České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačové grafiky a interakce

Bakalářská práce

Semi-automatický 2D anotátor architektonických prvků

Renata Musilová

Vedoucí práce: Ing. David Sedláček

Studijní program: Softwarové technologie a management, Bakalářský

Obor: Web a multimedia

3. ledna 2013

Poděkování

Děkuji vedoucímu své práce Ing. Davidu Sedláčkovi za náměty, připomínky a cenné rady, které mi dal a za čas, který této práci věnoval. Dále děkuji své rodině, příteli a přátelům za podporu při studiu a také všem účastníkům testování za ochotu.

Prohlášení

Prohlašuji, že jsem práci vypracovala samostatně a použila jsem pouze podklady uvedené v přiloženém seznamu.
Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 3. 1. 2013 ……

Abstract

Thesis describes analysis, design and implementation of simple tool for 2D annotation of architectural elements. User defines set of architectural elements. Saved annotations can be displayed and edited. Tool can find annotations automatically.

Abstrakt

Práce se zabývá analýzou, návrhem a implementací jednoduchého nástroje pro anotaci 2D architektonických prvků. Uživatel si definuje množinu stavebních elementů sám. Vytvořené anotace lze později zobrazit a změnit. Součástí nástroje je také možnost automatické anotace.

Obsah

1 Úvod

Tato bakalářská práce je výsledkem mého úsilí získat praktické zkušenosti při vytváření softwarového díla od návrhu, přes design až po implementaci aplikace. Vybrané téma mě zaujalo možnostmi, které nabízí k prostudování a zpracování. Také existuje mnoho rozšíření, kterými se dá výsledná aplikace vylepšit a tím pokračovat v započaté práci.

1.1 Cíl bakalářské práce

Cílem této bakalářské práce je vytvoření semi-automatického nástroje určeného pro anotaci architektonických prvků historických budov ve 2D zobrazení. Množinu těchto prvků bude postupně definovat sám uživatel. Anotace bude uchovávána v XML včetně vstupního obrázku, což zjednoduší její následné zobrazení a úpravy.

Úvodní část je zaměřena na analýzu způsobů vhodných k implementaci semi-automatické anotace od předzpracování obrazu po klasifikaci dat. Následuje analýza a návrh implementace aplikace. V následující části je popis řešení, které využívá vybranou metodu nalezenou v analýze. Závěrečná část je věnována testování aplikace a zhodnocení kvality automatické anotace.

2 Analýza rozpoznávání obrazu

V této kapitole najdete nastínění možností předzpracování obrazu, popis strukturálního a příznakového rozpoznávání a porovnání některých algoritmů klasifikace, ze kterých jeden bude vybrán a naimplementován.

2.1 Předzpracování obrazu

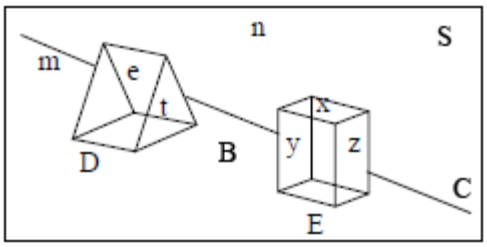
K předzpracování obrazu nás mohou vést dva důvody. Prvním je, že obraz je zkreslený například vlivem snímacího zařízení nebo nevhodných podmínek při jeho pořizováni. Pokud známe charakter těchto chyb a jejich vliv na zkreslení, můžeme pomocí korekcí chyby opravit. Druhým důvodem je snaha zjednodušit analýzu obrazů, například rozpoznávání a následnou klasifikaci. K předzpracování můžeme použít metody jasových transformací, geometrických transformací a další.

2.2 Strukturální rozpoznávání

Metody strukturálního rozpoznávání pracují s obrazy reprezentovanými pomocí množin základních popisných elementů – tzv. primitiv, a jejich vlastnostmi a vztahy mezi nimi - relacemi. Primitiva jsou nejjednodušší strukturální elementy obrazu. Jestliže jsou zvolena tak, že vystihují podstatné části objektu a relace mezi nimi vyjadřují důležité vztahy, jsou zachyceny podstatné strukturální vlastnosti obrazu. Relace jsou například prostorové nebo funkční a mohou být reprezentovány pomocí logických či matematických operací, nebo relačním grafem. Výhodou je, že kromě klasifikace do tříd umožňuje strukturální rozpoznávání zkoumat strukturu obrazu a popisovat jeho části a vztahy mezi nimi. Je využita podobnost mezi touto strukturou a gramatikou jazyka. Jako se věty skládají ze slov, slova ze slabik a slabiky z písmen, tak obrazy se skládají z jednodušších a jednodušších částí, až nakonec nejmenšími prvky jsou primitiva. Volba primitiv a relací mezi nimi je prvním krokem ve strukturálním rozpoznávání. Neexistuje na to žádná obecná metoda a tak záleží na zkušenosti konstruktéra a jeho znalosti úlohy. Při volbě by se měl snažit držet následujících požadavků, které jsou ale často protichůdné:

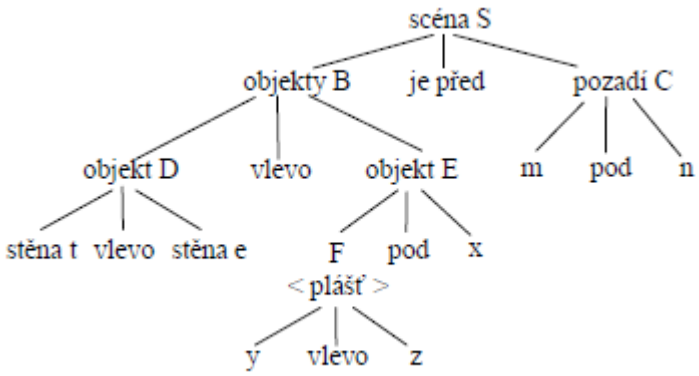
a) počet typů primitiv a relací by měl být co nejmenší.
b) primitiva by měla odpovídat přirozeným a výrazným strukturálním elementům popisovaného objektu,
c) určení primitiv a relací z naměřených dat by mělo být co nejjednodušší.

Předchozí seznam převzat z knihy [1]. Popis obrazu můžeme provést pomocí relačního grafu, nebo lépe pomocí stromové struktury. Ta nám umožňuje reprezentovat obrovské množství struktur v jedné třídě pomocí konečné množiny primitiv a jistých syntaktických pravidel.



Obrázek 1 - Jednoduchá scéna [2]

Například Obrázek 1 je tvořen pozadím a objekty složenými z jednotlivých primitiv. Jeho popis pomocí stromové struktury vypadá následovně, viz Obrázek 2.



Obrázek 2 - Stromová struktura [2]

2.2.1 Zhodnocení použitelnosti

Tento druh rozpoznávání není pro vytvoření automatické anotace vhodný. Předem nevíme, kolik a jakých prvků budeme anotovat ani jak budou vypadat. Nepotřebujeme tedy popsat celý obraz, ale jen určit, jestli něco je námi hledaný objekt nebo není.

2.3 Příznakové rozpoznávání

Metody příznakového rozpoznávání pracují s daty získanými přímo z fyzikálních vlastností pozorovaných objektů. Těmto číselným charakteristikám objektu se říká příznaky. Všechny tyto příznaky jsou nazývány vektorem příznaků. Prostor všech vektorů příznaků je nazýván příznakovým prostorem. [1] Abychom zjistili, ke které klasifikační třídě jednotlivé vektory náleží, použijeme jednu ze tří základních metod rozpoznávání - diskriminační funkci, kritérium minimální vzdálenosti nebo kritérium minimální chyby.

2.3.1 Diskriminační funkce

Při klasifikaci do R tříd se symbol ω_r přiřazený r-té třídě, kde $r = 1, \ldots, R$, nazývá indikátor třídy. Každému vektoru je jednoznačně (při deterministickém přiřazení) přiřazen indikátor třídy. Přiřazení se nazývá rozhodovací pravidlo a popisuje ho funkce d:

$$d(\mathbf{x}) = \omega \,,\ \omega \in \{\omega_1 \ldots \omega_n\}.$$

Pravidlo vymezuje R vzájemně disjunktních množin $\Re_R$, r = 1,…, R, v příznakovém prostoru $\Re$ definovaných vlastností

$$\Re_r = \{\mathbf{x} : d(\mathbf{x}) = \omega_r\}.$$

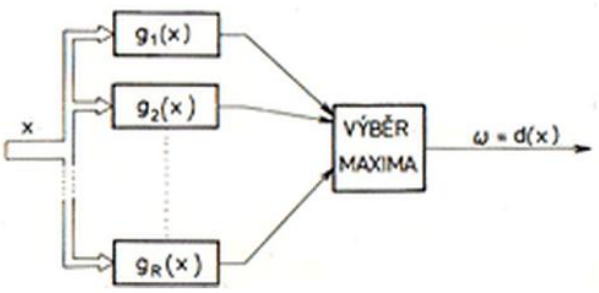
Rozhodovací pravidlo není definováno pro množinu vektorů $\mathbf{x} \in \Re$, které jsou hraničními body množin $\Re_R$ a $\Re_s$. Taková množina vektorů se nazývá rozdělující nadplochou mezi $\Re_R$ a $\Re_s$. Rozdělující nadplochy lze definovat pomocí soustavy R skalárních funkcí vektorového argumentu g_r (**x**), r= 1, …, R. Nazývají se diskriminační funkce a každá je přiřazena jedné z tříd. Diskriminační funkcí r-té třídy je každá taková funkce g_r (**x**) splňující nerovnost

$$g_r(\mathbf{x}) > g_s(\mathbf{x})$$

pro každý vektor $\mathbf{x} \in \Re_r$ a pro s =1, 2, …, R, $s \neq r$. [1] Z toho vyplývá, že vektor **x** přiřadíme do té třídy, jejíž diskriminační funkce bude maximální. Pouze pokud x leží na rozdělující ploše, nelze ho do žádné třídy jednoznačně přiřadit, protože rovnice rozdělujících nadploch mezi sousedními množinami $\Re_R$ a $\Re_s$ mají tvar

$$g_r(\mathbf{x}) = g_s(\mathbf{x}),$$

nemůžeme tedy vybrat maximum. Obrázek 3 znázorňuje, jak klasifikace probíhá.



Obrázek 3 - Klasifikace pomocí diskriminačních funkcí [1]

2.3.2 Kritérium minimální vzdálenosti

Každá klasifikační třída z R tříd musí mít vzorový vektor příznaků, tzv. etalon třídy. Etalony jednotlivých tříd se značí $\mathbf{v}_1$, $\mathbf{v}_2$, …, $\mathbf{v}_R$. Vektor příznaků klasifikovaného objektu **x** $\in \Re$ zařadíme do té třídy, jejíž etalon je objektu nejpodobnější. Což znamená, že v příznakovém prostoru má etalon třídy r od vektoru **x** nejmenší vzdálenost. Vzdálenost definujeme jako normu $\|v_r - \mathbf{x}\|$ a hledáme takovou třídu r, pro kterou platí

$$\|v_r - \mathbf{x}\| = \min_s \|v_s - \mathbf{x}\|, \qquad s = 1, 2, \ldots, R. \text{ [1]}$$

2.3.3 Kritérium minimální chyby

Někdy nelze jednoznačně určit, do které třídy vektor příznaků **x** přiřadit. Musíme tedy zvolit takové rozhodovací pravidlo, které minimalizuje ztráty způsobené chybnou

klasifikací. Předpokladem je, že známe všechny potřebné pravdivostní charakteristiky řešeného úkolu. Při klasifikaci do R tříd označíme indikátory tříd $\omega_1, \omega_2, ..., \omega_R$. Třídu, kam vektor **x** patří, označíme indikátorem ω. Protože neumíme rozhodnout do které třídy vektor **x** patří, je ω náhodná proměnná s možnými hodnotami $\omega_1, \omega_2, ..., \omega_R$ a s danými pravděpodobnostmi $P(\omega_1), P(\omega_2), ..., P(\omega_R)$, pro které platí

$$\sum_{r=1}^{R} P(\omega_r) = 1.$$

Hodnota $P(\omega_r)$ je předem danou pravděpodobností výskytu vektorů příznaků z třídy, která je dána indikátorem ω_r. Podle původního předpokladu známe všechny podmíněné hustoty pravděpodobnosti $p(x/\omega_i)$, které vyjadřují rozložení hodnot **x** v jednotlivých třídách daných indikátorem ω_i. Pravděpodobnost, že **x** patří do třídy, která má indikátor ω_r, je dána vztahem:

$$P(\omega_r / \mathbf{x}) = \frac{p(\mathbf{x}/\omega_r) P(\omega_r)}{p(x)},$$

kde

$$p(x) = \sum_{r=1}^{R} p(x/\omega_r) P(\omega_r)$$

je hustota rozložení vektorů příznaků **x** v příznakovém prostoru $\Re$ bez ohledu na třídu. Pravděpodobnost $P(\omega_r / \mathbf{x})$ je generovaná, aposteriorní, a vtah pro její výpočet se nazývá Bayesovým vztahem. Rozhodovací pravidlo je stanoveno pomocí aposteriorních pravděpodobností tak, že vektor **x** je zařazen do takové třídy *r*, pro kterou platí

$$P(\omega_r / \mathbf{x}) = \max_s P(\omega_r / \mathbf{x}).$$ [1]

2.4 Konkrétní algoritmy klasifikace

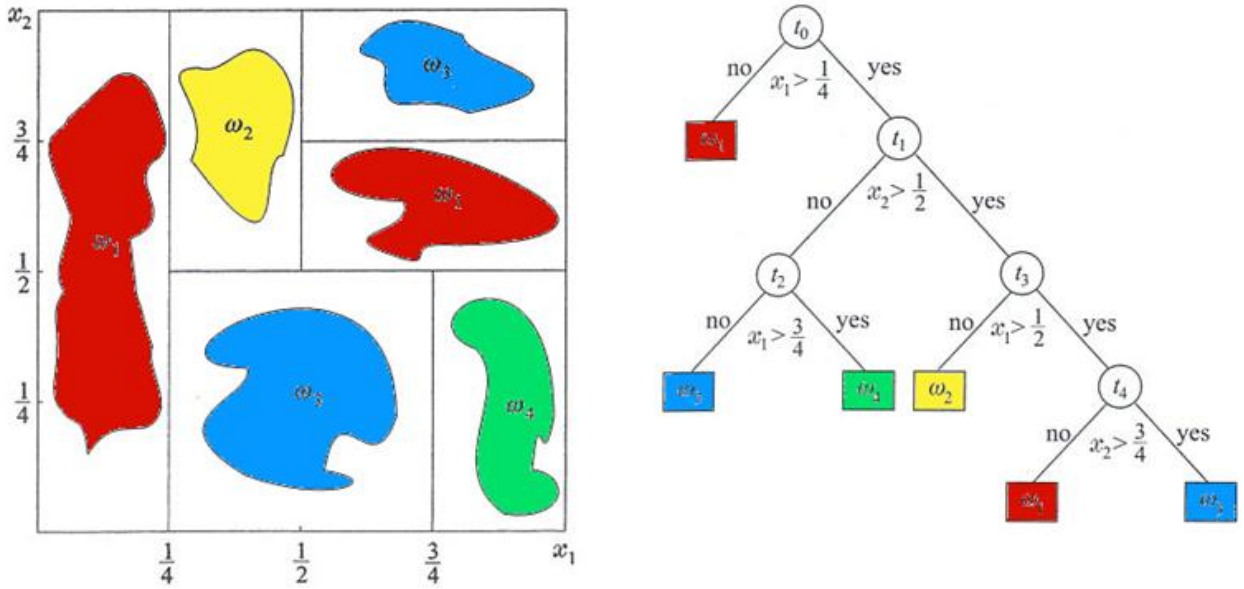
V následujících podkapitolách budou nastíněny některé konkrétní algoritmy klasifikace. Část z nich by bylo možné v implementaci anotátoru využít.

2.4.1 Markovské modely

Jedná se o metodu klasifikace závislé na kontextu. Nedělá se jedno rozhodnutí, ale posloupnost rozhodnutí, která jsou na sobě závislá. Obvykle se používá na analýzu pozorování měnících se v čase. Pro rozpoznávání v obraze se hodí jen při zvláštním uspořádání, například rozpoznávání registračních značek aut (x sloupce registrační značky, k znaky značky). Nehodí se tedy pro anotaci architektonických prvků, proto zde nebude více rozebírána. [3]

2.4.2 Rozhodovací stromy

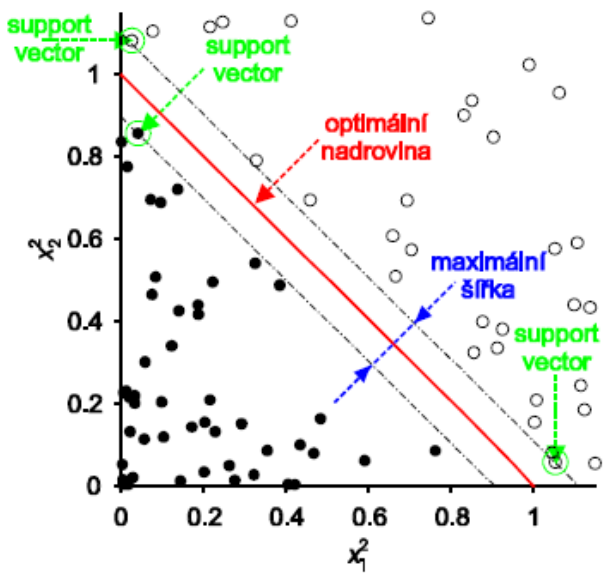
Mnoho úrovňový systém, kde jsou třídy postupně zamítány, do té doby, než je dosažena přijatelná třída. Pro příznakový vektor pomocí posloupnosti rozhodnutí ve stromové struktuře hledáme oblast, kam bude zařazen, viz Obrázek 4. Velikost stromu musí být dostatečně velká. Stromové schéma je vhodné pro velké množství tříd a nehodí se pro klasifikaci do dvou tříd. [4]



Obrázek 4 - Ukázka rozhodovacího stromu [4]

2.4.3 Support vector machines

Metoda strojového učení, která při klasifikaci hledá optimální rozdělující nadrovinu. Ta je optimální, když body leží v opačných poloprostorech co nejdál od nadroviny. Je popsána nejbližšími body, které se nazývají podpůrné vektory (angl. support vectors). Zároveň je žádoucí, aby šířka pásma, ve kterém se hranice mezi třídami může pohybovat kolmo k vlastní směrnici, aniž by se dotkla některého bodu, byla co největší. Tomuto pásmu se říká margin a je dobré ho maximalizovat, viz Obrázek 5.



Obrázek 5 - optimální oddělovací hranice [5]

Optimální lineární oddělovač se hledá pomocí metody kvadratického programování. Primární úlohou je najít optimální nadrovinu $\langle w, x\rangle + b = 0$. Trénovací množina je $\{(x_1, y_1), (x_2, y_2), \dots, (x_l, y_l)\}$, kde $x_i \in \mathbb{R}^n$ je trénovací vzor a $y_i \in \{+1, -1\}$ je jeho identifikátor třídy. Abychom zjistili vektor w a práh b, určující hledanou nadrovinu, musíme vyřešit optimalizační úlohu

$$(w, b) = argmin \frac{1}{2}\|w\|^2$$

za podmínek

$$\langle w, x_i\rangle + b \geq +1, \qquad y_i = +1,$$
$$\langle w, x_i\rangle + b \leq -1, y_i = -1. \text{ [6]}$$

Tuto úlohu můžeme převést na duální úlohu. Jejím cílem je nalézt čísla $\alpha_i, i = 1, \dots, l$, která jsou řešením

$$\vec{\alpha} = argmax\left(\sum_{i=1}^{l} \alpha_i - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \langle x_i, x_j\rangle\right),$$

za podmínky

$$\alpha_i \geq 0, i = 1,2, \dots l,$$

$$\sum_{i=1}^{l} \alpha_i y_i = 0\,.$$

Výhodou duální úlohy je, že datové body vstupují ve formě skalárního součinu jednotlivých dvojic. Nakonec je nutné nalézt optimální proměnné primární úlohy w a b pomocí rovnic

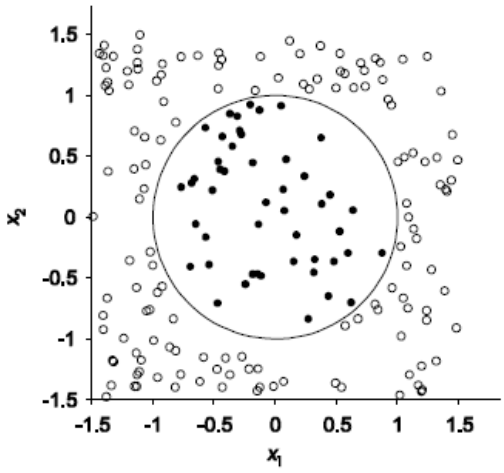
$$w = \sum_{i=1}^{l} \alpha_i y_i x_i \ ,$$

$$b = \frac{1}{|I|} \sum_{i \in I} \left(y_1 - \sum_{j=1}^{l} \alpha_j y_j \langle x_j , x_i \rangle \right),$$

kde

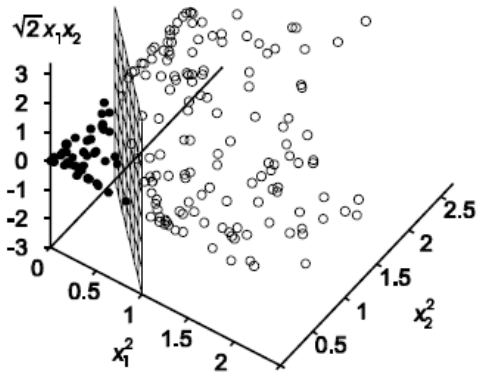
$$I = \{i : 0 < \alpha_i\}$$

je množina indexů trénovacích bodů, které leží ve vzdálenosti +1 nebo -1 od nadroviny. Je to tedy množina podpůrných vektorů. [6]



Obrázek 6 - Dvourozměrný prostor s nelineární hranicí [5]

Aby se metoda mohla aplikovat i na data, mezi kterými není lineární oddělovací hranice, viz Obrázek 6, využívá se jádrové funkce (angl. kernel function). S jejím využitím vypočítáme skalární součin dvojice vstupních dat v nějakém odpovídajícím vícerozměrném prostoru.

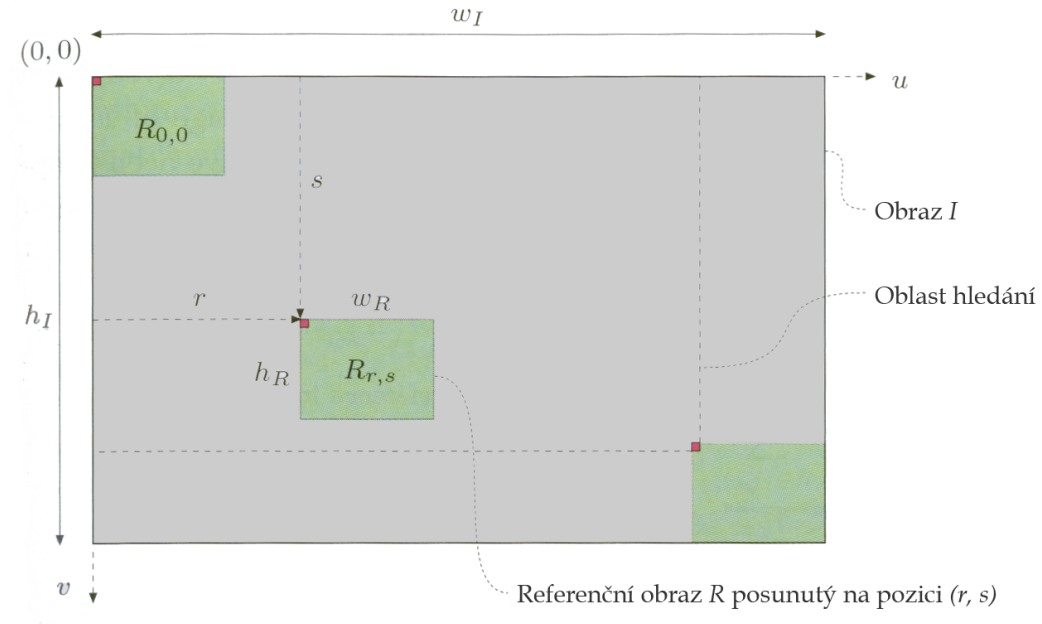


Obrázek 7- Třírozměrný prostor s lineární hranicí [5]

Tím získáme rovnici rozdělující nadroviny pro data, která v původním prostoru nešla lineárně oddělit, viz Obrázek 7, bez nutnosti počítání úplného seznamu atributů pro každý datový bod. [5] Existuje mnoho jádrových funkcí odpovídajících vícerozměrným prostorům, které zde ale nebudou popsány. Více se dozvíte na příklad v [7].

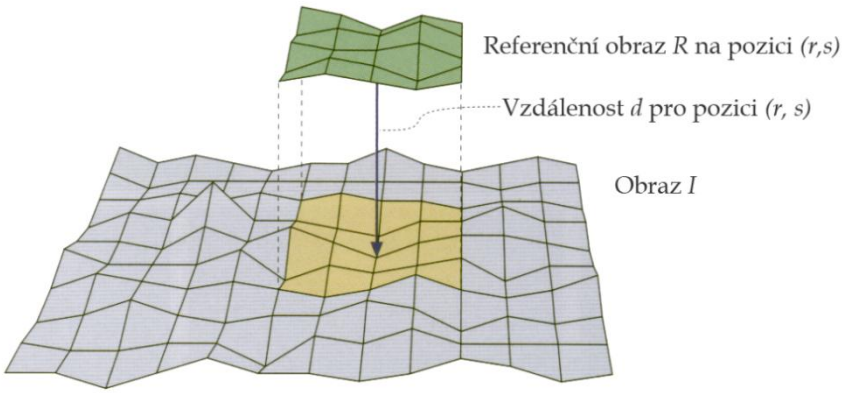
2.4.4 Registrace obrazu na základě intenzity

Pomocí registrace obrazu hledáme v obraze I vzdálenost posunutí (r, s) takovou, že je podobnost s referenčním obrazem R maximální. Referenční obraz R je posouván napříč obrazem I a v každém bodě jsou porovnávány rozdíly v intenzitě obrazů, viz Obrázek 8. Referenčními body jsou počátky porovnávaných obrazů.



Obrázek 8 - Nákres posunutí referenčního obrazu v obrazu I [8]

Je nutné stanovit toleranci, kdy je pro nás část obrazu I dostatečně podobná s referenčním obrazem R. Pro určení podobnosti spočítáme pro každé posunutí (r, s) vzdálenost $d(r, s)$ mezi referenčním obrazem R a odpovídající částí obrazu I, viz Obrázek 9.



Obrázek 9 - Vzdálenost referenčního obrazu R od obrazu I [8]

Pro jednoduché měření vzdálenosti mezi dvou-dimenzionálními obrazy se používají následující způsoby:

Suma absolutních rozdílů

$$d_A(r,s) = \sum_{(i,j)\in R} |I(r+i,s+j) - R(i,j)|,$$

Maximální rozdíl

$$d_M(r,s) = \max_{(i,j)\in R} |I(r+i,s+j) - R(i,j)|,$$

Suma čtverců rozdílů

$$d_A(r,s) = \left[\sum_{(i,j)\in R} |I(r+i,s+j) - R(i,j)|^2\right]^{\frac{1}{2}}.$$

Ve vzorcích pro i a j platí, že jsou to souřadnice obrazu R: $\{(i,j) \mid 0 \le i < w_R,\ 0 \le j < h_R\}$. Tyto metody pro výpočet vzdálenosti jsou citlivé na změny jasu a kontrastu v obraze. Pro získání vzdálenosti invariantní k lokálním změnám jasu a kontrastu se využívá korelace. Předchozí část kapitoly 2.4.4 je volně převzata z [8].

Normalizovaná vzájemná korelace

Koeficient $r(x,y)$ udává, do jaké míry jsou hodnoty x_i a y_i z množiny $\{(x_1,y_1),\dots,(x_1,y_1)\}$ svázány lineární funkcí. Čím více jsou svázány lineární funkcí s kladnou derivací, tím je $r(x,y)$ blíže 1 až do $r(x,y) = 1$ při přesném svázání. Se zápornou derivací se blíží -1 až do $r(x,y) = -1$. Následující výpočet je převzat z [9]:

1. Posuneme čísla $\{x_1,\dots,x_n\}$ tak, aby měla nulovou střední hodnotu. Nová čísla označíme jako

$$x'_i = x_i - \overline{x}, \qquad i = 1,\dots,n.$$

Totéž uděláme s čísly $\{y_1,\dots,y_n\}$.

2. Čísla $\{x'_1,\dots,x'_n\}$ vydělíme všechna stejnou konstantou tak, aby nová čísla měla jednotkovou varianci:

$$x''_i = \frac{x'_i}{\sqrt{\overline{x'^2}}}, \qquad i = 1,\dots,n.$$

Totéž uděláme s $\{x'_1,\dots,x'_n\}$.

3. Pomocí normalizovaných čísel x''_i a y''_i je lineární korelační koeficient dán jednoduchým vztahem

$$r(x,y) = \overline{x''\,y''} = \frac{1}{n}\sum_{i=1}^{n} x''_i\, y''_i.$$

2.4.5 AdaBoost

Metoda rozpoznávání tzv. s učitelem, která klasifikuje do dvou tříd. Název AdaBoost vznikl zkrácením výrazu adaptive boosting. Patří do skupiny boosting metod, které ze skupiny slabých klasifikátorů vytvářejí silný klasifikátor. Klasifikační přesnost slabého klasifikátoru musí být větší než 50%. Jediným požadavkem na slabý klasifikátor tedy je, aby byl úspěšnější než pouhé hádání. Přidáváním dalších slabých klasifikátorů se stejnou klasifikační vlastností nakonec vznikne silný klasifikátor s vysokou úspěšností. Slovo adaptive v názvu značí, že po každém přidání slabého klasifikátoru se změní váhy vzorků tak, aby špatně zařazené vzorky měly vyšší váhu. Tyto vzorky více ovlivňují výběr dalšího klasifikátoru a metoda je přesnější.

Vstup: $(x_1, y_1), \dots, (x_m, y_m);\ x_i \in X, y_i \in \{+1, -1\}$

Inicializuj váhy $D_1(i) = \frac{1}{m}$

Pro $t = 1, \dots, T$:

1. Najdi

$$h_t = \arg min\, \epsilon_j\ ;\ \epsilon_j = \sum_{i=1}^{m} D_t(i) I\left[y_i \neq h_j(x_i)\right]$$

2. Když

$$\epsilon_t \geq \frac{1}{2}$$

skonči

3. Spočítej váhu klasifikátoru

$$\alpha_t = \frac{1}{2} log\left(\frac{1 - \epsilon_t}{\epsilon_t}\right)$$

4. Aktualizuj

$$D_{t+1}(i) = \frac{D_t(i) exp\left(-\alpha_t y_i h_t(x_i)\right)}{Z_t}$$

kde

$$Z_t = \sum_{i=1}^{m} D_t(i)\, exp\left(-\alpha_t y_i h_t(x_i)\right)$$

Výsledný klasifikátor:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

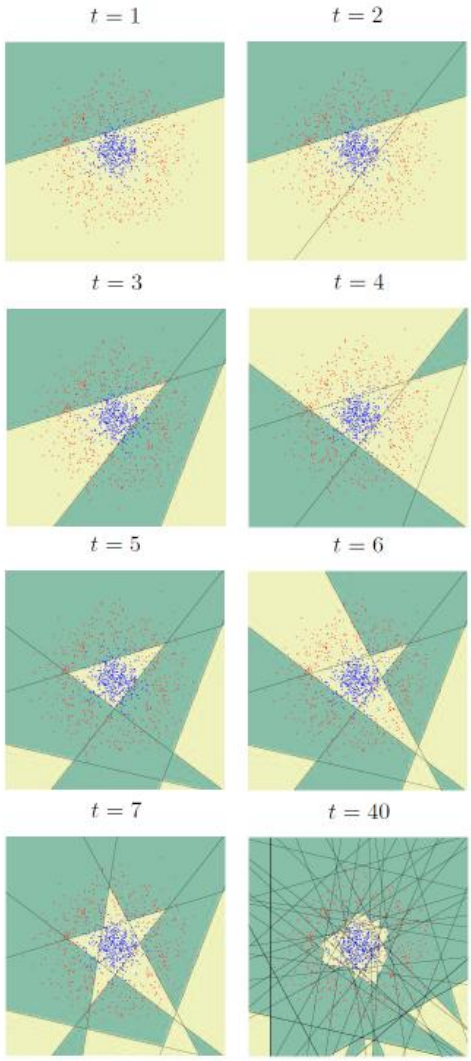
Algoritmus 1 – AdaBoost [10]

Vstupem klasifikátoru je sada trénovacích dat $\{(x_1, y_1), \dots, (x_m, y_m)\}$, kde $x_i \in X$ jsou vektory příznaků a $y_i \in Y$ je jejich ohodnocení, $Y = \{+1, -1\}$. Protože AdaBoost

klasifikuje do dvou tříd, můžeme třídě s hledanými objekty přiřadit hodnotu +1 a třídě s pozadím hodnotu -1. Trénovací množina je ovážena váhami D_t, které jsou na začátku rozděleny rovnoměrně. Po každém přidání slabého klasifikátoru je distribuce vah přepočítána. V každém kroku učení je jeden slabý klasifikátor h_t z množiny všech možných klasifikátorů $\mathcal{H}$ přidán do silného klasifikátoru. Výběr je v každém kroku realizován tak, aby byl minimalizován horní odhad chyby klasifikátoru, viz krok 1 v Algoritmus 1. Výraz $I[y_i \neq h_j(x_i)]$ vrací 1 pokud $y_i \neq h_j(x_i)$ a 0 pokud $y_i = h_j(x_i)$. Chyba klasifikátoru ϵ_t je rovna součtu vah chybně klasifikovaných vzorků. Krok 2 zaručuje, že bude použit jen dostatečně úspěšný slabý klasifikátor. Výpočet váhy klasifikátoru spolu s výběrem příznaku nám zajistí, že bude minimalizován horní odhad chyby výsledného silného klasifikátoru

$$\epsilon_{tr}(H) \leq \prod_{t=1}^{T} Z_t = \frac{1}{2^T} \prod_{t=1}^{T} \sqrt{\epsilon_t(1-\epsilon_t)}.$$

Aktualizace vah v kroku 4 způsobí, že v následujícím kroku bude hledán takový slabý klasifikátor, který bude muset lépe klasifikovat špatně klasifikovaná měření.



Obrázek 10- Vizualizace Adaboost [11]

Výhodou AdaBoostu je, že lineární kombinací slabých klasifikátorů získáme silný klasifikátor, který ale není lineární, viz Obrázek 10. Dokud budeme nacházet slabé klasifikátory s úspěšností klasifikace větší než 50%, bude chyba silného klasifikátoru exponenciálně klesat a v limitě se bude blížit nule. Nemá tedy tendenci se přetrénovat, jako se to může stát u jiných metod. Nevýhodou je malá odolnost vůči šumu v datech.

Haarovy příznaky

V učícím algoritmu AdaBoost mohou být jako slabé klasifikátory použity jakékoliv funkce, které dokáží rozhodovat lépe než náhodně. Pro detekci objektů v obraze se velmi často používají tzv. Haarovy příznaky, které budou použity i v této práci. Existuje několik Haarových příznaků, jejichž typ je dán tvarem tzv. konvolučního jádra, viz Obrázek 11.

Obrázek 11 – Příklady tvarů Haarových příznaků

AdaBoost vygeneruje množinu příznaků v obraze na všech pozicích a ve všech velikostech. Použije je jako klasifikátory a vybírá z nich ty, které mají nejmenší chybu. Každý příznak je definován pozicí v obraze a velikostí. V praxi se nemění velikost příznaku, ale samotný obraz.

Jeden příznak v obraze se pak získá funkcí

$$f(x) = \sum_{w \in W} x(w) - \sum_{b \in B} x(b)\,,$$

kde x je vzorek dat, W je množina pixelů příslušející k bílé a B k černé oblasti příznaku. Je to tedy rozdíl pixelů pod černou a bílou oblastí příznaku. Tyto sumy lze snadno získat z integrálního obrazu. Slabý klasifikátor obsahuje kromě obrazového příznaku i klasifikační práh θ a polaritu p. Práh je hodnota, podle které se rozhoduje o klasifikaci do jedné ze dvou tříd. Polarita ovlivňuje význam klasifikovaných tříd a obrácením znaménka nerovnosti ho může prohodit.

Integrální obraz

Tato reprezentace obrazu obsahuje v každém pixelu hodnotu A, která je součtem všech pixelů v obdélníku vlevo nahoru od A. V pravém dolním rohu tedy najdeme hodnotu součtu všech pixelů v obraze. Z integrálního obrazu můžeme získat součet pixelů v jakémkoliv obdélníku v obraze v konstantním čase. Stačí nám znát rohové pixely obdélníku.

3 Analýza a návrh řešení

3.1 Funkční požadavky

Kapitola předkládá požadavky kladené na aplikaci po funkční stránce, jaké chování se od aplikace vyžaduje. Požadavky jsou členěny do větších celků.

3.1.1 Správa anotačních kategorií

Aplikace umožní uživateli vytvářet, rušit a přejmenovávat anotační kategorie a jejich podkategorie. Tím si sám definuje množinu stavebních elementů a anotačních atributů.

3.1.2 Uchování v hierarchické struktuře

Veškeré údaje potřebné pro znovu zobrazení anotací musí být uchovávány v hierarchické struktuře ve formátu xml. Tuto strukturu bude aplikace schopna rozpoznat a informace v ní obsažené znovu použít například pro zobrazení uložených anotací včetně anotační kategorie.

3.1.3 Správa anotací

Anotace bude možné jak ukládat, tak zobrazit a mazat. Bude to možné i po zavření aplikace a opětovném načtení informací z xml souboru. Po smazání anotace se smaže i v xml souboru.

3.1.4 Automatická anotace

Aplikace bude schopná sama nalézt jednoduché anotace architektonických prvků. Pro tuto možnost jí uživatel musí poskytnout vzorovou anotaci, kterou bude aplikace v obraze hledat. Takto nalezené anotace se řádně uloží stejným způsobem, jakým se bude ukládat anotace uložená uživatelem.

3.2 Nefunkční požadavky

Nefunkční požadavky na aplikaci a obecné požadavky byly následující:

1) Přehlednost grafického rozhraní – všechny důležité prvky jsou vidět,
2) Uživatelská příručka pro seznámení s chodem a funkcemi aplikace,
3) Operační systém – Windows XP a vyšší,
4) Pouze jedna role – uživatel,
5) Java Runtime Environment verze 1.7 a vyšší.

3.3 Případy užití

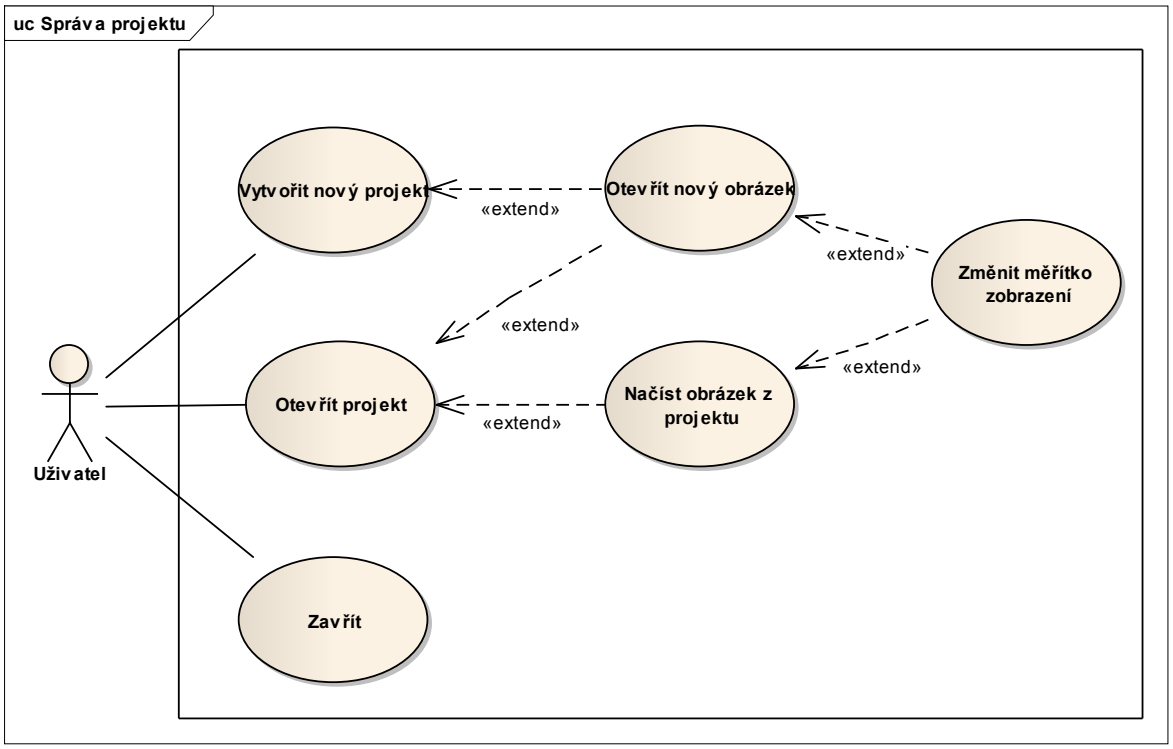
V následujících kapitolách budou popsány případy užití aplikace. Vhledem k přehlednosti jsou rozděleny do čtyř kapitol a diagramů dle svého účelu a každá kapitola kromě diagramu obsahuje popis jeho jednotlivých částí. V aplikaci není potřeba zohledňovat více

než jednu uživatelskou roli, která je nazvána uživatel a je ve všech diagramech případů užití. Uživatelem je myšlena osoba, která s aplikací pracuje.

3.3.1 Správa projektu

Uživatel musí vytvořit nebo otevřít projekt, aby mohl s programem dále pracovat. Obrázek 12 zobrazuje, jak na sebe jednotlivé případy užití navazují. Při vytvoření nového projektu je projekt vytvořen v aplikaci a zároveň se na lokálním úložišti vytvoří xml soubor obsahující informace o projektu. Pomocí tohoto souboru je později možné otevřít již existující projekt a použít informace v souboru uložené, například se načte strom anotačních kategorií. Další možností, co v aplikaci dělat, je zavřít ji. Je nutné, aby před zavřením bylo zajištěno uložení všech informací, které nemají být ztraceny, do xml souboru.

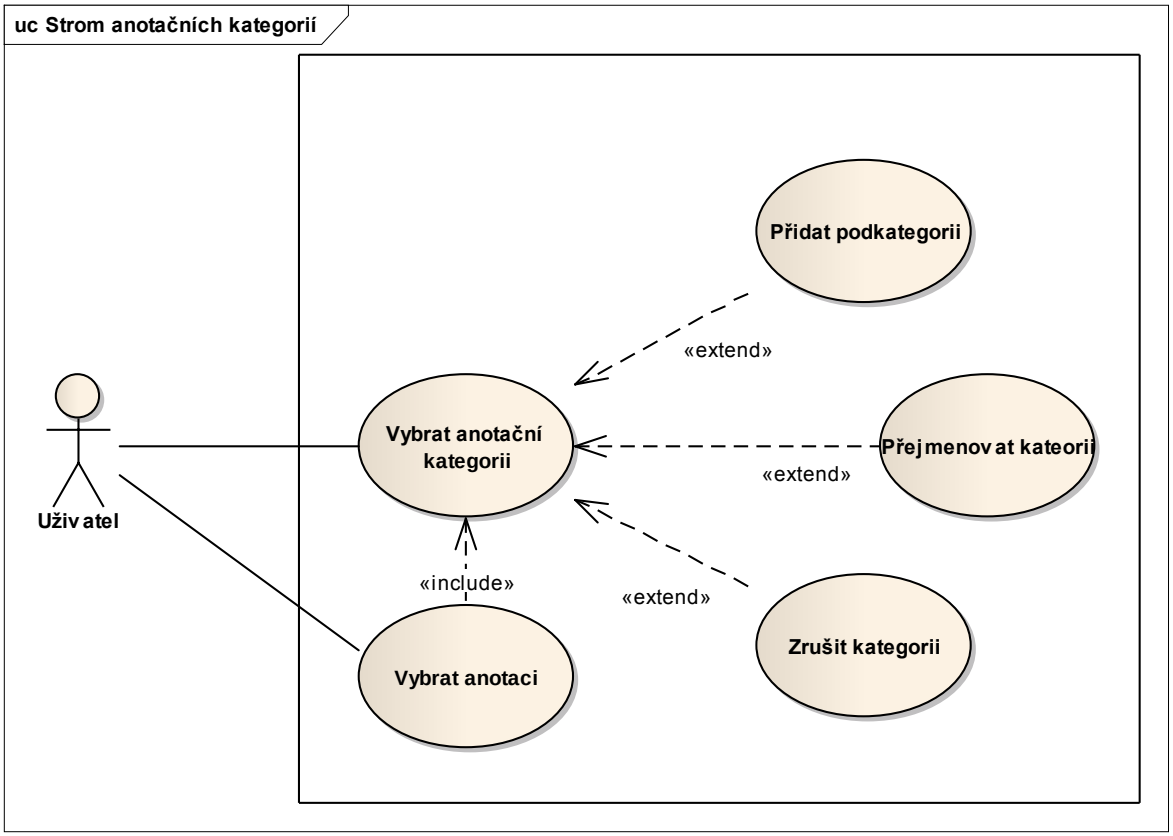
V projektu je potřeba otevřít obrázek, který má být anotován. Způsoby anotace a související případy užití budou popsány v následujících kapitolách. Obrázek se vybírá z lokálního úložiště a je zobrazen v aplikaci. Cesta k otevřenému obrázku je pro potřeby anotace uložena do xml souboru projektu. Díky tomu je možné otevřít obrázek, který v projektu již byl otevřený, bez toho, aby v něm byla uložena anotace. Při načítání obrázku z projektu je prohledán xml soubor a nabídnuty obsažené obrázky. Obrázek je zobrazen v aplikaci stejně jako nový obrázek, ale navíc jsou z xml souboru načteny anotace, které se v něm nachází a jsou zobrazeny v seznamu anotací i obrázku samotném. Pokud je zobrazen obrázek, může uživatel měnit měřítko zobrazení. Obrázek včetně anotací v něm zobrazených se zmenší na požadovanou velikost vybranou ze seznamu.



Obrázek 12 - Případy užití - správa projektu

3.3.2 Strom anotačních kategorií

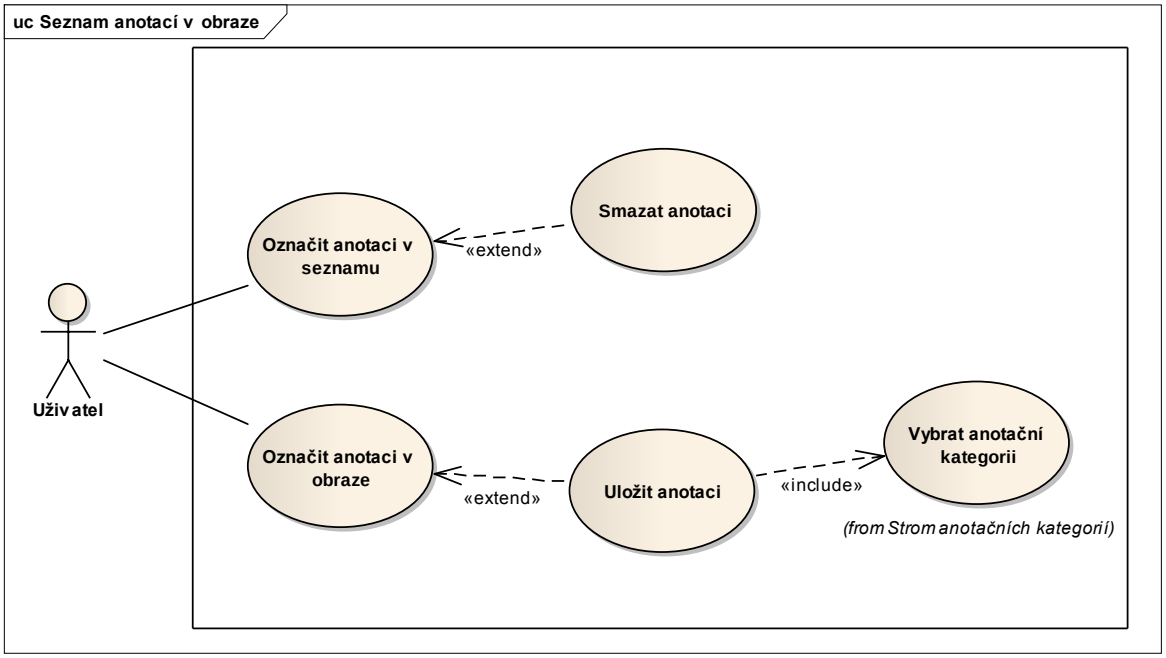
Jakmile je vytvořen nebo otevřen projekt, je možné spravovat strom anotačních kategorií, viz Obrázek 13. Tento strom se vždy váže k projektu, a proto se v aplikaci vytvoří po otevření projektu z xml souboru. Dále ho uživatel upravuje sám. Po vybrání anotační kategorie je několik možností. Aby byla přidána podkategorie, musí uživatel zadat název takový, jaký se ještě v úrovni, nižší než je vybraná kategorie, nevyskytuje. To samé platí i při přejmenování kategorie, na stejné úrovni nesmí být dvě kategorie se stejným názvem. Pokud uživatel zvolí možnost zrušit kategorii, budou zrušeni i všichni její potomci, včetně všech anotací, které do těchto kategorií patřily. Všechny předchozí změny jsou provedeny i v xml souboru. Uživatel také může ve stromě vybrat jednotlivé anotace. Tímto se kategorie, do které patří, stane vybranou kategorií a je možné přidat jí podkategorii, či jí přejmenovat nebo zrušit. Navíc je zobrazen náhled této anotace a, jak bude popsáno v kapitole 3.3.4, bude možné ji pomocí automatické anotace hledat v otevřeném obrázku.



Obrázek 13 - Případy užití - strom anotačních kategorií

3.3.3 Seznam anotací v obraze

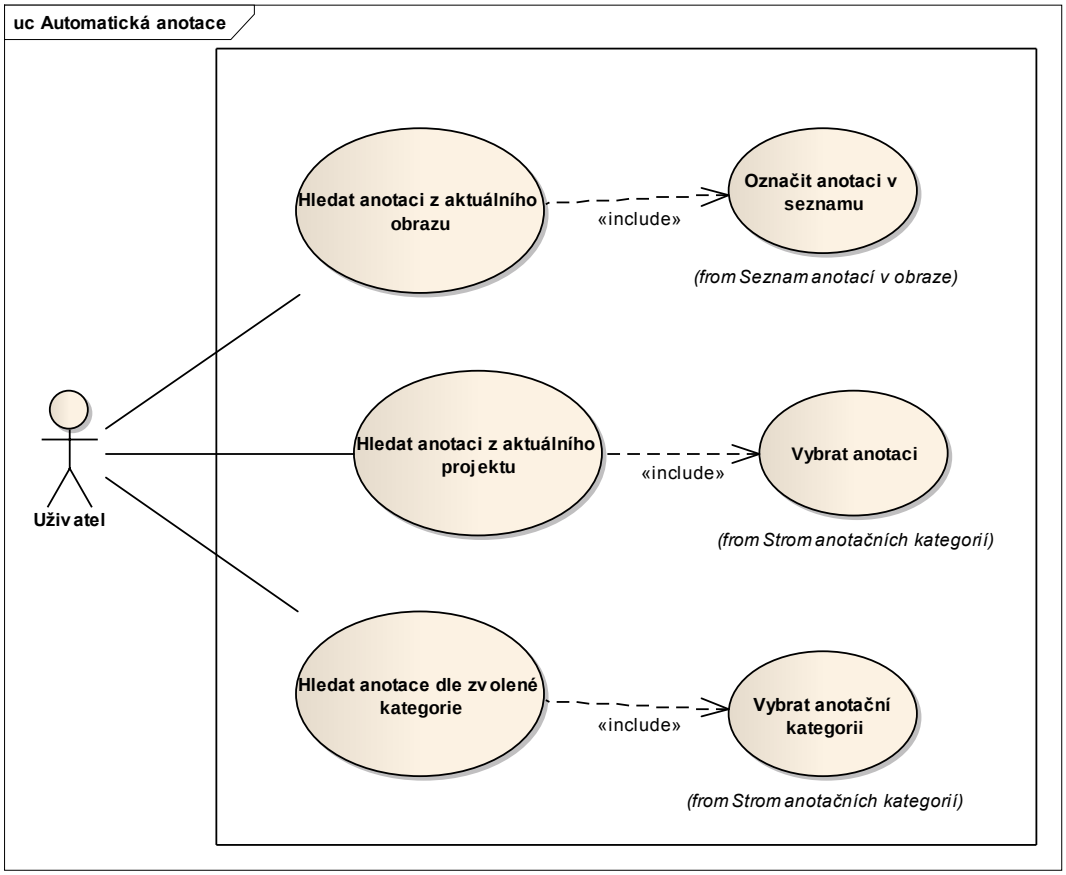
Obrázek 14 zobrazuje diagram případů užití spojených se seznamem anotací v aktuálním obrázku. Pokud uživatel označí anotaci v seznamu anotací, zvýrazní se v obrázku zobrazeném v aplikaci. Pokud chce, může uživatel tuto anotaci smazat. Tím se smaže ze seznamu, z obrázku i z xml souboru. Tažením myši lze v obrázku označit anotaci. Pokud není uložena, je při dalším označení anotace ta předchozí zapomenuta. Aby uživatel mohl anotaci uložit, je třeba mít vybranou vhodnou anotační kategorii. Po uložení se aktualizuje strom kategorií i seznam anotací v obraze a zobrazení anotací v obrázku. Anotace je přidána také do xml souboru.



Obrázek 14 - Případy užití - seznam anotací v obraze

3.3.4 Automatická anotace

Uživatel má několik možností automatické anotace, jsou to hledání anotace z aktuálního obrazu, hledání anotace z aktuálního projektu a hledání anotace dle zvolené kategorie, viz Obrázek 15.

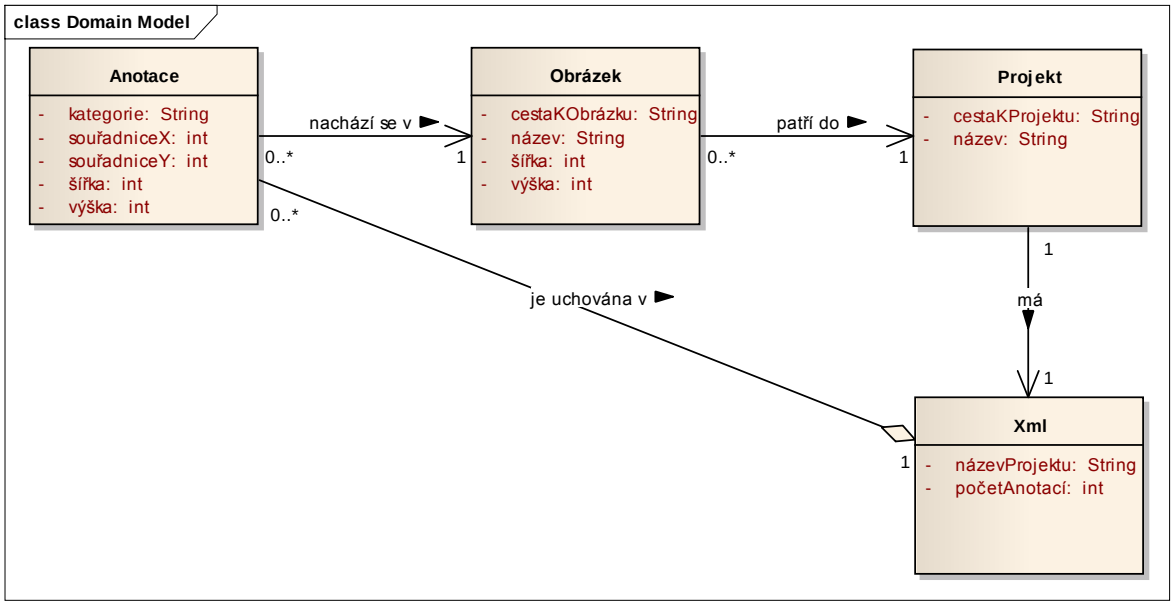


Obrázek 15 - Případy užití - automatická anotace

Když je označena anotace v seznamu anotací, lze ji hledat v aktuálním obrázku. Aplikace se sama postará o nalezení totožných architektonických prvků. Tyto označí v obrázku, uloží do stejné kategorie, jako byla hledaná anotace a aktualizuje strom kategorií a seznam anotací. Ukládané anotace zapíše i do xml souboru. Pokud uživatel vybere anotaci ve stromě anotačních kategorií, může tuto anotaci nechat vyhledat aplikací stejně, jako v předchozím případě. Při hledání anotací dle zvolené anotační kategorie aplikace vybere jednu anotaci a projde všechny podkategorie a z každé vezme jednu další anotaci. Všechny takto vybrané anotace poté hledá v obrázku. Jednotlivé nalezené anotace uloží do patřičných kategorií do xml souboru i aplikace a aktualizuje strom kategorií, seznam anotací a anotace zobrazené v obrázku.

3.4 Doménový model

Obrázek 16 znázorňuje vztah mezi důležitými entitami aplikace. Při vytvoření projektu se zároveň vytvoří i entita Xml, která bude vše ukládat do xml souboru. Pokud v projektu otevřeme obrázek, je tento obrázek přiřazen k projektu. V projektu může existovat více obrázků. V každém obrázku můžeme vytvářet libovolný počet anotací a každá anotace patří do právě jednoho obrázku. Xml uchovává množinu všech anotací obsažených v aktuálním obrázku.



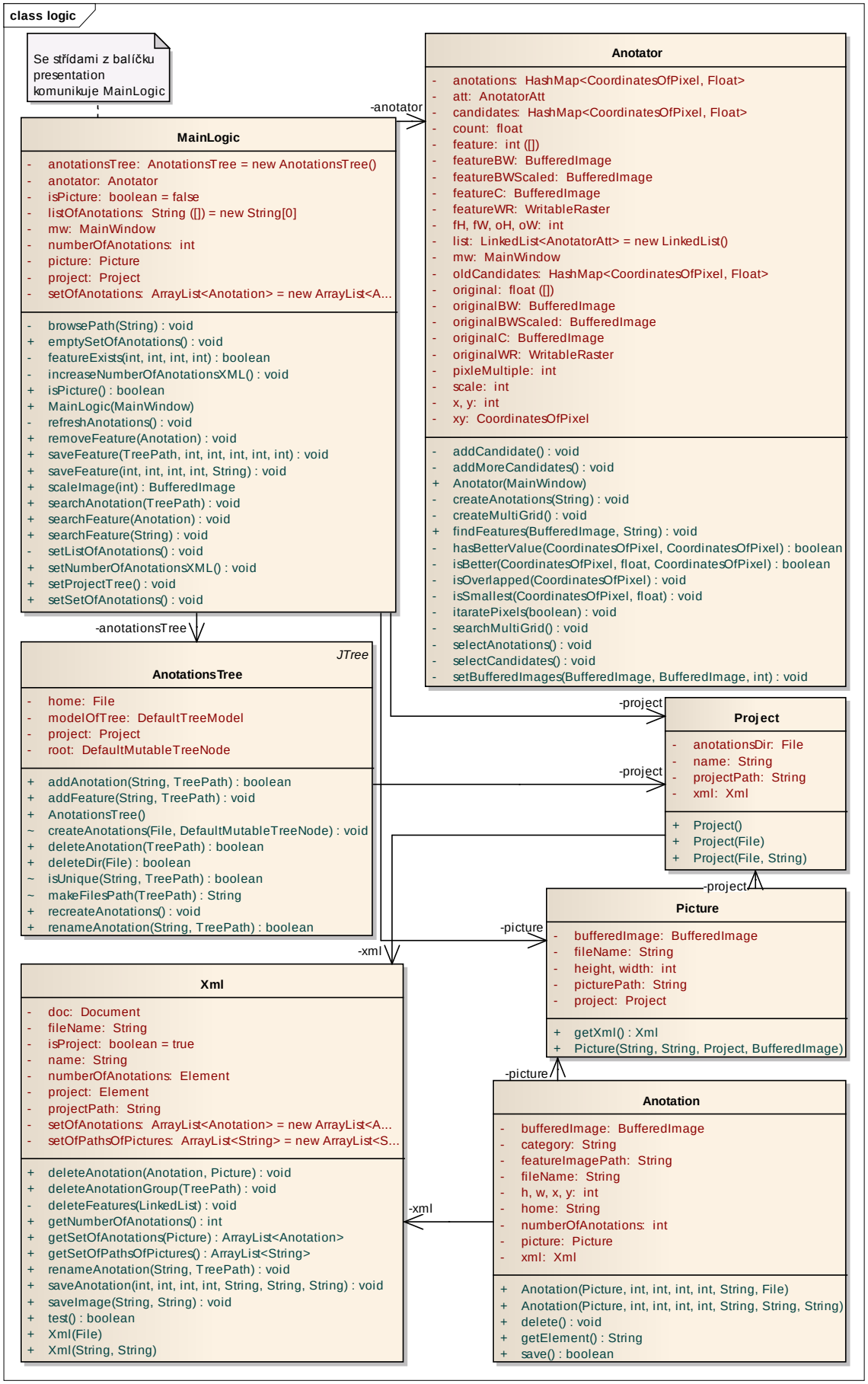
Obrázek 16 - Doménový model

4 Realizace

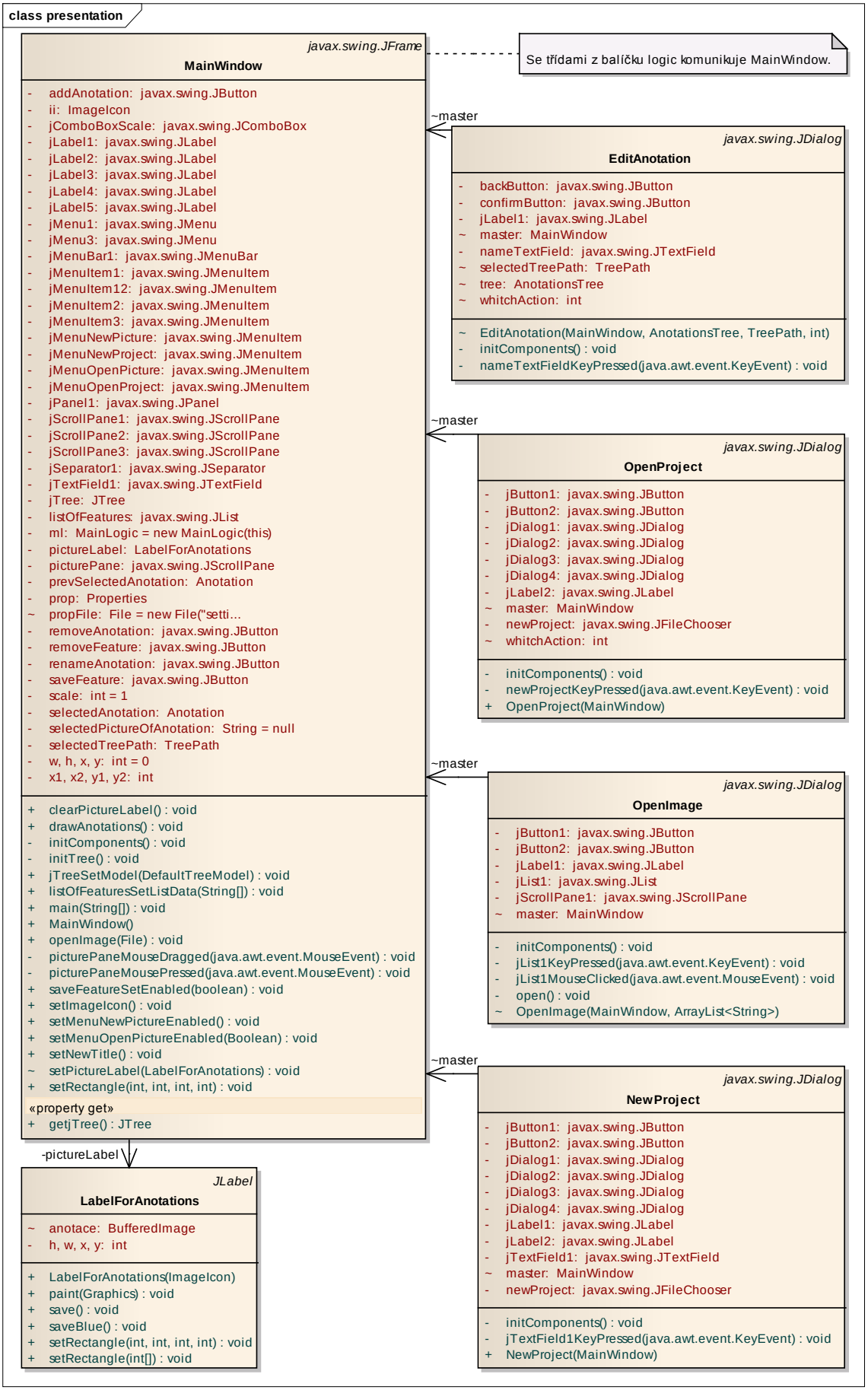
Kapitola se zabývá implementací aplikace. Bude popsán výběr programovacího jazyka a použitých technologií. Dále bude popsána realizace dílčích problémů a jejich řešení, včetně vysvětlení fungování programu.

Po vyjasnění návrhu aplikace bylo nutné vybrat vhodný programovací jazyk. Po přihlédnutí k jednotlivým požadavkům byla vybrána Java. Její vlastní knihovny postačí na pokrytí všech požadovaných funkcí. Pro vývoj byl použit JDKverze 1.6.

Aplikace je rozdělena do dvou balíčků. Balíček logic se stará o logiku aplikace, tedy práci s projekty, obrázky a anotacemi a propojení s informacemi uloženými v xml souboru projektu. Balíček presentation se stará o zobrazení uživatelského rozhraní a ovládání programu uživatelem. V následujících diagramech tříd je vidět, jak byla aplikace implementována z hlediska tříd. Obrázek 17 - Diagram tříd - balíček logic je diagram balíčku logic. Obrázek 18 je diagram balíčku presentation. Tyto dva balíčky spolu komunikují prostřednictvím tříd presentation.MainWindow a logic.MainLogic. V diagramech jsou vynechány některé metody tzpu getter, setter a actionListener tlačítek a položek menu.



Obrázek 17 - Diagram tříd - balíček logic



Obrázek 18 – Diagram tříd – balíček presentation

4.1 Logika aplikace

Během návrhu byly základními entitami aplikace prohlášeny entity projekt, obrázek, anotace a xml. Pro každou z entit vznikla v aplikaci samostatná třída. V následujících podkapitolách jsou tyto třídy, včetně několika dalších, potřebných k chodu aplikace, blíže popsány.

4.1.1 Hlavní logika

Třída `MainLogic.java` obsahuje hlavní logiku aplikace. Zprostředkovává komunikaci mezi grafickým rozhraním a logickou částí aplikace.

4.1.2 Projekt

Poté, co uživatel vytvoří nový projekt a jeho název projde validací, vytvoří se nová instance třídy `Project.java`. Odkaz na tuto instanci je uložen do proměnné třídy `MainLogic.java`. Je vytvořena nová instance třídy `Xml.java`, její funkcionality jsou popsány v kapitole 4.1.5. Je smazán seznam anotací, nastaven počet anotací v projektu a vytvořena nová instance třídy `AnotationsTree.java`.

Pokud uživatel otevírá existující projekt, je nejdříve zkontrolován soubor xml, zda obsahuje prvky projektu. Když projde validací, provede se vše z předchozího odstavce. Navíc se vytvoří model stromu anotačních kategorií. Rekurzivním voláním metody `createAnotations (File dir, DefaultMutableTreeNode parent)` ve třídě `AnotationsTree.java` se prochází všechny složky ve složce anotations v projektovém adresáři a přidávají se do modelu, včetně obsažených souborů anotací.

4.1.3 Obrázek

Obrázek lze otevřít pouze, pokud je otevřený projekt a to buď nový, nebo načtený z projektu. V prvním případě se cesta k obrázku získá z dialogového okna obsahujícího `JFileChooser` z knihovny Swing. V druhém z dialogového okna obsahujícího seznam obrázků projektu, který je získán z instance třídy `Xml.java`. Po předání cesty je ve třídě `MainWindow.java` zkontrolováno, zda daný soubor existuje a v případě, že ano, nastaví se proměnná `Picture` třídy `MainLogic.java`. Dále se nastaví seznam anotací daného obrázku, který se získá z instance třídy `Xml.java` a tyto anotace jsou vykresleny, viz kapitolu 4.2.3. Pokud je obrázek větší než okno aplikace, objeví se posuvníky. Také lze zmenšit měřítko, v jakém je obrázek zobrazen, což je implementováno v třídě `MainWindow.java` a pro potřeby anotace je uloženo měřítko.

4.1.4 Práce s anotacemi

Pokud není otevřen obrázek, jsou pouze anotace obsažené v projektu zobrazeny ve stromě anotačních kategorií jako listy v příslušných kategoriích. Po vybrání se anotace zobrazí vpravo od stromu jako náhled.

Pokud je vybrán obrázek lze v něm spravovat anotace. Všechny anotace aktuálního obrázku jsou v seznamu v levé dolní části obrazovky. Seznam je načten po načtení obrázku, jak je popsáno v kapitole 4.1.3. Zobrazení anotací v obrázku je popsáno v kapitole 4.2.3.

Anotaci vybranou v seznamu anotací aktuálního obrázku lze smazat. Kliknutím na tlačítko smazat anotaci se zavolá metoda removeFeature z třídy MainLogic.java, která zavolá metodu pro smazání anotace z xml pomocí třídy Xml.java. Smaže anotaci z množiny anotací, ze které se tvoří seznam anotací, ten potom aktualizuje. Zavolaná metoda delete() z třídy Anotation.java smaže obrázek anotace z lokálního úložiště. Dále je zavolána metoda recreateAnotations() z třídy AnotationsTree.java, nastaven nový model stromu anotačních kategorií a nakonec je znovu zobrazen obrázek a vykresleny anotace.

Pro uložení anotace je třeba označit anotovaný prvek v obrázku, vybrat anotační kategorii, do které bude anotace uložena a teprve potom kliknout na tlačítko uložit anotaci. Označená, ale zatím neuložená anotace bude v obrázku zobrazena červeně. Po kliknutí na tlačítko uložit anotaci je zavolána metoda **`saveFeature(TreePath selectedTreePath, int scale, int x, int y, int w, int h)`** z třídy **`MainLogic.java`**. Zvýší se počet anotací v projektu a tato informace se uloží do xml. Je vytvořena nová instance třídy **`Anotation.java`** s parametry **`Picture, x, y, w, h`**. Anotace je přidána do množiny anotací a je aktualizován seznam anotací v obrázku. Aktualizuje se strom anotačních kategorií. Anotace je nastavena a uložena v instanci třídy **`labelForAnotations.java`** a ta je překreslena.

4.1.5 Uchovávání v xml

Pro práci s xml dokumentem bylo využito rozhraní DOM, které umožňuje manipulaci s obsahem. Do paměti je načten celý obsah dokumentu a lze s ním pracovat jako s hierarchicky uspořádanými objekty. Oproti tomuto rozhraní je rozhraní SAX rychlejší, protože k obsahu dokumentu přistupuje sekvenčně. Ale chybí možnost úpravy dokumentu, což je pro aplikaci zásadní. Proto rozhraní SAX nebylo použito při implementaci aplikace.

V xml je vždy uchován název projektu a počet anotací v projektu. Dále pokaždé, když je v projektu otevřen obrázek, uloží se jeho jméno a cesta k souboru jako parametry elementu **`img`**. Při ukládání anotace je uložen nový element **`feature`** do elementu **`img`**, jako parametry je uvedena anotační kategorie **`name`**, cesta k obrázku anotace **`path`**, poloha v obrázku **`x`** a **`y`** a velikost anotace **`w`** a h. Definice podoby ukládaného xml souboru ve formátu DTD:

```
<!ELEMENT input (project)>
<!ELEMENT project (anotations, img*)>
<!ATTLIST project
      name CDATA #REQUIERED >
<!ELEMENT anotations EMPTY>
<!ATTLIST anotations
      numberOfAnotations CDATA #REQUIERED >
<!ELEMENT img(feature*)>
<!ATTLIST img
      name CDATA #REQUIERED
      path CDATA #REQUIERED>
<!ELEMENT feature EMPTY>
<!ATTLIST feature
      name CDATA #REQUIERED
      path CDATA #REQUIERED
```

```
x CDATA #REQUIERED
y CDATA #REQUIERED
w CDATA #REQUIERED
h CDATA #REQUIERED >
```

Při zakládání nového projektu je vytvořena nová instance třídy **Xml.java** a ve složce projektu vytvořen nový soubor **project.xml** s elementy **input, project** a **anotations.** Při otevření existujícího projektu je načten soubor xml do paměti. Při otevření nového obrázku je název a cesta k tomuto vložena do xml souboru. Při dotazu na seznam obrázků projektu se projdou všechny elementy **img** a atribut **path** je uložen do seznamu. Při ukládání anotace je do odpovídajícího elementu obrázku vložen nový element **feature** a do elementu **anotations** je vložena nová hodnota atributu **numberOfAnotations.** V případě smazání anotace je tento element z xml souboru smazán na základě názvu. V případě smazání anotační kategorie, se projdou všechny elementy **feature** v dokumentu a porovnává se, zda anotace nepatřila do této kategorie, pokud ano, je element smazán. Analogicky se děje při přejmenování kategorie.

4.1.6 Automatická anotace

Pro implementaci automatické anotace byla z analýzy rozpoznávání obrazu zvolena metoda 2.4.4 Registrace obrazu na základě intenzity. Byla vybrána, protože její implementace není složitá a je názorná. Logika automatického hledání anotací je obsažena ve třídě **Anotator.java.** V metodě **findFeatures(BufferedImage bufferedImage, String featurePath)** je třídě předán obrázek **BufferedImage** a cesta k anotaci, která bude v obrázku hledána. Tato metoda je volána ze třídy **MainLogic.java** ze třech míst pro dvě možnosti hledání – hledání anotační kategorie a hledání anotace. Při hledání kategorie je procházen strom anotačních kategorií a z každé kategorie a podkategorie je metoda zavolána pro jednu anotaci. V paměti jsou obrázek a hledaná anotace uloženy do proměnných typu **BufferedImage** v šedé škále. Postupně jsou volány následující metody **createMultiGrid(), searchMultiGrid(), selectCandidates(), selectAnotations(), createAnotations(String path).**

Metoda **createMultiGrid()** přidá do spojového seznamu **list** instanci vnitřní třídy AnotatorAtt, která obsahuje obrázek a anotaci typu BufferedImage a měřítko typu int. Poté pomocí while cyklu přidává do seznamu obrázek a anotaci polovičních rozměrů než v předchozím cyklu dokud není jeden z rozměrů obrázku menší než 300 pixelů nebo jeden z rozměrů anotace menší ne 15 pixelů nebo měřítko větší než 16. Před prvním průchodem cyklu je měřítko 0 a v každém cyklu se navýší o 1.

Metoda **searchMultiGrid()** prochází **list** s atributy potřebnými k nalezení anotací. V každé iteraci prochází obraz a volá metodu **iteratePixels(Boolean b)** pro počítání chyb. V případě, že ještě nebyli nalezeni žádní kandidáti – **HashMap candidates** je prázdná, projde všechny přípustné pixely v obraze. Jinak prohledává pouze již nalezené kandidáty z **candidates,** dále jen kanidáty. Metoda **iteratePixels(Boolean b)** prochází pixely patche a odpovídající pixely obrazu na aktuální pozici a počítá z jejich

rozdílů chybu. Pro počítání chyby je použit a metoda sumy absolutních rozdílů. Pokud je chyba dostatečně malá, přidá aktuální pozici do množiny kandidátů.

Metoda `selectCandidates()` prochází množinu kandidátů a hledá kandidáty s nejmenší chybou. Kandidáti, kteří mají ve svém okolí alespoň jednoho kandidáta s menší chybou, jsou smazáni ze seznamu kandidátů.

Metoda `selectAnotations()` na základě procházení kandidátů a hledání překrývajících se anotací vybírá konečné anotace. Pokud se anotace překrývají, ta s větší chybou se smaže. Pokud se chyby rovnají, smaže se ta druhá.

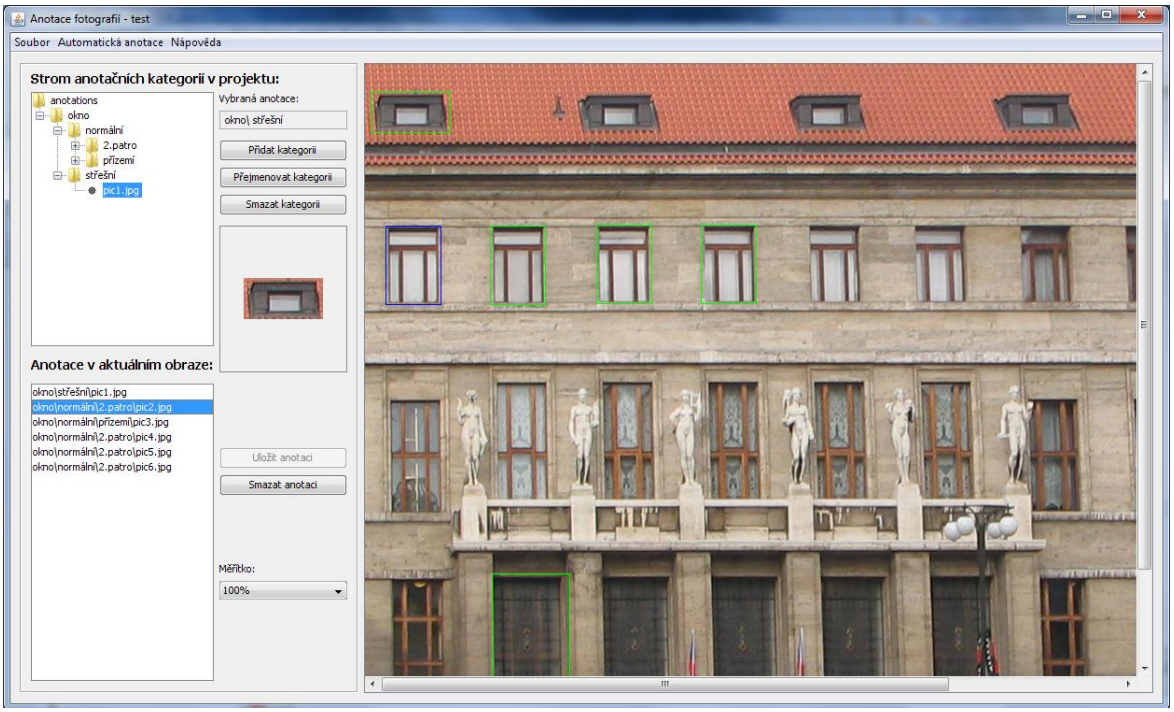
Metoda `createAnotations(String path)` ze zbylých kandidátů vytváří anotace voláním metody z třídy `MainLogic.java`.

4.2 Uživatelské rozhraní

Pro vytvoření uživatelského rozhraní byla použita grafická knihovna Swing, která je součástí standardní instalace JDK. Převážná část byla navržena pomocí nástroje pro návrh GUI obsaženého ve vývojovém prostředí NetBeans verze 6.7.1. S tímto nástrojem se pracuje tak, že po vytvoření třídy rozšiřující třídu `JFrame` nebo `JDialog` z knihovny Swing se lze ze zobrazení zdrojového kódu přepnout do návrhu designu. V tomto návrhovém zobrazení se pracuje v grafickém prostředí a z palety prvků GUI, obsaženými v knihovně Swing, se přetahují prvky do plochy formuláře respektive dialogu. Umístěným prvkům lze v postranní liště programu měnit vlastnosti. NetBeans automaticky generují kostru kódu, do kterého programátor jen dopíše implementaci. Vygenerované části kódu není možné v NetBeans upravovat přímo, ale opět jen změnou vlastností v zobrazení návrhu designu.

4.2.1 Třída MainWindow

Hlavní okno aplikace je rozdělené na dvě části. V pravé části se po otevření zobrazí obrázek a budou se zobrazovat anotace. Levá část je ještě rozdělena na dva celky. V horní části se nachází strom anotačních kategorií a prvky pro jeho správu. Dolní část zobrazuje seznam anotací v aktuálním obraze a taktéž prvky pro správu. V horním menu jsou položky pro práci s projektem, automatickou anotaci a nápovědu. Obrázek 19 zobrazuje hlavní okno aplikace.



Obrázek 19 - Hlavní okno aplikace

4.2.2 Dialogová okna

V aplikaci se nacházejí dialogová okna pro vytvoření a otevření projektu, otevření a načtení obrázku, jeden společný dialog pro přidání, přejmenování a smazání kategorie. Dále je v aplikaci okno pro nápovědu. Hlášky aplikace se otevírají v dialogovém okně pomocí metody `JOptionPane.showMessageDialog(Component parentComponent, Object message)` a nemají proto svojí třídu.

4.2.3 Vykreslování anotací

Pro účel vykreslování anotací byla napsána třída `LabelForAnotations.java`, která rozšiřuje třídu `JLabel` z knihovny Swing. Po otevření obrázku v aplikaci je tento obrázek uložen ve formátu `BufferedImage` do nové instance třídy `LabelForAnotations`. Pokud jsou v obrázku uloženy anotace, je pro každou z nich zavolána metoda `setRectangle` a `saveBlue()`, tím se do obrázku nakreslí zelené obdélníky ohraničující anotace. Pokud uživatel klikne do plochy obrázku a táhne myší je při každé změně volána metoda `repaint`, která způsobí vykreslení červeného obdélníku. Pokud uživatel vybere anotaci v seznamu anotací, je zavolána metoda `saveBlue()`, která nakreslí modré ohraničení. Při smazání anotace se postup opakuje.

5 Testování

Kapitola je rozdělena na tři podkapitoly – testování funkčnosti, kvality automatické anotace a uživatelského rozhraní. V jednotlivých částech bude popsáno, jak testování probíhalo a jaké jsou závěry.

5.1 Testování funkčnosti

Testování funkčnosti probíhalo v několika úrovních. Vzhledem k rozsahu a jednoduchosti případů užití nebyly psány scénáře testů a v případě potřeby bylo nahlédnuto do kapitoly 3. Ihned po přidání části kódu byl kód prověřen pohledem, zda neobsahuje chyby typu překlepů, chybějících závorek a podobně. V tomto bylo velmi nápomocno vývojové prostředí NetBeans, ve kterém byla aplikace implementována, protože provádí kontrolu kódu již během psaní bez nutnosti kompilace. Upozorní programátora například na chybějící deklaraci proměnných, použití špatných datových typů, nutnost importovat knihovnu a podobně.

Po přidání celku funkcionality, například případu užití, či jeho samostatně fungující části, byla tato funkčnost otestována spuštěním aplikace. Uživatelské rozhraní bylo téměř kompletní implementováno jako první, takže nové funkčnosti na něj byly ihned navázány a mohly se spouštět například stiskem tlačítka. Pro potřeby ladění byly v kódu dočasně umístěny různé výpisy na konzoli, které po tom, co byly testy v pořádku, byly smazány. Pokud se objevila nějaká chyba, byla ihned napravena a znovu otestována. Toto se opakovalo při přetrvání chyby či nalezení nové dokud neproběhl test v pořádku.

V průběhu implementace byly rovněž několikrát provedeny regresní testy. Zpočátku pomocí spuštění aplikace v rámci vývojového prostředí, později i spuštěním ze souboru Anotator.jar. Měly za cíl otestovat, že všechny dříve fungující případy užití jsou stále funkční a neovlivnily je později naprogramované případy. Pokud nějaký případ užití neprošel testem, byla hledána chyba a opakoval se průběh testování, jako by šlo o novou funkcionalitu.

5.1.1 Závěr

Všechny případy užití prošly předposledním kolem regresních testů a aplikace byla funkční. Po zapracování výsledků z testování uživatelského rozhraní a posledních drobných úpravách přestalo spolehlivě fungovat vyhledávání anotací z vybrané anotační kategorie. Vzhledem k časové tísni přestala být tato možnost v programu podporována a tlačítko bylo vyřazeno z činnosti. Zbytek aplikace, včetně zbylých možností automatické anotace, je plně funkční.

5.2 Testování kvality automatické anotace

Případy užití ze skupiny automatické anotace prošly testováním funkčnosti, takže vyhledané anotace mohou být podrobeny testům kvality. Vzhledem k metodě, použité pro vyhledávání anotací je vhodné použít fotografie, kde je objekt zabrán tak, aby v něm bylo co nejvíc pravoúhlých linií. Některé fotografie dodal vedoucí práce, jiné jsou ze soukromého archivu. Obrázky použité pro anotaci spolu s ukázkovým projektem naleznete na přiloženém CD ve složce testovani_kvality, která obsahuje i soubor README.txt s návodem. Zde je jen pár obrázků pro ilustraci, více jich je v ukázkovém projektu.

Obrázek 20 - Automatická anotace - modře prvky označené uživatelem, zeleně automaticky nalezené

Obrázek 21 - Automatická anotace pilastru - modře prvek označený uživatelem, zeleně automaticky nalezené

V aplikaci jsou parametry pro vyhledávání zadány v kódu. Při testování funkčnosti byly laděny i tyto parametry. Na některé vyhledávané anotace parametry fungují dobře, viz Obrázek 20 a Obrázek 21. Oproti tomu Obrázek 22 ukazuje špatný výsledek vyhledávání. Všechny zelené anotace v obrázku byly nalezeny hledáním modré anotace. Aplikace nalezla vhodné kandidáty, ale i velmi mnoho špatných kandidátů.

Obrázek 22 - Automatická anotace - nekvalitní výsledek

5.2.1 Závěr

Parametry, které jsou využity pro určení, zda je anotace vhodná či nikoliv nejsou nastavené ideálně. Bylo by vhodné, aby byly uloženy v proměnných, které by uživatel pomocí uživatelského rozhraní mohl měnit. Například pro anotace menších rozměrů by bylo vhodné snížit přípustnou hodnotu chyby, přestože je přepočítána na počet pixelů. Při průchodu aplikací se totiž obrázky zmenšují a tím se ztrácí množství informací využitelných pro porovnání. V aplikaci jsou podmínky, které neumožní přílišné zmenšení obrázku a anotace, ale to není dostačující. Tento nedostatek se tedy stane námětem pro zlepšení aplikace.

5.3 Testování uživatelského rozhraní

Kapitola se věnuje testování uživatelského rozhraní aplikace. Toto testování má za cíl odhalit chyby v designu a případně prvky k vylepšení. Nejdříve musí být definováno nastavení a průběh testu a vybráni participanti. Těm se potom předloží úkoly testu a při plnění se zaznamenávají reakce a jejich chování. Kromě úkolů každý vyplní i pre-test a post-test dotazník, který bude nápomocný při zhodnocení testu.

5.3.1 Nastavení a průběh testu

Test probíhal u běžného pracovního stolu, kde měl participant k dispozici kromě notebooku s touchpadem i počítačovou myš. Na notebooku byl na ploše soubor Anotator.jar, pomocí kterého se spouští aplikace, a dvě složky. V jedné složce se nacházely obrázky, které participant využíval v testu a ve druhé ukázkový projekt. Konfigurace počítače, na kterém participant plnil úkoly:

- Intel Core2 Duo T5470, 1,6 GHz,
- RAM 3GB,
- Windows 7 Professional SP1, 32bit,
- rozlišení displeje 1280x800.

Po celou dobu testování byl u participanta přítomný moderátor, který předem informoval o průběhu testu a poprosil participanta, aby své myšlenky a očekávání sděloval nahlas. Na začátku byl formou rozhovoru vyplněn pre-test dotazník, viz přílohu E. Následně participant dostal zadání úkolů a začal je plnit. Moderátor neradil s řešením úkolů a zapisoval si vše podstatné, co participant řekl nebo udělal. Pokud se participant při plnění nějakého úkolu dlouho trápil, mohl mu moderátor říci, aby úkol přeskočil. Po dokončení testu opět formou rozhovoru společně vyplnili post-test dotazník, viz přílohu F.

5.3.2 Výběr participantů

Pro testování bylo vhodné získat alespoň tři participanty, kteří dobrovolně otestují aplikaci. Vzhledem k tomu, že aplikace není určena pro uživatele, kteří nemají zkušenosti s používáním počítače, byli hledáni participanti, kteří používají počítač v průměru alespoň jednou denně. Jako vodítko pro získání participantů s rozdílnou zdatností ovládání počítače posloužil výběr alespoň jednoho studenta vysoké školy se zaměřením na IT, alespoň jednoho studenta jiné vysoké školy a alespoň jednoho nestudenta. Pro screener viz přílohu D. Bylo osloveno pět potencionálních participantů, kterým byl předložen screener dotazník. Tabulka 1 obsahuje výsledky těchto dotazníků. Po porovnání s neveřejnou částí screener dotazníku se osoby 1, 3 a 4 jeví jako vhodné pro test a pokračovaly v další části testu. Osoby 2 a 5 nevyhověly požadavkům a testu se nezúčastnily. Povedlo se získat dostatečný počet vhodných dobrovolníků a testování mohlo proběhnout dle předpokladů.

Tabulka 1 - Výsledek screeneru

	Osoba 1	Osoba 2	Osoba 3	Osoba 4	Osoba 5
Otázka 1	Používá pravidelně počítač	Používá pravidelně počítač	Používá pravidelně počítač	Používá pravidelně počítač	Používá pravidelně počítač

Otázka 2	Více než 8 hodin denně	1-4 hodiny denně	4-8 hodin denně	4-8 hodin denně	Nepoužívá počítač každý den
Otázka 3	Není studentem VŠ	Není studentem	Je studentem VŠ	Je studentem VŠ	Je studentem VŠ
Otázka 4	-	-	Jiný než IT obor	IT obor	Jiný než IT obor

5.3.3 Úkoly testu

Úkoly testu byly vytvořeny tak, aby obsáhly škálu nejdůležitějších případů užití aplikace. Každý participant plnil tyto úkoly:

1) Otevřete aplikaci *Anotator.jar*, která se nachází na ploše, a vytvořte nový projekt. Pojmenujte ho svým příjmením a uložte na plochu.
2) Vytvořte novou anotační kategorii s názvem *okno* a přidejte jí podkategorii *normální*. Vytvořte novou anotační kategorii *sloup*.
3) Otevřete obrázek s názvem *novy_obrazek.jpg*, který je uložený na ploše ve složce *images*.
4) Označte v obrázku normální okno a uložte anotaci do odpovídající kategorie.
5) Smažte právě vytvořenou anotaci.
6) Smažte kategorii *normální*.
7) Otevřete projekt, který se nachází na ploše ve složce *test_#vašePořadí*. Číslo vašeho pořadí Vám sdělí moderátor.
8) Načtěte z projektu obrázek s názvem *nacteny_obrazek.jpg* a změňte měřítko na 50%.
9) Najděte anotaci střešního okna a automaticky vyhledejte další střešní okna.
10) Automaticky vyhledejte anotace z kategorie normálních oken. Zavřete aplikaci.

Po otevření aplikace měl participant vytvořit nový projekt, aby mohl pracovat s kategoriemi. Vytvoření kategorie a podkategorie testovalo, zda je reprezentace stromem dostatečně intuitivní. Aby mohla být uložena anotace, musel si participant nejdříve otevřít obrázek. Úkol číslo 4) ověřil, zda je posloupnost akcí pro uložení anotace dostatečně zřejmá. Pokud se povedlo vytvoření anotace, měl participant za úkol jí zase smazat. V dalším kroku měl smazat i anotační kategorii. Pro otestování práce s automatickou anotací si participant otevřel předpřipravený projekt, kde si načetl obrázek z projektu, který již obsahoval anotace. Tím se ukázalo, zda uživatel vnímá rozdíl mezi otevřením nového a již použitého obrázku. Měl za úkol automaticky vyhledat anotace střešního okna, což mohl provést dvěma způsoby, a anotace z kategorie střešních oken, což měl udělat pomocí hledání z anotační kategorie. Úkoly na automatickou anotaci testovaly, jestli jsou položky v menu názorné a uživatelsky přijatelné. Nakonec participant zavřel aplikaci.

5.3.4 Pre-test dotazník

Tento dotazník, viz přílohu E, byl s každým participantem vyplněn formou rozhovoru před samotným testováním. Jeho účelem bylo získat větší povědomí o zkušenostech participanta pro lepší vyvození závěru. Tabulka 2 obsahuje odpovědi participantů na jednotlivé otázky.

Tabulka 2 - Výsledky pre-test dotazníku

	Participant 1	Participant 2	Participant 3
Otázka 1	Ano	Ano	Ne
Otázka 2	Ano	Ano	Ano
Otázka 3	Ano	Ano	Ano

5.3.5 Test

Průběh testu bude popsán pro každého participanta zvlášť. Participanti si před zahájením testu nečetli uživatelskou příručku, ale měli k dispozici nápovědu, která je součástí programu. V následujícím textu budou popsány jen ty úkoly, které participantům dělaly potíže, nepovedly napoprvé nebo k nim měli komentář.

Participant 1

- Úkol č. 4 (uložení anotace): Participant nevěděl, jak má v obrázku označit normální okno. Otevřel si nápovědu a po jejím prohlédnutí úkol splnil bez potíží.
- Úkol č. 5 (smazání anotace): Uživatel hledal v horním menu. Poté nahlédl do nápovědy a smazal anotaci. Prý si nevšiml zašedlého tlačítka.
- Úkol č. 8 (načtení obrázku z projektu): Dlouho hledal v ploše hlavního okna, pak se koukl do menu soubor a načetl obrázek.
- Úkol č. 10 (Vyhledání anotace z kategorie normálních oken): Vybral v seznamu anotací v aktuálním obraze anotaci v kategorii okno/normální a zvolil v menu Hledat anotaci zvolenou v aktuálním obraze.

Participant 2

- Úkol č. 4 (uložení anotace): Vybral kategorii, do které chtěl anotaci uložit, označil anotaci v obrázku. Chtěl přejít na další úkol, protože si myslel, že je anotace již uložená.
- Úkol č. 6 (smazat kategorii) Přišlo mu matoucí, že anotace se maže a kategorie se má zrušit, ale klikl na tlačítko zrušit s očekáváním, že se kategorie smaže.
- Úkol č. 9 (nalezení anotace a vyhledání dalších): Byl zmaten, že byly vyhledány i jiné prvky, které nebyly nikdy anotovány, než které dal vyhledat.

Participant 3

- Úkol č. 9 (nalezení anotace a vyhledání dalších): Chvíli zkoumal menu automatické anotace a pak konstatoval, že by se, vzhledem k vzájemné poloze stromu anotačních kategorií a seznamu anotací v obraze, měly prohodit položky menu.

5.3.6 Post-test-dotazník

Po splnění úkolů byl, opět formou rozhovoru, s každým participantem vyplněn post-test dotazník, viz přílohu F. Výsledky dotazníku mohou obsahovat i jiné odpovědi než předem dané, tak, aby vyjádřily názory participanta. Tabulka 3 shrnuje odpovědi všech participantů.

Tabulka 3 - Výsledky post-test dotazníku

	Participant 1	Participant 2	Participant 3
Otázka 1 – rozložení grafických prvků	Grafické prvky jsou rozmístěny přehledně	Grafické prvky jsou rozmístěny přehledně	Grafické prvky by mohly být rozmístěny lépe – seřadit automatickou anotaci dle rozložení v hlavním okně programu
Otázka 2 – srozumitelnost popisků	Nesrozumitelný popisek – „Hledat vybranou anotační kategorii“	Srozumitelné, nekonzistentní pojmenování akce smazat kategorii a anotaci	Nesrozumitelný popisek – „Hledat vybranou anotační kategorii“
Otázka 3 – orientace v aplikaci	Občas jsem si nebyl jistý – „Moje nepozornost“	Snadná	Snadná
Otázka 4 – nejobtížnější úkol	10 – špatně pochopeno, nevhodný popisek	4 – myslel, že je anotace uložena po označení kategorie a nakreslení v obraze	9 – čekal akci na jiném místě v menu
Otázka 5 – nejsnazší úkol	1 – stejné chování jako většina aplikací	5 - intuitivní	3 - stejné chování jako většina aplikací

5.3.7 Soupis problémů a návrh řešení

Problémy, které byly během testování zjištěny, jsou rozděleny podle závažnosti. Ke každému problému je připsán návrh řešení.

Středně závažné problémy

1) Nevhodný název položky v menu – „Hledat vybranou anotační kategorii“
    - Participanty zmátlo, že chtěli hledat anotace z kategorie a bylo jim nabízeno pouze hledat kategorii.
    - Řešení: Přejmenování položky na „Hledat anotace, které jsou ve vybrané anotační kategorii“. Toto řešení bylo implementováno.
2) Anotace se uloží až po stisku tlačítka „Uložit“
    - Paticipanti se domnívali, že po označení kategorie a anotovaného prvku v obrázku se již nemusí anotace ukládat. Poté, co jim bylo ukázáno tlačítko „Uložit anotaci“, řekli, že si ho nevšimli.
    - Řešení: Před otevřením obrázku by v aplikaci na jeho místě mohl být stručný návod pro práci s anotacemi.

Málo závažné problémy

1) Nekonzistentní názvy tlačítek
    - Tlačítka pro stejnou akci, smazání kategorie a anotace, nejsou pojmenována stejně. Jedno má název „Zrušit kategorii“ a druhé „Smazat anotaci“
    - Řešení: Přejmenovat tlačítko „Zrušit kategorii“ na „Smazat kategorii“. Toto řešení bylo implementováno.
2) Pořadí položek v menu Automatická anotace

- Pořadí položek je „Hledat anotaci zvolenou v aktuálním obraze“ a pod ní „Hledat anotaci zvolenou v anotačním stromě“. Přitom v okně aplikace je strom anotačních kategorií první a pod ním je seznam anotací v aktuálním obraze.
- Řešení: Prohození položek v menu tak, aby odpovídaly pořadí prvků v aplikaci. Toto řešení bylo implementováno.

5.3.8 Závěr

Testování uživatelského rozhraní odhalilo dva středně závažné a dva málo závažné problémy. Řešení tří z nich byla implementována, jak je u jednotlivých problémů výše popsáno. Až na tyto problémy neměli uživatelé s orientací potíže a hodnotili rozložení prvků a srozumitelnost popisků kladně. Testování pomohlo vylepšit uživatelské rozhraní, což bylo jeho cílem.

6 Závěr

Povedlo se vytvořit aplikaci, která uživateli umožňuje vytvářet si vlastní strom anotačních kategorií a následně do nich přiřazovat anotace z obrázků. Aplikace také umí automaticky vyhledávat anotace. Tím plní implementační cíle této bakalářské práce. Jak je patrné z tohoto dokumentu, jsou splněny i cíle rešeršní a návrhové.

Na této práci by se dalo pracovat ještě velmi dlouho, neboť z výukových důvodů by aplikace mohla implementovat větší množství metod pro rozpoznávání obrazu a výsledky těchto metod by mohly být porovnávány. Také by se dalo v mnohém vylepšit uživatelské rozhraní. Například by bylo uživatelsky příjemné moci vybrat anotaci rovnou v kliknutím myši v obrázku. Nebo by byla přínosná možnost zobrazit v obrázku anotace jen vybrané kategorie. Jak bylo zmíněno v závěru testování kvality automatické anotace, uživatel by měl mít možnost měnit vyhledávací parametry. Pro tyto účely by se hodila i možnost vzít automatickou anotaci zpět, aby mohly být parametry nastaveny lépe a vyhledávání spuštěno znovu bez uložení nevhodně vyhledaných anotací.

Tato práce pro mě byla velkým přínosem hlavně v oblasti samotné implementace, kde jsem se dosud k tak velkému projektu zatím nedostala. Zkoumala jsem mně neznámé možnosti programovacího jazyka java, které jsem hned mohla v této práci využít a tak si vše snáze zapamatovat. Rešeršní část mi rozšířila obzory v oblasti rozpoznávání obrazu, se kterou jsem se dříve setkala jen velmi okrajově.

Literatura

1. **KOTEK, Zdeněk, et al.** *Metody rozpoznávání a jejich aplikace.* Praha : Academia, 1993. ISBN 80-200-0297-9.

2. **Železný, Miloš.** Strukturální metody rozpoznávání. [Online] [Citace: 30. 1. 2012.] http://www.kky.zcu.cz/uploads/courses/smr/Smr-101012.pdf.

3. **Hlaváč, Václav.** Rozpoznávání s markovskými modely. *Center for Machine Perception.* [Online] [Citace: 5. 2. 2012.] http://cmp.felk.cvut.cz/~hlavac/TeachPresCz/31Rozp/61MarkovianPR.pdf.

4. **Anon.** Rozhodovací stromy. *Center for Machine Perception.* [Online] 27. 4. 2005. [Citace: 6. 2. 2012.] http://cmp.felk.cvut.cz/cmp/courses/recognition/Resources/course_stanclova_pattrec/08%20-%20Rozhodovaci_stromy.ppt.

5. —. Support vector machines. *Masarykova univerzita.* [Online] [Citace: 3. 2. 2012.] http://is.muni.cz/el/1433/podzim2006/PA034/09_SVM.pdf.

6. **Vojtěch Franc, Tomáš Werner.** Cvičení z RPZ - Support Vector Machines. *Center for Machine Perception.* [Online] 2008. [Citace: 3. 2. 2012.] http://cmp.felk.cvut.cz/cmp/courses/recognition/Labs/svm/rpzcv-svm.pdf.

7. **Cristianini, Nello a Shawe-Taylor, John.** *Kernel Methods for Pattern Analysis.* Cambridge : Cambridge University Press, 2004. ISBN 0-521-81397-2.

8. **Burger, Wilhelm a Burge, Mark J.** *Principles of Digital Image Processing.* Guildford, Surrey : Springer London, 2009. ISBN 978-1-84800-194-7.

9. **Werner, Tomáš.** Lineární korelační koeficient. *Center for Machine Perception.* [Online] říjen 2004. [Citace: 29. 12. 2012.] http://cmp.felk.cvut.cz/cmp/courses/383ZS/ZSO2005-6/cvic2/ncc.pdf.

10. **Šochman, Jan.** Cvičení z RPZ – AdaBoost. *Center for Machine Perception.* [Online] 26. 5. 2005. [Citace: 5. 2. 2012.] http://cmp.felk.cvut.cz/cmp/courses/recognition/Labs/adaboost/adaboost.pdf.

11. **Šochman, Jan a Matas, Jiří.** AdaBoost. *Centre for Machine Perception.* [Online] [Citace: 5. 2. 2012.] http://cmp.felk.cvut.cz/~sochmj1/adaboost_talk.pdf.

12. **Bc. Juránek, R.** Rozpoznávání vzorů v obraze pomocí klasifikátorů. *diplomová práce.* Brno : FIT VUT v Brně, 2007.

A Seznam obrázků

B Seznam tabulek

C Seznam zkratek a slovníček pojmů

GUI Graphical User Interface, grafické uživatelské rozhraní

JDK Java Development Kit, soubor základních nástrojů pro vývoj aplikací v jazyku Java

XML Extensible Markup Language, obecný značkovací jazyk

DTD Document Type Definition, jazyk pro popis struktury XML

SAX Simple API for XML

DOM Document Object Model

Participant účastník testování uživatelského rozhraní

Screener dotazník zjišťující, zda se uchazeč pro test hodí

Pre-test dotazník, který se s participantem vyplňuje před samotným testem

Post-test dotazník, který se s participantem vyplňuje po testu

D Screener

Veřejná část

Otázka 1. Používáte pravidelně počítač?

a) Ano
b) Ne

Otázka 2. Jak často používáte počítač (průměrně)?

a) Méně než hodinu denně
b) 1-4 hodiny denně
c) 4-8 hodin denně
d) Více než 8 hodin denně
e) Nepoužívám počítač každý den

Otázka 3. Jste studentem vysoké školy?

a) Ano
b) Ne

Otázka 4. Pokud jste student, jaký typ vysoké školy studujete?

a) Se zaměřením na IT
b) Jiný

Neveřejná část

Tabulka 4 - Počet participantů zamýšlené skupiny

<table>
<tr><td>Číslo otázky/ Odpověď</td><td>1</td><td>2</td><td>3</td><td>4</td></tr>
<tr><td>a)</td><td>3</td><td>0</td><td>2</td><td>1</td></tr>
<tr><td>b)</td><td>0</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>c)</td><td>-</td><td rowspan="2">3</td><td>-</td><td>-</td></tr>
<tr><td>d)</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td>e)</td><td>-</td><td>0</td><td>-</td><td>-</td></tr>
</table>

E Pre-test dotazník

Otázka 1. Setkal/a jste se již někdy s pojmem anotace fotografií?

a) Ano
b) Ne

Otázka 2. Používáte nějaký grafický editor (např. Gimp, Photoshop, Zoner Photo Studio) alespoň jednou za měsíc?

a) Ano
b) Ne

Otázka 3. Umíte v grafickém editoru používat nástroj pro výběr?

a) Ano
b) Ne

F Post-test dotazník

Otázka 1. Jak byste ohodnotil/a rozložení grafických prvků v aplikaci?

a) Grafické prvky jsou rozmístěny přehledně
b) Grafické prvky by mohly být rozmístěny lépe
c) Grafické prvky jsou rozmístěny nevhodně

Otázka 2. Byly pro vás popisky a pojmenování akcí srozumitelné?

a) Ano
b) Ne
c) Jak které

Otázka 3. Jak jste se v aplikaci orientoval?

a) Snadno
b) Občas jsem si nebyl jistý
c) Často jsem se ztrácel
d) Nemohl jsem nic najít

Otázka 4. Který úkol byl pro vás nejobtížnější a proč?

Otázka 5. Který úkol vám přišel nejsnazší a proč?

G Instalační a uživatelská příručka

Instalace programu

Aplikace nevyžaduje žádnou instalaci, je připravena ve spustitelné podobě v souboru Anotator.jar. Pro spuštění ale musí být na počítači nainstalováno Java Runtime Environment ve verzi alespoň 1.7.

Uživatelská příručka

Po spuštění se objeví okno rozdělené na dvě hlavní části. V pravé části se po otevření zobrazí obrázek a budou se zobrazovat anotace. Levá část je ještě rozdělena na dva celky. V horní části se nachází strom anotačních kategorií a prvky pro jeho správu. Dolní část zobrazuje seznam anotací v aktuálním obraze a taktéž prvky pro správu. V horním menu jsou položky pro práci s projektem, automatickou anotaci a nápovědu.

Projekt

Pro jakoukoli práci v aplikaci je třeba mít otevřený projekt. Ten se buď může vytvořit nový, nebo otevřít existující. Název projektu se zobrazí v titulku okna za titulkem Anotace fotografií.

Anotační kategorie

Anotační kategorie se váží k celému projektu. Jsou zobrazeny jako strom, kde se všechny kategorie vkládají do adresáře anotations. Struktura funguje stejně jako adresářová, neboť je tak i reprezentována na lokálním úložišti. Kategorie lze libovolně přidávat. Kategorie se přidá vždy do označené části stromu, pro snadnou orientaci je vybraná anotace vypsána. Dále lze kategorie přejmenovávat a rušit, což zruší i všechny podkategorie a anotace obsažené v mazaných kategoriích.

Obrázek

Obrázek lze otevřít pouze, pokud je otevřený projekt a to buď nový, nebo načtený z projektu. Pokud je obrázek větší než okno aplikace, objeví se posuvníky. Také lze zmenšit měřítko, v jakém je obrázek zobrazen. Jestliže obrázek obsahuje anotace, jsou tyto anotace zobrazeny v obrázku.

Anotace

Pokud není otevřen obrázek, jsou pouze anotace obsažené v projektu zobrazeny ve stromě anotačních kategorií jako listy v příslušných kategoriích. Po vybrání se anotace zobrazí vpravo od stromu jako náhled.

Pokud je vybrán obrázek lze v něm spravovat anotace. Všechny anotace aktuálního obrázku jsou v seznamu v levé dolní části obrazovky. Také jsou zeleně zobrazeny přímo v obrázku. Po vybrání anotace ve stromu se vybraná anotace změní v obrázku ze zelené na modrou. Vybranou anotaci lze smazat.

Pro uložení anotace je třeba označit anotovaný prvek v obrázku, vybrat anotační kategorii, do které bude anotace uložena a teprve potom kliknout na tlačítko uložit anotaci. Označená, ale zatím neuložená anotace bude v obrázku zobrazena červeně.

Automatická anotace

Pomocí automatické anotace lze vyhledávat třemi způsoby. Prvním z nich je vyhledání anotací z vybrané anotační kategorie. Aplikace projde kategorii a všechny jejích podkategorie a v každé vybere poslední anotaci. Každou vybranou anotaci poté hledá v obrázku a nalezené anotace uloží do takové kategorie, z které byla referenční anotace. Druhým způsobem je hledání anotace, která byla označena v seznamu anotací v aktuálním obraze. Která to je, lze vidět v obrázku – je označena modře. Třetím způsobem je hledání anotace, která byla vybrána ve stromě anotačních kategorií. Referenční anotace se tedy může, ale nemusí nacházet v aktuálním obrázku. Náhled anotace je vidět vpravo od stromu.

Ukládání

Jediné ukládání, o které se uživatel musí starat, je ukládání anotací (bylo popsáno výše). Vše ostatní je ukládáno aplikací automaticky při změně.

H Obsah přiloženého CD

Na CD se nachází:

- **Anotator –** složka obsahující spustitelný soubor, zdrojové kódy a dokumentaci
  - **src –** složka obsahující zdrojové kódy
  - **javadoc –** složka obsahující dokumentaci zdrojového kódu
  - **Anotator.**jar – spustitelný soubor aplikace
- **text –** složka obsahující text bakalářské práce ve formátech pdf a docx
  - **bakalarska_prace.pdf**
  - **bakalarska_prace.docx**
- **testovani_kvality –** složka obsahující vzorový projekt a fotografie
  - **Ukazkovy_projekt –** složka
    - **anotations –** složka obsahující kategorie a anotace projektu
    - **project.xml –** soubor s projektem, který se otevírá v aplikaci
  - **Fotografie_test**
    - **fotografie_dodane_vedoucim**
    - **ostatní_fotografie**
  - **README.txt –** návod pro použití ukázkového projektu
- **README.**TXT – instalační a uživatelská příručka