykoná iba jednorázovo a nebude prebiehať neustále.

Ďalší už bude zobrazovať video (zo súboru alebo z kamery). Bude zobrazené tiež na celú plochu. Z videa bude postupne vyberať jednotlivé snimky. Výber snímky nastane vždy po vypršaní časovača. Tie sa budú zobrazovať rovnakým spôsom ako pri prvom plugine. Ukončenie činnosti je nutné, aby sa zbytočne nezvyšovali nároky na pamäť a procesor, pri vykreslovaní do neaktívnej záložky. Toto umožní metóda na zastavenie, v ktorej sa jednoducho ukončí časovač a k ďalšiemu výberu snímok už nebude dochádzať.

Posledný ukážkový plugin už bude obsahovať aj zmysluplnejšiu funkčnosť. Bude schopný vo vybranom videu detekovať pohyb a sledovať pohybujúci sa objekt. Túto činnosť bude vykonávať na základe metódy odčitania pozadia (angl. background substraction). Tento algoritmus je vhodný na sledovanie objektov na statickom pozadí a pracuje celkom spoľahlivo. Návrh grafického rozhrania s nastaveniami pluginu a triedu, ktorá bude vykonávať funkčnosť poskytol do mojej práce Boris Švancar. S mojimi úpravami v poskytnutých materiáloch môžem vytvoriť triedy, ktoré je možné použiť v module. Tiež je potrebné vyrobiť triedu, ktorá bude zdedená od rozhrania a bude ich zastrešovať. Táto trieda bude vyzerať a pracovať podobne ako predchádzajúci plugin.

# Kapitola 5

# Implementácia

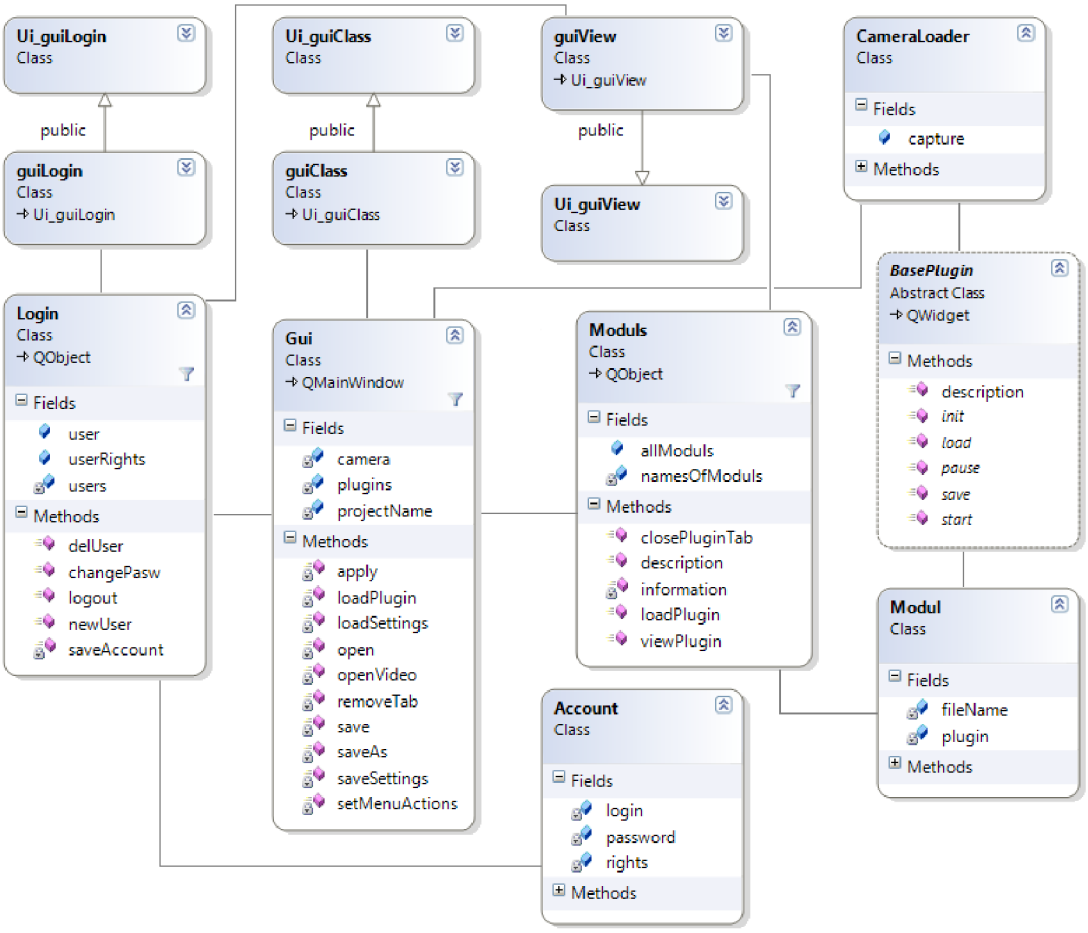
Aplikáciu som implementoval v jazyku C++ s využitím multiplatformnej knižnice Qt, ktorá mi ponúkla nástroje na tvorbu grafického užívateľského rozhrania a pluginov. Ďalej som použil aj funkcie knižnce OpenCV pre prácu s obrazom z kamery alebo videosúboru. Ako vývojové prostredie som použil *Microsoft Visual Studio 2008* a na tvorbu grafického užívateľského rozhrania *Qt Designer*, kde je možné rýchlo a jednoducho vložiť požadované komponenty.

## 5.1 Implementácia tried aplikácie

Táto časť sa bude venovať popisu jednotlivých tried, ktoré boli impemntované v aplikácii. Na začiatok vidíme na obrázku 5.1 diagram použitých tried s vybranými metódami a premennými, ktoré stoja za zmienku. Je v ňom znázornené vzájomné poprepájanie tried a dedičnosť. Ďalej popíšem jednotlivé logické celky aplikácie a triedy, ktoré sú použité na ich implemntáciu.

### 5.1.1 Riadiaca trieda a hlavné okno

Grafické užívateľské rozhranie hlavného okna, vytvoreného v *QT Designer*-i popisujú dve triedy `Ui_guiClass` a `guiClass`. Pri tvorbe sú využité komponetny `QMenuBar` na tvorbu panela s menu a `QTabWidget` na záložkové okno pre moduly. Ďalej som použil rôzne rozloženia (`QHBoxLayout` a `QVBoxLayout`) a niekoľko `QLabel`-ov na vloženie textových popiskov a `QTextEdit` na výpis hlásení. Táto trieda je potom priradená do triedy `Gui`, ktorá riadi všetku funkčnosť aplikácie. Gui je zdedená z triedy `QMainWindow` a je použtá ako hlavné okno aplikácie. Obsahuje všetky prepojenia slotov na signály vyvolané v menu. Sloty `open()`, `save()` a `saveAs()` implmentujú otváranie a ukladanie relácie(projektu). Ukladanie sa vykonáva pomocou triedy `QDataStream` na prevod do binárnych dát a uloženie do súboru. Každý projekt pozostáva zo súboru `*.proj` (uložená konfigurácia hlavnej aplikácie a zoznam pluginov) a priečinka pomenovaného rovnako ako projekt, ktorý obsahuje súbory `*.dat` (pre každý plugin jeden) s uloženou konfiguráciou. V týchto slotoch sú využité aj metódy triedy `Moduls`, pretože potrebujem uložiť aj nastavenia jednotlivých pluginov. Sloty tejto triedy sú pripojené ku všetkým akciám vyvolaným z menu, ktoré pracujú s pluginmi (odobratie pluginu, informácie o nich). Pridanie pluginu sa vykonáva pomocou slotu `loadPlugin()` hlavnej triedy, v ňom sa využíva metóda z `Moduls`, aby bolo možné pridanie záložky s prisluchajúcim pluginom. Pridanie je vykonané pomocou nasledovnej konštrukcie, kde `ui.tabWidget` je záložkové okno, `plug` je inštancia triedy `BasePlugin` zo zvoleného

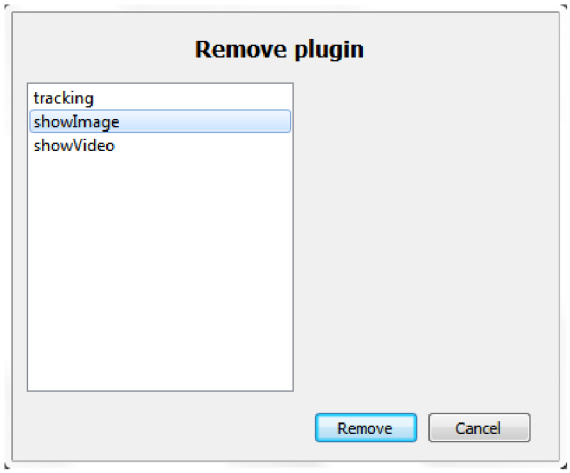



Obrázok 5.1: Diagram tried aplikácie (vygenerovaný vo Visual Studiu 2008)

pluginu a metóda `init()` má ako návratovú hodnotu jeho názov:

```
ui.tabWidget->addTab(plug, plug->init())
```

Slot `openVideo()` používam na výber videosúboru a z vybraného súboru vytvorí trieda `CameraLoader` pomocou funkcií z knižnice OpenCV `CvCapture`, ktorý umožňuje postupný výber jednotlivých snímkov. Podobný slot je aj `openCamera()` (nenachádza sa v diagrame tried) s tým rozdielom, že sa ako zdroj obrazu zvolí USB kamera. Slot `apply()` je prepojený so signálom, ktorý je vyvolaný pri zmene záložky. Vykoná zavolanie metódy `start(CameraLoader *cam)` pluginu z aktuálnej záložky (parameter je ukazovateľ na triedu obsahujúcu zvolený obrazový vstup) a `pause()` z predchádzajúcej. Na akcie spojené s prihlasovaním, zmenou hesla, pridaním, vymazaním užívateľa a odhlasením sú pripojené sloty triedy `Login`, ktoré to vykonajú samostatne. Jedine po úspešnom prihlásení a odhlasení vyšlú signály (`updateGuiIn()`, `updateGui()`), aby sa mohlo obnoviť hlavné okno aplikácie. Reakciou na tieto signály sú 3 sloty. Prvý je `setMenuActions()`, ktorý podľa práv užívateľa nastaví prístup k jednotlivým možnostiam. Nasleduje `saveSettings()`, ktorý je volaný po odhlásení alebo pri ukončení aplikácie a pomocou neho uložím posledný aktuálny




Obrázok 5.2: Dialógové okno na odobranie pluginu

názov projektu do súboru `config.conf` nachádzajúceho sa v priečinku so spustiteľným súborom. Na načítanie názvu projektu z daného súboru po prihlásení slúži `loadSettings()`.

### 5.1.2 Operácie s modulmi

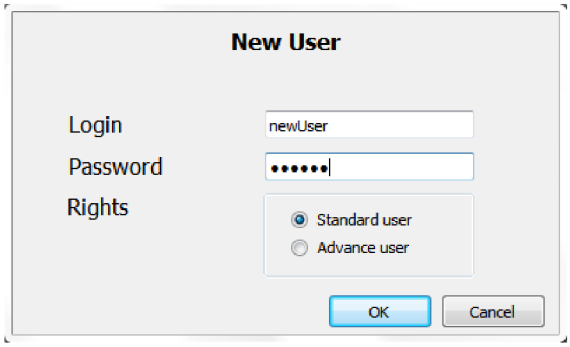
Všetky operácie ohľadom modulov zastrešuje trieda `Moduls`. Obsahuje 2 zoznamy. Jeden sa naplňuje názvami pridaných pluginov a druhý je zoznam tried typu `Modul`. Táto trieda reprezentuje jeden plugin, obsahuje inštanciu triedy `BasePlugin` a celú cestu s názvom dll knižnice. Teraz popíšem významné metódy triedy `Moduls`. Pri pridaní pluginu sa vykonáva metóda `loadPlugin(QString fileName)`, kde parameter je kompletná cesta k pluginu. V nej sa najskôr skontroluje, či modul z danej knižnice už nie je náhodou pridaný (ak je, vypíše hlásenie). Ďalej sa vytvorí inštancia pluginu a potom následne inštancia triedy `Modul`. Nakoniec sa vloží do zoznamov kompletný názov dll knižnice a inštancia `Modul`-u. Mazanie pluginu je v réžii metódy `removePlugin()` (nie je v diagrame tried). Prebieha tak, že sa otvorí dialógové okno (obr. 5.2), kde zvolíme podľa názvu plugin, ktorý chceme odobrať a pokračujeme stlačením tlačítka OK. Zo zoznamov sa vymažú prislúchajúce položky a vyšle sa signál (`closePluginTab(int index)`) hlavnej triede na zatvorenie potrebnej záložky. Posledná dôležitá metóda je `viewPlugin()` (zobrazuje informácie o pridaných pluginoch), ktorá rovnako ako `removePlugin()` otvorí okno, kde sú zobrazené názvy pluginov. Po kliknutí na názov sa vyšle signál `clicked(const QModelIndex &)` triedy `QListView`, na ktorý je napojený slot `information(const QModelIndex &)`. Ten obsahuje popis k danému pluginu a emituje naspäť signál `description(QString text)`, kde parameter je spomenutý popis. Tento signál je zachytený a popis sa zobrazí v komponente `QTextEdit`.

Vzhľad oboch okien (pri mazaní na obrázku 5.2 alebo informáciách o pluginoch) a rozloženie komponentov popisuju triedy `Ui_guiView` a `guiView` vygenerované z *Qt Designer*-u. Okno ako základ využíva `QDialog` a obsahuje ďalej `QListView`, `QLabel`, `QTextEdit` a `QPushButton`.

### 5.1.3 Správa užívateľských účtov

Správu užívateľov, prihlasovanie a odhlasovanie má na starosti trieda `Login`. Obsahuje zoznam, ktorý má obsah typu triedy `Account` a dve statické premenné meno užívateľa (`login`)

a jeho práva na používanie aplikácie (`rights`). Trieda `Account` reprezentuje jeden užívateľský účet s menom, heslom a právami. `Login` implmentuje sloty, ktoré sú pripojené priamo na akcie z menu spojené s položkami na možnosti účtov. Jedným takým je `signIn()` (nie je v diagrame tried), ktorý zabezpečuje otvorenie prihlasovacieho okna. Po zadaní a potvrdení zabezpečí načítanie všetkých užívateľských účtov zo súboru (`users.dat`). Nasleduje kontrola mena a pri nájdenej zhode aj kontrola hesla. Po úspešnom prihlásení nastaví statické premenné a vyšle signál do hlavnej triedy `Gui`. Po vložení chybného údaju zobrazí opäť prihlasovacie okno aj s upozornením. Podobný slot je `changePasw()` a implementuje zmenu hesla práve prihláseného užívateľa. Tiež otvorí podobné okno, kde je možné zadať nové heslo a potvrdiť operáciu. V zozname účtov sa aktuálny nahradí novým so zadaným heslom a pomocou triedy `QDataStream` sa uloží do súboru `users.dat`. Vytvorenie nového účtu poskytuje `newUser()`. Otvorí okno s rovnakým základom u predošlých prípadoch a navyše obsahuje zvolenie typu užívateľa (standard user alebo advance user). Následné potvrdenie vložených údajov zavolá `saveAccount()`. Ten kontroluje meno, či nie je už rovnaké vložené. Ak sa zhoduje okno sa zobrazí znovu s hlásením, ak nie vloží nový účet do zoznamu a uloží ho do súboru. Posledným napojeným slotom je `delUser()`. Tento slot otvorí okno podobné tomu ako je na obrázku 5.2. V ňom vyberieme požadovaného užívateľa a potvdíme voľbu vymazania. Účet bude odobraný zo zoznamu a zoznam opätovne uložený do súboru. Okno




Obrázok 5.3: Dialógové okno na vytvorenie účtu

na prihlasovanie, zmenu hesla a vytvorenie nového účtu je popísané v triedach `Ui_guiLogin` a `guiLogin`, ktoré sú vygenerované z *Qt Designer*-u. Príklad je na obrázku 5.3. Druhý typ okna je popísaný rovnakými triedami ako v predchádzajúcej časti.

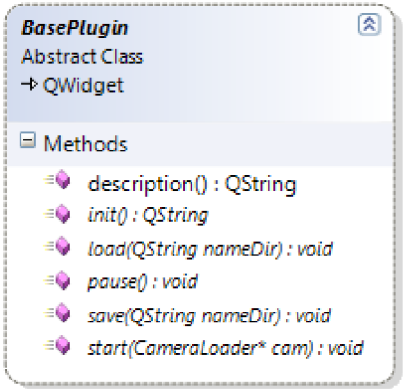
## 5.2 Implementácia tried pluginu

Implementácia pluginov je veľmi dôležitá, pretože vlastne celú funkčnosť inteligentného kamerového systému zabezpečujú práve oni. V tejto časti naskôr úplne definujem triedu rozhrania pluginu a potom popíšem triedy vybraného ukážkového pluginu.

### 5.2.1 Rozhranie pluginu

Je dôležité, aby bolo umožnené použitie pluginu v riadiacej aplikácii. Preto musí byť každý modul zdedený od triedy `BasePlugin`, ktorá definuje rozhranie. V triede je použité makro knižnice Qt `Q_DECLARE_INTERFACE(BasePlugin,''ioss.plugin.interface'')` potrebné na vytvorenie rozhrania. Príklad triedy uvádzam na obrázku 5.4. Trieda je zdedená od

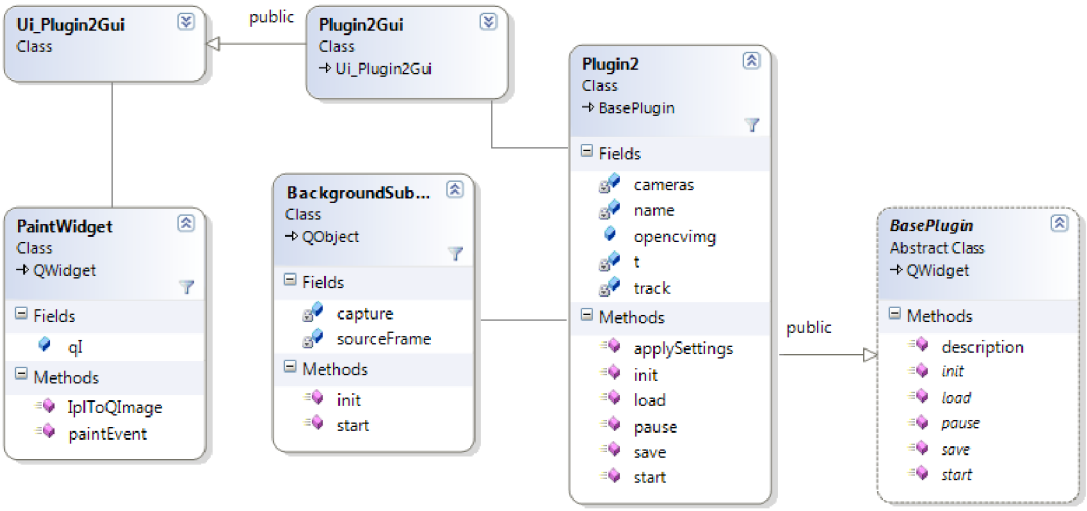
triedy knižnce Qt `QWidget`. Obsahuje 5 abstraktných metód, ktoré nutne musí plugin implentovať a jednu virtuálnu, ktorú môže preťažiť. Viruálna metóda je `description()`, ktorej návratovou hodnotou je popis modulu v reťazci `QString`. Bez preťaženia vracia iba prázdny reťazec. Ďalej metóda `init()` poskytuje, tiež v reťazci `QString`, názov pluginu. Metódy `load(QString)` a `save(QString)` volá riadiaca aplikácia pri načítaní (prípadne ukladaní) projektu. Ako parameter dostávajú celú cestu k priečinku, v ktorom bude súbor s nastaveniami uložený. Posledné dve sú asi najpodsatnejšie. Spustenie pluginu z aplikácie bude implementované v metóde `start(CameraLoader* cam)`. Argumentom je ukazovateľ na inštanciu triedy `CameraLoader`, ktorá obsahuje prístup ku kamere alebo zvolenému videu. Nakoniec `pause()`, kde bude modul implemenentovať to, čo má nastať v prípade neaktívnej záložky (najčastejšie zastavenie činnosti). Trieda `BasePlugin` obsahuje ešte signál `logInformation(QString info)`, ktorý nesie so sebou popis momentálne vykonávanej činnosti alebo údaj zistený z obrazu a môže byť emitovaný v ktoromkoľvek okamihu za behu činnosti.




Obrázok 5.4: Trieda BasePlugin

### 5.2.2 Ukážkový plugin

Na ukážku implementácie som si vybral popis najzložitejšieho z vytvorených pluginov. Je to modul na detekovanie a sledovanie objektov vo videosekvenciách. Po skompilovaní má názov `plugin2.dll`. Je vytvorený pomocou niekoľkých tried a ich diagram s výberom metód je zobrazený na obrázku 5.5. Hlavná trieda `Plugin2`, ako už bolo povedané, je zdedená od rozhrania. Je v nej nastavené grafické užívateľské rozhranie, ktoré popisujú triedy vygenerované v programe *Qt Designer* (`Ui_Plugin2Gui` a `Plugin2Gui`). Obsahujú množstvo rôznych prvkov, pomocou ktorých je možné meniť parametre úrovne detekcie. Ďalej je v nich zakomponovaná trieda `PaintWidget`, ktorý je zdedený od triedy `QWidget`. Do tohto kompnentu sa zobrazuje obraz. Preťažuje `paintEvent(QPaintEvent*)`, kde sa s využitím triedy `QPainter` vykreslí `QImage`. Táto metóda sa volá pri prekreslovaní záložky pomocou metódy `repaint()`. Jedným s najväčších problémov bolo, že algoritmy na spracovanie obrazu pracujú so snímkom vo formáte `IplImage` (knižnca OpenCv) a ten nie je možné vykreslovať do `QWidget`-u priamo. Implmentuje preto tiež metódu `IplToQImage(IplImage* img)`, ktorá zabezpečuje prekonvertovanie obrázku na `QImage`. Problémom tejto metódy je, že sa musí každý zobrazovaný snímok opätovne kopírovať v pamäti, čo celý proces vykreslenia spomaľuje.




Obrázok 5.5: Diagram tried pluginu

Trieda `Plugin2` implementuje ďalej abstraktné metódy rozhrania. `Init()` iba jednoducho vracia názov modulu z premennej `name`. Pre príklad použitia metóda `load()/save()` načíta respektívne uloží názov pluginu zo súboru. Metóda na spustene činnosti `start()` najskôr zavolá metódu `init()` z triedy `BackgroudSubtraction`, kde jej predá ukazovateľ na `CvCapture`. Potom pomocou časovača volá metódu `start()` tiež rovnakej triedy. Tá má návratovú hodnotu zobrazovaný snímok vo formáte `IplImage`. Tento sa skonvertuje a zavolá sa prekreslenie záložky(`repaint()`). Zastavenie je vykonané zrušením časovača v metóde `pause()`. Obsahuje ďalej metódu `applySettings()`, ktorá sa volá po potvrdení zvolených nastavení. Tie aplikuje opätovným volaním `init()` z `BackgroudSubtraction`. Nakoniec aby mohol byť použiteľný a bola exportovaná celá trieda je potrebné, aby obsahoval makro z knižnice Qt `Q_EXPORT_PLUGIN2(pluginTracking, Plugin2)`.

Trieda `BackgroudSubtraction` implemnetuje aj ďalsie metódy, ktoré potrebuje na vykonanie detekcie a sledovania pomocou algoritmu odčítania pozadia. Tento spracovaný algoritmus poskytol a spracoval vo svojej bakalárskej práci Boris Švancar.[8]

# Kapitola 6

# Testovanie

Neexistuje veľa možností, ako by som mohol otestovať vytvorenú aplikáciu. Rozhodol som sa pre posúdenie aplikácie z dvoch hľadísk. Prvé zistí ako vhodne je zvolené grafické rozhranie a jeho užívateľskú prívetivosť pri práci. V druhom prípade zhodnotím funkčnosť aplikácie.

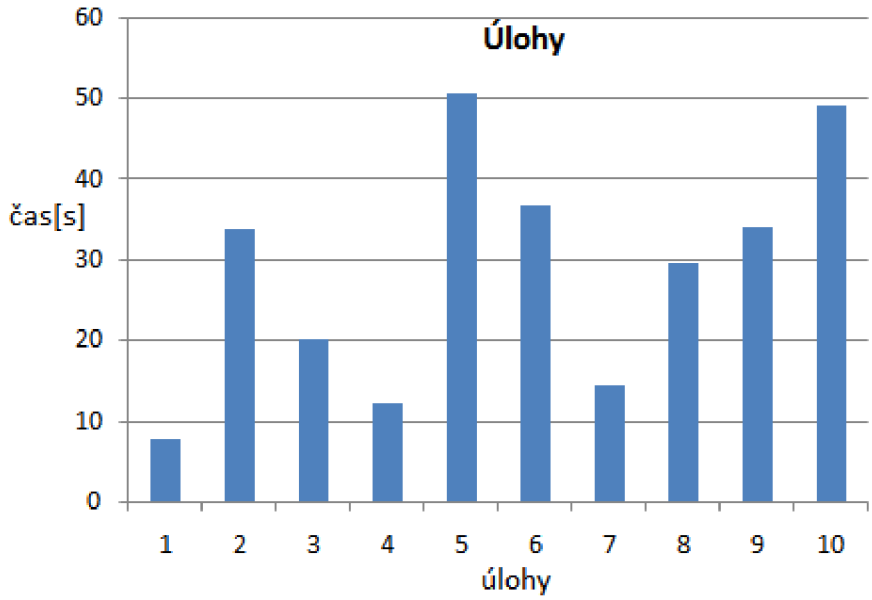
## 6.1 Užívateľská prívetivosť

Ceľom tohto testovania je zistenie informácií ohľadom používania aplikácie. Zameriam sa na rýchlosť vyhľadania ovládacích prvkov pri vykonávaní jednotlivých úkonov. Ďalej zistím mieru prehľadnosti vytvoreného návrhu užívateľského rozhrania a intuitívnosť práce s ním. Na toto testovanie som prizval niekoľkých respondentov, ktorých úlohou bolo vykonať súbor predom určených činností. Testy prebiehali osobne, preto som zvolil ako optimálny počet respondentov 15. Väčšia časť z nich (10) bola z radov študentov vo veku 18-26 rokov, zvyšok (5) boli zamestnanci v rôznych profesiách vo veku 30-55 rokov. Pred začiatkom som každého v krátkosti oboznámil s tým, čo aplikácia umožňuje.

Do aplikácia bol pred testovaním pridaný iba jeden administrátorský účet, nebol uložený ani jeden projekt, preto sa začínalo z nulového stavu. Na vyskúšanie ovládania aplikácie museli vykonať nasledovné úlohy:

1. Prihlásiť sa na administrátorský účet (login: admin, heslo: 12345).
2. Vytvoriť 2 nové účty (standard a advance user).
3. Odhlásiť sa a prihlásiť sa ako pokročilý užívateľ.
4. Zmeniť heslo k účtu.
5. Pridať všetky 3 pluginy a uložiť projekt.
6. Odobrať plugin, ktorý vykonáva sledovanie objektov a uložiť ako nový projekt.
7. Odhlásiť sa a prihlásiť sa ako štandardný užívateľ.
8. Vybrať USB-kameru a spustiť plugin zobrazujúci video.
9. Vybrať ľubovolný súbor s videom a spustiť plugin so sledovaním.
10. Odobrať obidva pridané účty, odobrať pluginy, uložiť, odhlásiť a ukončiť.

Každému respondentovi som zmeral čas, za ktorý bol schopný jednotlivé činnosti vykonať a sledoval som jeho postup, hlavne pri posledných dvoch úlohach, ktoré sú v porovnaní s predchádzajúcimi trošku náročnejšie. Výsledky časov som spracoval a priemerné časy vykonávania úloh sú zaznamenané v grafe 6.1.




Obrázok 6.1: Premerný čas vykonávania jednotlivých úloh

S úspešným vykonaním prvých 4 úloh nebol žiadny problém a aj tí, ktorý s počítačom prichádzajú do styku menej si s tým poradili celkom rýchlo. To znamená, že ovládacie prvky ohľadom správy užívateľov sú prehľadné a zrozumiteľné. ďalšie 2 mali za úlohu ohodnotiť spôsob pridávania a odoberania pluginov a ukladanie projektu. Niektorí respondenti nevedeli, čo je to plugin, a preto nevedeli ako to majú urobiť. To malo za následok relatívne vysoký priemerný vykonávací čas. Po následnom vysvetlení ich už úspešne pridali a uložili projekt, s čím už problém nebol. Vytýkané bolo iba, že nie je možné súčasne pridať viac pluginov a musia potom opätovne vykonávať tú istú činnosť. V praxi by ale nemalo byť bežné, že bude užívateľ naraz pridávať viac modulov (iba pri tvorbe projektov), keď si môže otvoriť už vytvorený projekt. Odobratie a následným uloženie zvládli úspešne. Úlohy pokračovali opäť odhlásením a prihlásením, čo prebehlo v priemere o 5 sekúnd rýchlejšie ako predtým, takže po zoznámení sa s týmto ovládacím prvkom vidieť zlepšenie. Veľmi rýchlo zvládli aj výber kamery a spustenie požadovaného pluginu. ďalšiu úlohu som považoval za náročnejšiu, keďže potrebný plugin nebol v danom projekte a nemali práva na jeho pridanie, ale zjavne som sa mýlil. Úplne samozrejme všetci až na jedného otvorili predtým uložený projekt, kde bol daný plugin pridaný, zvolili video a spustili ho. Ten jeden respondent sa prihlásil ako pokročilý užívateľ, pridal plugin a až potom ho spustil, čo tiež hodnotím kladne, pretože už po jednom vyskúšaní si zapamätal ovládacie prvky a vhodne a rýchlo ich použil. Zložitosť poslednej úlohy spočívala iba v zistení, že sa musia prihlásiť ako administrátor, aby mohli vymazať účty. To zvládli relatívne dobre a následné odstránenia účtov a pluginov už bola len formalita. Čas potrebný na vykonanie sa zdá byť vysoký, ale pri počte dielčích úloh, ktoré mali spraviť je dobrý. Celkový čas celého súboru úloh neprekročil v priemere 5 minút. Najskúsenejší s respondentov to stihol dokonca za

menej ako 2,5 minúty a najmenej skúsený za niečo viac ako 6,5 minúty.

Vzhľadom na rýchle zoznámenie sa s prostredím a intutívnu prácu, hodnotím navrhnuté rozhranie za vhodné. Respondenti, ktorí mali tú možnosť vyskúšať si aplikáciu hodnotia jej užívateľskú prívetivosť ako dobrú až veľmi dobrú. Veľmi kladne sa vyjadrili ohľadom panela s menu, ktorý už poznali z iných aplikácií. Ani najmenší problém nemali s tým, aby sputili požadovaný plugin, preto aj toto riešenie, považovali za účelné a dobre vymyslené. Výsledkom tohto testu je zistenie, že práca s aplikáciu je intuitívna, jednoduchá a po krátkom zoznámení sa je čas potrebný na ovládanie jednotlivých činností ešte kratší, čo návrhu dáva veľký potenciál.

# Kapitola 7

# Záver

Cieľom bolo navrhnúť a implementovať základnú riadiacu aplikáciu pre kamerové sledovacie systémy, ktorá by našla svoje využitie pri vývoji inteligentného poloautomatického sledovacieho systému. Veľmi doležité je, aby výsledná aplikácia bola rozšíriteľná a umožnila pridávať nové moduly.

V práci sme sa najskôr oboznámili s obecným sledovacím systémom a následne boli predvedené a porovnané niektoré dostupné aplikácie, ktoré umožňujú ich riadenie. Predstavili sme si knižnicu Qt, jej vlastnosti, prvky potrebné na tvorbu grafického užívateľského rozhrania. Zoznámili sme sa s pojmom modularita, statické a dynamické knižnice. Predviedli sme si spôsob návrhu užívateľského rozhrania, štruktúry aplikácie a jej naslednú implmentáciu. Nakoniec sme vytvorenú aplikáciu podrobili testovaniu.

Z testovania užívateľskej prívetivosti vyšla aplikácia veľmi dobre. Prebiehalo tak, že vybraný počet respondentov rôzneho zamerania a veku muselo vykonať súbor úloh. Pri ich práci som sledoval reakciu na užívateľské rozhranie a čas, za ktorý boli schopný úspešne zvládnuť jednotlivé činnosti. Namerané hodnoty boli dobré, čo zanamená, že ovládacie prvky aplikácie sú vhodne združené do skupín, práca je intuitívna a reakcia je rýchla. Celkový návrh rozhrania bol hodnotený veľmi kladne a spôsob ovládania vyhovolal každému z prizvaných respondentov.

Posúdenie z hľadiska funkčnosti implementovanej aplikácie spočívalo v nájdení nedostatkov aplikácie a určení riešenia ako ich vhodne odstrániť. Niektoré z objavených problémov boli závažnejšie a ich odstránenie bude zložitejšie (frekvencia zobraziteľných snímkov s vysokým rozlíšením). Tiež sa však našli aj nedokonalosti, ktorých opravenie bude takmer triviálna záležitosť. Z komplexného pohľadu je ale aplikácia funkčná a vhodná na demonštráciu fungovania systému.

Zhodnotením výsledkov som dospel k záveru, že aplikácia v tomto štádiu nie je úplne vhodná na použitie v praxi, obsahuje viacero nedostatkov, ktoré je potrebné predtým odstrániť. Navrhnuté a vytvorené jadro je funkčné, a preto táto práca položila základný kameň vývoja riadiacej aplikácie pre inteligentný sledovací systém. Na tvorbe a vylepšovaní tejto aplikácie by som sa chcel aj naďalej podieľať ako súčasť tímu vo výskumnej skupine na našej fakulte.

# Literatúra


[1] Stránka projektu Qt. [online], 2011, [cit. 2011-01-20].
URL `<http://qt.nokia.com>`

[2] Stránka spološnosti Milestone. [online], 2011, [cit. 2011-04-07].
URL `<http://www.milestonesys.com>`

[3] Stránka spoločnosti. [online], 2011, [cit. 2011-04-06].
URL `<http://www.netcam.cz>`

[4] Stránka spoločnosti BOSCH pre bezpečnostné systémy. [online], 2011, [cit. 2011-04-08].
URL `<http://products.boschsecuritysystems.eu>`

[5] Dalheimer, M.: *Programming with Qt.* Cambridge: O'Reilly, 2002, ISBN 0-596-00064-2.

[6] Javed, O.: *Automated multi-camera surveillance: algorithms and practice.* New York: Springer, 2008, ISBN 978-0-387-78880-7.

[7] Molkentin, D.: *The book of Qt 4:the Art of Building Qt Applications.* San Francisco: No Starch Press, 2007, ISBN 978-1-59327-147-3.

[8] Švancar, B.: *Sledování objektů v sekvenci snímků.* Bakalářská práce, Vysoké učení technické, Brno, 2011.

# Dodatok A

# Obsah CD

Priložené CD obsahuje tieto zložky:

- bin/ – spustiteľná aplikácia
- src/ – projekt pre Microsoft Visual Studio 2008
- config/ – konfiguračné súbory a uložené projekty
- videos/ – ukážkové videá
- latex/ – zdrojový kód textu bakalárskej práce v LATEXu
- pdf/ – text bakalárskej práce v .pdf

# Dodatok B

# Manuál

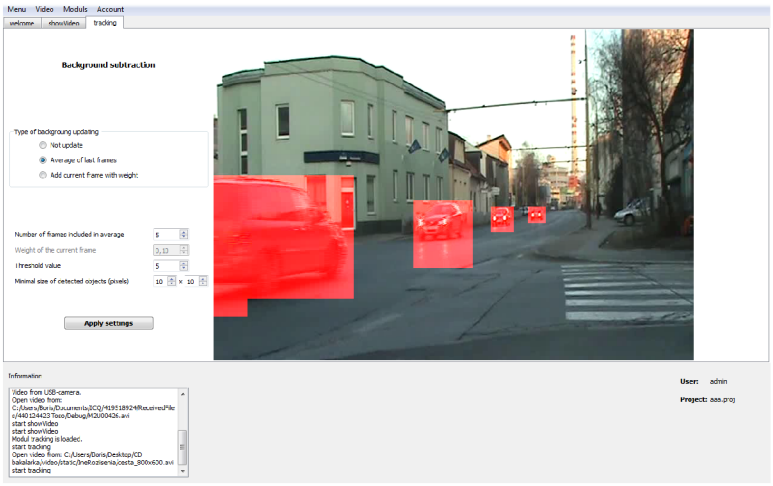
Na spustenie aplikácie je nutné sa prihlásiť. Pre uspešné prihlásenie je potrebné nakopírovať súbor users.dat z priečinka config/ do priečinka so spustiteľnou aplikáciou.

Uložené užívateľské účty s súbore users.dat:

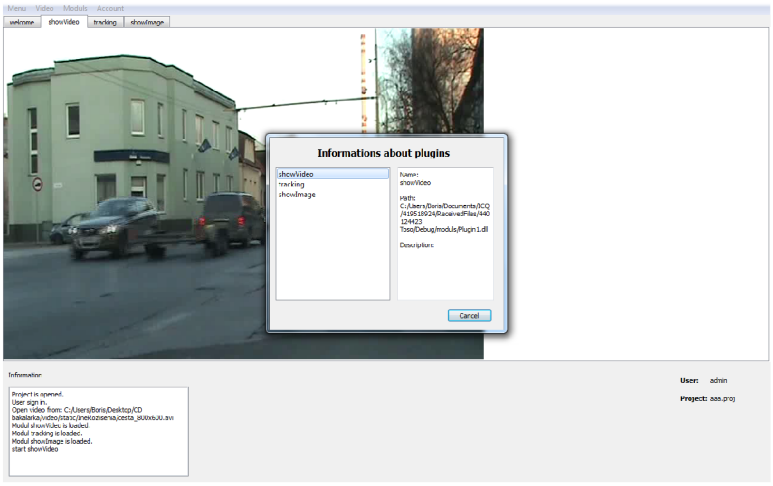
- login: admin heslo: 12345
- login: adv heslo: 12345
- login: std heslo: 12345

# Dodatok C

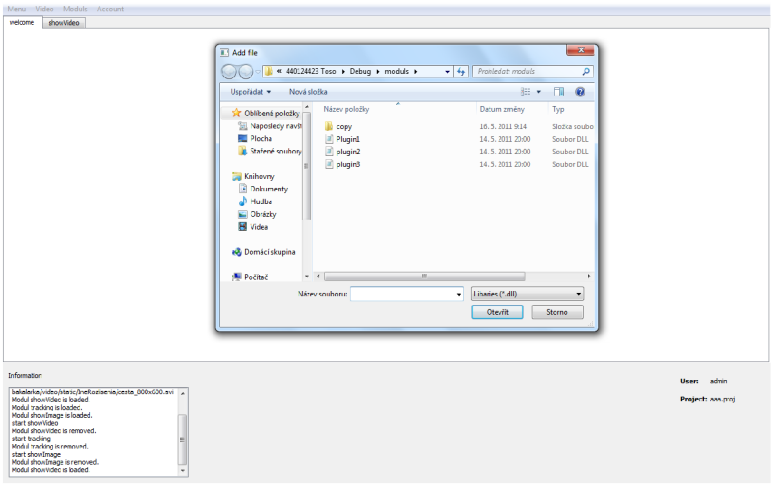
# Ukážky



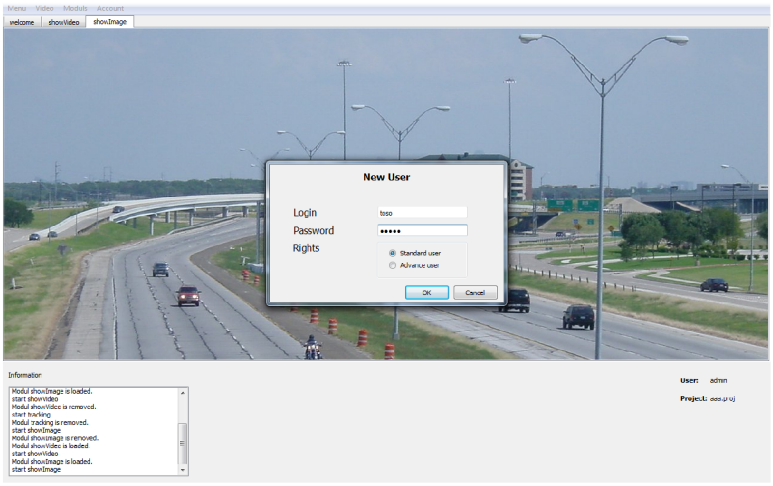

Obrázok C.1: Ukážka modulu na sledovanie objektov.



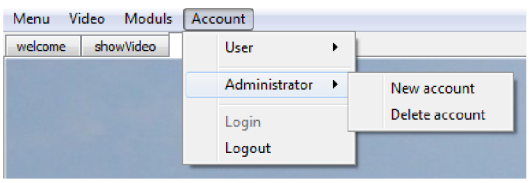

Obrázok C.2: Ukážka okna s informáciami o moduloch.




Obrázok C.3: Ukážka okna na pridávanie modulov.




Obrázok C.4: Ukážka okna na vytvorenie nového účtu.




Obrázok C.5: Ukážka panela s menu.