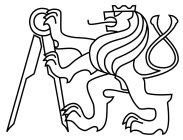
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Detekce kvality návrhu procesního modelu na základě nástrojů strojového vidění
Student:	Oliver Findra
Vedoucí:	Ing. Josef Pavlíček, Ph.D.
Studijní program:	Informatika
Studijní obor:	Informační systémy a management
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2017/18

Pokyny pro vypracování

1) Seznamte se s používanými standardy/formáty pro ukládání obsahu procesních diagramů (např. XPDL, BPMN, …).
2) Seznamte se s již existujícími nástroji pro výpočet hodnot měr kvality procesních modelů (BPMN Quality Tool, CoCoFlow, ProM, BPMN Measures, …).
3) Proveďte rešerši nástrojů strojového vidění za účelem detekce objektů v procesním modelu (diamant = proxy, obdélník se zaoblenými hranami = aktivita, šipka = procesní tok atd.)
4) Navrhněte řešení rozpoznání symbolů procesního modelu (na základě analýzy obrazu či s využitím nástrojů umělé inteligence) na základě provedené rešerše a konzultací s vedoucím práce.
5) Vyberte vhodnou oblast implementace (např. rozpoznání počtu aktivit a rozhodování v modelu) a tu implementujte v jazyce Java.
6) Implementaci zdokumentujte a zhodnoťte získané výsledky v závěru práce.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 23. ledna 2017

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Bakalárska práca

Detekcia kvality návrhu procesného modelu na základe nástrojov strojového videnia

Oliver Findra

Vedúci práce: Ing. Josef Pavlíček, Ph.D.

16. mája 2017

Poďakovanie

Touto cestou chcem poďakovať vedúcemu bakalárskej práce Ing. Josefovi Pavlíčkovi, Ph.D. za pomoc, odborné vedenie, cenné rady a pripomienky pri vypracovaní mojej bakalárskej práce.

Prehlásenie

Prehlasujem, že som predloženú prácu vypracoval(a) samostatne a že som uviedol(uviedla) všetky informačné zdroje v súlade s Metodickým pokynom o etickej príprave vysokoškolských záverečných prác.

Beriem na vedomie, že sa na moju prácu vzťahujú práva a povinnosti vyplývajúce zo zákona č. 121/2000 Sb., autorského zákona, v znení neskorších predpisov, a skutočnosť, že České vysoké učení technické v Praze má právo na uzavrenie licenčnej zmluvy o použití tejto práce ako školského diela podľa § 60 odst. 1 autorského zákona.

V Prahe 16. mája 2017 .....

Odkaz na túto prácu

Findra, Oliver. *Detekcia kvality návrhu procesného modelu na základe nástrojov strojového videnia.* Bakalárska práca. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Táto práca sa zameriava na oblasť procesných modelov a merania ich kvality na základe počítačového videnia. Teoretická časť popisuje používané štandardy procesných modelov, identifikuje procesné miery a nástroje na ich výpočet. Taktiež pokrýva oblasť počítačového videnia a známe knižnice pre prácu s obrazom. Praktická časť je venovaná návrhu a implementácii nástroja, ktorý dokáže detegovať základné prvky BPMN diagramu s použitím knižnice OpenCV.

Klíčová slova podnikový proces, procesný model, BPMN, procesné miery, počítačové videnie, OpenCV

Abstract

This work focuses on field of business process models and calculating its metrics based on computer vision. Theoretical part describes most used standards in business process modelling, identifies business process metrics and tools for its calculation. It also covers field of computer vision and known libraries for image processing. Practical part is devoted to design and implementation of application, which can detect basic elements of BPMN diagram with usage of OpenCV library.

Keywords business process, business process model, BPMN, business process metrics, computer vision, OpenCV

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

Priorita procesného riadenia a postavenie samotných procesov v podnikoch za posledné roky rapídne rastie. Vzniká neustály tlak na zvyšovanie efektivity fungovania firiem. Spôsobovať to môže viacero faktorov. V minulosti bolo zákazníkov veľa a firmy veľakrát nedokázali všetkých uspokojiť. Dnes je už ale trh nasýtený a zákazník sa stáva pánom. Na trh ale prichádza čím ďalej tým viac konkurencie a tí, ktorí zaostávajú, neprežijú. Zvyšovanie efektivity je preto kľučové. Jedným zo spôsobov je procesné riadnie - *business process management*. Procesy treba kontrolovať ako pri ich vytváraní, tak aj po tom a aj pri ich vykonávaní.

V mojej práci sa venujem procesom a ich zápisu - procesným modelom. Mojim cieľom je zoznámiť sa s procesnými modelmi, ich ukladaním a hodnotením ich kvality. Následne by som chcel vytvoriť nástroj, ktorý dokáže ohodnotiť procesný model. Tohto problému sa chcem ale ujať z iného uhla. Nástrojov na hodnotenie modelov existuje mnoho, väčšina z nich hodnotí procesný model na základe mier vypočítaných z XML dokumentov a pod. V mojej práci budem riešiť hodnotenie modelu na základe obrazovej informácie o ňom. To znamená, že sa budem musieť oboznámiť s témou počítačového videnia a nájsť správny spôsob implementácie.

Práca je rozdelená na 5 častí. V prvej sa venujem pojmom proces, procesné riadenie a procesný model a rozoberiem štandardy pre ukladanie procesov. Druhá časť patrí kvalite procesného modelu a existujúcim riešeniam jej implementácie. V tretej časti rozoberiem tému počítačového videnia, popíšem niektoré metódy spracovania obrazu. Predposledná časť sa venuje už existujúcim nástrojom, ktoré uľahčujú užívateľom prácu s obrazom. Nakoniec si zo znalostí z predošlých častí vyberiem spôsob implementácie už spomínaného nástroja a naprogramujem ho v jazyku Java.

Kapitola 1

Procesné modely a ich ukladanie

Na začiatok by som chcel ozrejmiť, čo to vlastne podnikový proces je, akú zastáva funkciu v podnikoch a prečo je dôležité kontrolovať kvalitu procesných modelov.

1.1 Podnikový proces a jeho riadenie

S procesmi sa môžme stretnúť všade okolo nás. Ako proces sa dá charakterizovať väčšina vykonávaných činností, ktoré vidíme. Najčastejšie sa ale tento pojem spája s firmami a organizáciami, ktoré sa živia v oblasti informačných systémov a technológií.

Organizácia je vlastne organizovaná sústava procesov a činností, ktoré na seba vzájomne nadväzujú, vzájomne interagujú, prebiehajú naprieč jednotkami organizácie, reagujú na rôzne podnety z vnútorného i vonkajšieho prostredia. Každý proces má isté vstupy, ktoré sa transformujú na výstupy.[1] Procesy sú teda základnou časťou každej organizácie a podniku, tiež sa im hovorí podnikové procesy.

Podnikový proces (niekedy tiež obchodný proces alebo angl. „business process“) je definovaný v [2] ako *„objektívne prirodzenú postupnosť činností, konaných s úmyslom dosiahnutia daného cieľa v objektívne daných podmienkach“*.

V [3] je zase definovaný proces podľa organizácie Workflow Management Coalition takto:

„Podnikový proces je množina jednej alebo viacerých prepojených činností spoločne prispievajúcich k dosiahnutiu podnikového cieľa, zvyčajne vo väzbe na organizačnú štruktúru, ktorá definuje funkčné role a vzťahy“.

V podnikových procesoch je teda hlavné mať cieľ, presne danú postupnosť konania činností, ich štruktúru, vzťahy a podmienky ich vykonania. V minulosti sa k procesom pristupovalu úplne inak ako je to dnes. Postupom času

môžeme pozorovať rôzne časové vlny obchodných procesov ako sa to opisuje v [4].

Najprv je obdobie, keď procesy v rámci firiem boli len akési návyky, ktorými boli realizované činnosti. Väčšinou išlo len o slovný popis práce. S rýchlym napredovaním informačných technológií sa stredom záujmu stali informácie. Pri návrhu informačných systémov sa zostavovali aj procesy organizácie (pretože neboli dobre definované) a tie potom nemohli byť jednoducho zmenené bez nákladných zmien v informačných systémoch. V tomto období sa dajú procesy označiť ako „obete“ informačných systémov.[4]

S neustálym tlakom na zvyšovanie efektivity fungovania podniku sa firmy snažia procesy zlepšovať a optimalizovať. Procesy sa postupne dostávali do hlavnej úlohy podnikov a dnes už presne určujú postupy väčšiny firiem. Tým vzniklo nové odvetvie – procesné riadenie (alebo aj BPM - angl. *business process management*).[4]

„*Procesným riadením sa rozumie riadenie firmy takým spôsobom, v ktorom business (podnikové) procesy hrajú kľučovú rolu*“.[2]

Základom procesného riadenia je hlavne pochopenie základnej logiky biznisu - hlavným činnostiam v podniku a súvislostiam medzi nimi. Tieto informácie potom vo forme obchodných procesov určujú základné fungovanie celej firmy. Význam ďalších záležitostí vo firme je potom odvodený z významu týchto procesov.[2]

1.2 Procesný model

Textový popis procesu pomocou činností nie je dostatočný popis. Človek, ktorý má daný proces vykonávať, sa môže stratiť v množstve informácií. K zachyteniu väzieb a celkovej štruktúry procesov sa používa procesný model. Model nám má slúžiť ako zdroj informácií, ktoré sú nutné na pochopenie procesu, jeho analýzu a riadenie.[5]

Pri modelovaní procesov v podniku vznikajú dva modely - globálny model systému procesov a detailný model procesu. Globálny model popisuje existenciu procesov a vzťahov medzi nimi. Detailný model popisuje už daný proces ako usporiadanú štruktúru akcií, nutných k dosiahnutiu daného cieľa procesu. Pre tento model sa používa diagram procesu (*Process Diagram*).[2]

Diagram procesu teda znázorňuje priebeh jedného procesu. V [2] je diagram opísaný ako *„kombinácia logiky postupu dosiahnutia príslušného cieľa procesu a vplyvu okolitých procesov, s ktorými sa musí synchronizovať“*. Diagram má vlastnú vnútornú logiku, ktorá je nazávislá od ostaných procesov, ale zase s nimi musí byť synchronizovaný, väčšinou cez spojenie počiatočných a koncových udalostí. Pre jasné a presné definovanie diagramu sú potrebné určité pravidlá, aby nevznikali rozdielne interpretácie. Spočiatku pre tento účel dominoval jazyk UML, ktorý sa používal hlavne v oblasti softvérového

inžinierstva. Pre človeka, ktorý tento jazyk nepozná, bolo niekedy obtiažne pochopiť logiku týchto diagramov a to otvorilo cestu pre iné jazyky.

1.3 Štandardy

Štandardom procesného modelu sa dá označiť súbor pravidiel a princípov používaných pri modelovaní, ukladaní a následnom vykonávaní obchodných procesov. Môžeme rozoznávať rôzne typy štandardov pre procesné modely. Štandardy môžu udávať pravidlá grafického zobrazenia modelu (napríklad BPMN), pravidlá ukladania modelu (napríklad XPDL) alebo sa zameriavať na vykonávanie modelu (napríklad BPEL).[6]

Počiatky štandardov siahajú do 90. rokov 20. storočia, kedy zaznamenávame niekoľko pokusov o vytvorenie používaných štandardov pre procesné modely. Ako jedna z prvých sa o to pokúsila spoločnosť Workflow Management Coalition (WfMC) v roku 1994. Jej štandardy boli (a v súčasnosti ešte stále sú) založené na základe formátu XML. WfMC ale zo začiatku nedosiahlo veľkého úspechu, dokonca ani medzi spoločnosťami, ktoré ju sponzorovali. Štandardy WfMC zaostávali za rýchlo napredujúcim softvérovým priemyslom. Podobne zozačiatku dopadol aj SWAP (Simplified Workflow Access Protocol), nasledovník od WfMC v rokoch 1997-1998.[7]

V súčasnosti existuje mnoho štandardov pre procesné modely. Jeden z najpoužívanejších jazykov na prenos procesných modelov je XPDL (XML Process Definition Language). Medzi používané štandardizované notácie patria napríklad EPC (Event-driven Process Chains), UML (Unified Modelling Language) alebo BPMN (Business Process Modelling Notation). V nasledovných sekciách ich popíšem.

1.3.1 XML Process Definition Language

XML Process Definition Language (XPDL) je jazyk, ktorý vytvorila organizácia WfMC. Je založený na formáte XML a jeho účel je slúžiť ako formát pre výmenu procesných modelov medzi rôznymi nástrojmi.[8]

Hlavné prvky XPDL uvedené v [8] sú:

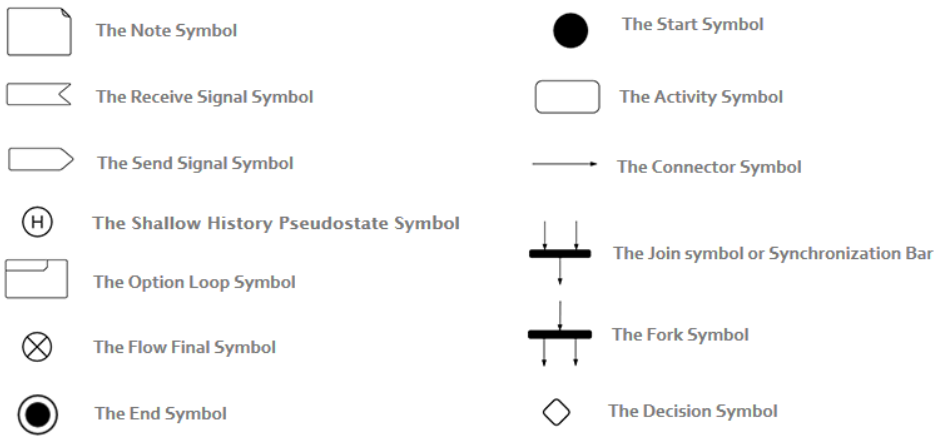
- Package - koreňový element združujúci ďalšie elementy.
- Application - aplikácia vyvolaná procesom
- Workflow Process - definuje proces alebo jeho časť
- Activity - základný stavebný blok procesu
- Transition - prepojuje aktivity
- Participant - účastník

- DataField - dátové pole
- DataType - dátový typ

1.3.2 Unified Modelling Language

Unified Modelling Language (UML) je vizuálny, objektovo orientovaný a viacúčelový modelovací jazyk. Primárne je určený na modelovanie softvérových systémov, ale používa sa aj pri modelovaní procesov. Tvorcami UML sú Rumbaugh, Booch, and Jacobson. Prvá vezia (UML 1.0) vyšla v roku 1997 a bola akceptovaná ako štandard organizáciou Object Management Group (OMG) v tom istom roku. OMG pokračovala vo vývoji UML aj ďalšie roky. Softvérový priemysel rýchlo akceptoval tento jazyk, hlavne po jeho vezii UML 1.3 v roku 1999.[8]

Jazyk UML obsahuje viacero druhov diagramov určených pre rôzne účely modelovania. Pre modelovanie procesov sú základnými nájstrojmi tzv. *activity diagrams* (diagramy aktivít). Diagramy aktivít vizualizujú sekvencie činností, ktoré sa majú vykonať, vrátane toku riadenia a toku údajov. Prvky diagramu aktivít sú zobrazené na obrázku 1.1.[8]



Obr. 1.1: Prvky UML diagramu aktivít[9]

1.3.3 Event-driven Process Chain

Event-driven process chain (ďalej EPC) bola vyvinutý v roku 1992 Inštitútom pre informačné systémy (Institute of Information Systems) Univerzity v Saarlande, Nemecku. EPC je notáciou, ktorá sa spája hlavne s modelovacími konceptami SAP a platformou ARIS, ktorá slúži na návrh, implementáciu a

kontrolu podnikových procesov. Je intuitívna a čitateľná, čo bolo hlavným dôvodom jej uplatnenia. Avšak sémantika notácie je niekedy nejednoznačná, čo môže spôsobovať problémy.[8] EPC diagram sa skladá z týchto prvkov:

Funkcie sú hlavným elementom diagramu. Plní funkciu aktivity, ktorá sa má vykonať.

Udalosti popisujú situácie pred alebo po tom, čo sa funkcia vykoná. Funkcie sú spájané udalosťami. Udalosť môže byť zároveň podmienkou jednej funkcia a zároveň výstupom druhej funkcie.

Logické konektory (spojky) spájajú funkcie a udalosti a určujú tok riadenia procesu. V EPC existujú tri typy konektorov - AND, XOR a OR, ktoré plnia funkciu logických operátorov medzi konektormi.[10]

Obr. 1.2: Prvky EPC diagramu[10]

1.3.4 Business Process Modelling Notation

Primárnym cieľom Business Process Modelling Notation (ďalej BPMN) je poskytnúť štandard pre diagramy jasne popisujúce proces, ale zároveň ľahko pochopiteľné a užitočné pre všetky začlenené strany, od analytikov, ktorí vytvárajú a vylepšujú podnikové procesy, vývojárov, ktorí sú zodpovední za implementáciu až po manažérov na kontrolu a usmernenie.[11]

BPMN bol pôvodne vytvorený spoločnosťou Business Process Modeling Initiative ako grafická notácia pre obchodné procesy. Rastúci záujem o túto notáciu spôsobil prijatie notácie spoločnosťou Object Management Group (OMG) ako štandard. Novšia verzia BPMN 2.0 si zachovala znaky predošlej verzie, ale vylepšila jej schopnosti. Okrem iných poskytuje XML schémy, ktoré slúžia na transformáciu modelov.[12]

V BPMN sa proces zobrazuje ako graf z tzv. flow elementov (prvkov), čo môžu byť aktivity, udalosti, rozhodovacie brány a sekvenčné toky. Proces sa môže definovať na akejkoľvek úrovni podniku, nezáleží na počte účastníkov či veľkosti procesu.[12]

Ako je uvedené v [11], prvky BPMN diagramu sa definujú takto:

Tokové objekty (Flow Objects) Tokové objekty sú kľúčové prvky modelu, ktoré definujú správanie procesu. Pozostávajú z troch hlavných elementov – udalosť, aktivita a brána.

Udalosť (Event) Udalosť a značí vonkajšiu udalosť ktorá sa udiala počas chodu procesu. Udalosti sa delia na 3 druhy podľa toho, kedy ovplyvňujú proces - počiatočné (start), stredné (intermediate) a koncové (end). Počiatočné udalosti sa značia jednoduchým kruhom s tenkou čiarou, stredné kruhom s dvojitou tenkou čiarou a koncové sú odlíšené hrubou čiarou.

Aktivita (Activity) Aktivita je výraz pre prácu v procese, ktorú má účastník vykonať. Môžu byť jednoduché a zložité. Typy aktivít v procesnom modeli sú podproces a úloha. Obidve sa značia obdĺžnikom so zaoblenými rohmi.

Brána (Gateway) Rozhodovacie brány slúžia na rozdeľovanie a spojovanie sekvenčného toku procesu. Definujú vetvenie, rozdeľovanie, zlučovanie a spájanie toku v modeli. Brány sa značia štvorcom otočeným o 45 stupňov a ich vnútorné značky indikujú ich typ.

Dáta (Data) Dáta sa rozdeľujú na dátové objekty, dátové vstupy a výstupy. Dátové objekty zobrazujú informácie, ktoré potrebujú aktivity, aby mohli byť vykonané, alebo informácie, ktoré aktivity vyprodukujú. Takú istú úlohy plnia aj dátové vstupy a výstupy, ale pre celý proces.

Spojovacie objekty (Connecting objects)

Sekvenčný tok (Sequence flow) Sekvenčný tok určuje poradie, ktorým sa vykonávajú aktivity v procese. Značí sa šípkou s plnou čiarou.

Tok správ (Message flow) Používa sa na zobrazenie toku správ medzi 2 zúčastnenými v rámci procesu, ktorí sú pripravení ich poslať alebo prijať. V BPMN sú dvaja rôzni účastníci reprezentovaní dvomi rôznymi bazénmi v diagrame. Tok správ je označený šípkou s prerušovanou čiarou.

Asociácie(Association) Asociácie sa používajú na spojenie informácií a artefaktov s ostatnými grafickými prvkami. Označujú sa bodkovanou čiarou. Ak je potrebný aj smer toku, označí sa šípkou na konci.

Plavecké dráhy (Swimlanes)

Bazén (Pool) Bazén je grafická reprezentácia účastníka. Je to veľký obdĺžnik, ktorý v sebe obsahuje elementy diagramu, ktoré vykonáva daný účastník. Bazén môže mať vnútorné detaily, ktoré budú vykonané, ale nemusí, t. j. môže sa chovať ako „čierna skrinka“.

Dráha (Lane) Dráha je menší úsek procesu alebo bazénu. Používa sa na organizáciu a kategorizáciu aktivít.

Artefakty (Artifacts) Artefakty prinášajú ďalšie informácie o procese. Existujú dva štandardizované artefakty, pri modelovaní je ale povolené pridávať vlastné, ak je to potrebné.

Skupina (Group) Slúži na zoskupenie prvkov diagramu, ktoré patria do rovnakej kategórie. Skupiny nemajú žiadny vplyv na sekvenčný tok. Slúžia iba ako označenia istých skupín. Kategórie môžu byť vhodné pre dokumentáciu alebo analýzu procesu.

Textová anotácia (Text annotation) Slúži ako prídavná informácia pre čitateľa BPMN diagramu.

1.4 Zhodnotenie

V tejto sekcii som zadefinal základné pojmy súvisiace s procesným modelom. Popísal som najpoužívanejšie štandardy pre modelovanie procesov. Najpoužívanejšou notáciou pre procesné modely je BPMN, ktorá je na jednej strane ľahko čitateľná a má jednoznačne danú sémantiku. Notácia UML sa viac využíva v oblasti softvérového inžinierstva.

Kapitola 2

Kvalita procesného modelu

Pri tvorení procesného modelu je hlavné, aby sa zjednodušila komunikácia a zefektívnili činnosť medzi začlenenými stranami. Najdôležitejšími faktormi u modelu je, aby boli ľahko pochopiteľné a dobre sa udržovali. Miery kvality procesného modelu nám väčšinou poskytujú adekvátne informácie o už vyššie spomínanej pochopiteľnosti a ľahkej údržbe procesného modelu.[13]

2.1 Miery kvality procesného modelu

Procesný model má veľa podobných vlastností ako tradičné programovacie jazyky. Program je väčšinou rozdelený na triedy alebo funkcie, ktoré dostávajú určitý vstup a poskytujú výstup pre ďalšie použitie. Podobnú štruktúru majú aj procesné modely. Podobné prvky medzi modelmi a programami sú načrtnuté v tabuľke 2.1. Miery kvality modelu a softvérového programu sú preto veľmi podobné a vo veľkej miere na seba nadväzujú.[13]

Miery používané pri hodnotení procesných modelov uvedené v [14]:

- Veľkosť (size)
- Spojitosť (coupling)
- Súdržnosť (cohension)
- Zložitosť (complexity)
- Modularita (modularity)

2.1.1 Veľkosť

Najľahšou mierou kvality pre softvér je počet riadkov v kóde – LOC („lines of code“), ktorý reprezentuje veľkosť programu. Pri vyšších programovacích jazykoch, LOC určuje počet spustiteľných príkazov (bez komentárov, prázdnych riadkov, ...).

Program	**BPMN**
Balík/Trieda	Proces/Podproces
Metóda	Aktivita
Premenná	Dátový objekt
Komentár	Anotácia
Zavolanie metódy	Aktivita, ktorej predchádza sekvenčný tok alebo tok správ
Rozhranie triedy	Rozhranie procesu - aktivity, ktoré príjmajú alebo odosielajú tok správ
Premenné použité v triede	Dáta použité alebo generované v procese
Lokálne premenné triedy	Dáta vytvorené v rámci procesu, ktoré nie sú používané mimo neho

Tabuľka 2.1: Podobnosť medzi štruktúrou objektovo orientovaného softvérového programu a podnikového procesu v notácii BPMN[13]

Niečo podobné ako LOC by sa dalo v procesných modeloch definovať ako počet aktivít. Poznáme však viacero variant týchto mier, ako napríklad počet aktivít a spojení, počet udalostí, atď. Sú to veľmi jednoduché miery, ktoré majú ale značnú nevýhodu. Dva procesné modely môžu síce obsahovať rovnaký počet aktivít, ale ich štruktúra a zrozumiteľnosť môžu byť kvalitatívne veľmi odlišné.[14]

2.1.2 Spojitosť

Spojitosť meria počet spojení (alebo prepojenosť) v modeli. Tento typ miery je blízky meraniu hustoty a stupňa v (sociálnych) sieťach. Hustota sa meria ľahko ak je model dostupný ako graf. Hustota (ako aj spojitosť sama o seba) sa najmä spája s predikovaním chýb v procesnom modeli. Ukázalo sa, že medzi hustotou a počtom chýb je isté prepojenie, avšak čím viac sa od seba modeli líšia veľkosťou, tým rozdielnejšie výsledky dostávame. Preto je pre procesné modeli v rámci spojitosti lepšie používať metriku priemerného stupňa prvku - ako už z názvu vyplýva, miera počíta priemerný stupeň prvkov, väčšinou aktivít.[14]

Reijers a Vanderfeesten vytvorili mieru spojitosti, kde počítajú prekrytie dátovými prvkami pre každý pár aktivít - *Process Coupling.*

„Miera vyjadruje zložitosť procesu ako zložitosť prechodov medzi jednotlivými aktivitami, stupeň ich spojitosti a závislosti“.[15]

Postup pri výpočte miery je zhruba nasledovný. Dve aktivity tvoria „pár“, ak obsahujú jeden alebo viac spoločných dátových prvkov. Aktivity sú volené po pároch a spočíta sa počet „spárovaných“ párov. Miera je napokon vypočítaná na základe celkového počtu aktivít. Výsledok leží niekde medzi 0 a

1.

2.1.3 Súdržnosť

Reijers a Vanderfeesten taktiež vytvorili mieru súdržnosti pre procesy, ktorá sa hlavne pozerá na súdržnosť medzi aktivitami. Pre každú aktivitu sa vypočíta celková súdržnosť (pomocou informačnej a relačnej súdržnosti danej aktivity). Pre celý proces sa spočíta aritmetický priemer súdržností aktivít. Konečný výsledok súdržnosti leží medzi 0 a 1. Po skombinovaní s predošlou mierou spojitosti dostávame tzv. „coupling – cohension ratio“ (čiže pomer spojitosti a súdržnosti). Pre čo najlepší model podľa tohto pomeru sa snažíme dosiahnuť nízkej spojitosti a vysokej súdržnosti.[14]

2.1.4 Zložitosť

Zložitosť skúma ako zrozumiteľný a jednoduchý je daný procesný model. Pri softvérových programoch sa používa miera, ktorú vyvinul Thomas J. McCabe v roku 1976. Nesie názov cyclomatic complexity. Metrika počíta počet lineárne nezávislých ciest cez zdrojový kód programu. Táto miera zložitosti sa dá aplikovať aj na procesy. Výsledné ohodnotenie je znázornené v tabuľke 2.2.

Počet nezávislých ciest	**Zložitosť modelu**
1 - 10	jednoduchý
11 - 20	mierne komplexný
21 - 50	zložitý
nad 50	netestovateľný

Tabuľka 2.2: Hodnotenie podľa miery cyclomatic complexity

V roku 2005 bola definovaná miera *Control-Flow Complexity* (CFC), ktorá bola odvodená zo softvérového inžinierstva. CFC hodnotí zložitosť v procese podľa výskytu XOR, OR a AND brán (rozdeľovačov). Pri XOR bráne, sa CFC rovná hodnote fan-out danej brány. Fan-out znamená počet vstupov do brány. U OR je to hodnota $2^n - 1$, kde n je tak isto fan-out. Pre AND bránu platí, že CFC je rovný 1. *Control-Flow Complexity* je aditívna, čo znamená, že výsledná hodnota sa jednoducho spočíta sčítaním CFC všetkých brán. Čím väčšia je hodnota tejto miery, tým zložitejší je proces.[14]

2.1.5 Modularita

Rozdelenie procesného modelu do pod-modelov môže pomôcť pri porozumení modelu a taktiež týmto procesom vznikajú menšie modeli, ktoré sú zase použiteľné v iných nad-modeloch. Použiteľná v procesných modeloch je softvérová miera, ktorú navrhli Henry a Kafura. Miera využíva počet lokálnych informácií

vstupujúcich do modulu programu (fan-in), a počet lokálnych informácií opúšťajúcich modul (fan-out). Vzorec pre výpočet miery modelu sa dá definovať ako:

$$modularization = (in * out)^2$$

Kde *in* (fan-in) sčítava všetky podprocesy volajúce skúmaný proces a *out* (fan-out) je počet podprocesov, ktoré sú volané z daného procesu. Podľa tejto miery náročnosť používania procesu stúpa s veľkosťou modularity.[16]

2.2 Nástroje hodnotiace kvalitu procesných modelov

Na výpočet mier kvality procesného modelu existujú rôzne nástroje. Podarilo sa mi nájsť nasledujúce z nich.

2.2.1 ProM tool

Na rozdiel od väčšiny ostatných nástrojov, ProM sa zaoberá hlavne analýzou reálnych procesov a porovnáva ich s tými namodelovanými. Používa tzv. process mining (alebo tiež „dolovanie procesov“). Process mining umožňuje analýzu podnikových procesov na základe zaznamenaných udalostí (anglicky „event logs“). Myšlienkou dolovania je získavanie informácií o udalostiach z informačných systémov. Process mining má za cieľ zlepšiť získavanie týchto dát.

Okrem dolovania má ProM niekoľko modulov, kde sú implementované aj miery kvality modelu. Sú to hlavne zložitosť a veľkosť, ale tiež spojitosť a súdržnosť. ProM obsahuje modul na spočítanie mier hustoty modelu, weighted coupling a control-flow complexity. Ďalším je modul pre vypočítanie súdržnosti a spojitosti. Tento modul sa venuje výpočtu týchto mier podľa Reijersa a Vanderfeestena a teda poskytuje aj „*coupling-cohension ratio*“.[14]

2.2.2 CoCoFlow

Hlavnou funkcionalitou nástroja CoCoFlow (COhesion-COupling metrics for workFLOW models) je výpočet mier daného procesu a navrhnutie jeho najlepšieho dizajnu. Už podľa názvu je vidieť že CoCoFlow sa zameriava na miery súdržnosti (cohension) a spojitosti (coupling). Užívateľské rozhranie tohto nástroja pozostáva z troch častí. CoCoFlow pracuje s XML súbormi. V prvej časti užívateľovi umožňuje pôvodný súbor čítať a meniť priamo v programe. Druhá časť je zameraná na grafickú vizualizáciu modelu. Posledná časť sa venuje mieram kvality a najvhodnejšiemu návrhu procesu.[17]

2.2.3 BPMN Quality

BPMN Quality je nástroj implementovaný v jazyku Java. Pozostáva zo štyroch hlavných modulov. Prvý modul (nazvaný *extractor*) dostáva ako vstup XMI súbor s procesným modelov a následne si z nich načíta všetky prvky modelu. Použitie tohto štandardu zaručuje integráciu nástroja s inými modelovacími nástrojmi podporujúcimi XMI. Výstup sa predáva ďalšiemu modulu (*constructor*). Ten dáta spracuje a vytvorí strom prvkov podľa perspektívy, ktorú si volí používateľ.

Modul *calculator* implementuje všetky výpočty mier kvality. Sú to miery súdržnosti, spojitosti a zložitosti. Pracuje s informáciami z výstupu predošlého modulu (modul *constructor*) a mierami, ktoré si zvolil používateľ. Výsledok uloží ako XML súbor. Nakoniec vyhodnotí posledný modul (*interpreter*) konečnú kvalitu procesného modelu.[18]

2.2.4 BPMN Measures

Program BPMN Measures je vyvíjaný v programovacom jazyku Java. Ako vstup sa používajú procesné modely vo formáte XPDL. Funkcionalita programu pozostáva z troch tried zameriavajúcich sa na výpočet procesných mier, validáciu vstupných súborov a integráciu do webových služieb.

Tento nástroj dokáže vypočítať hodnoty 10 procesných mier – napríklad počet aktivít, control-flow complexity, cyclomatic number alebo fan-in a fan-out.[16]

2.2.5 Zhodnotenie nástrojov

Z vyššie spomenutých nástrojov by sme holi oddeliť ProM od ostatných. ProM sa ujal hodnotenia procesných modelov cez process mining (dolovanie procesov). Aj keď implementuje aj niekoľko mier kvality modelu, ich úloha v programe je skôr vedľajšia. Najmenší počet mier počíta CoCoFlow, ktorý zhodnotí miery súdržnosti a spojitosti. Za to najviac ich meria BPMN Measures.

Žiadny z týchto nástrojov ale v svojom hodnotení nepočíta pri hodnotení modelu s jeho grafickým zobrazením. V mojom hľadaní sa mi takýto nástroj nepodarilo nájsť.

Kapitola 3

Počítačové videnie

V mojej práci chcem dostávať informácie o procesnom modeli z jeho grafickej podoby- diagramu. Na tento prístup je potrebné poznať oblasť počítačového videnia.

[19] definuje počítačové videnie takto: *„Počítačové videnie je disciplína, ktorá sa snaží technickými prostriedkami aspoň čiastočne napodobniť ľudské vnímanie“*.

Pri vyhodnocovaní vizuálnej informácie sú dôležité znalosti a poznatky človeka o okolitom svete. Pre počítačové videnie je typická snaha porozumieť všeobecnej trojrozmernej scéne. V mojej časti problematiky sa jedná o procesné diagramy. Bude to teda ľahšie ako zaznamenávať trojrozmerný svet, ale v princípe ide o podobné problémy. Aké tvary sa snaží človek zaznamenať pri pohľade na diagram? Aké informácie sú pre nás z obrazu potrebné?

Počítačové videnie sa dá rozdeliť na 2 časti. Jadrom pokročilejších postupov sú znalostné systémy a techniky umelej inteligencie, ktoré patria do tzv. vyššej úrovne. Druhou časťou je nižšia úroveň počítačového videnia. Cieľom tejto úrovne je analýza dvojrozmerných obrazových dát, ktoré dostaneme na vstupe. Nižšia úroveň sa používa napríklad na odstraňovanie šumu z obrazu, rozpoznanie jednoduchých objektov, atď. Táto úroveň sa tiež nazýva spracovanie obrazu (z anglického image processing).[19]

Spracovanie a rozpoznanie obrazu reálneho sveta sa dá rozložiť do nasledujúcich krokov:

1. Snímanie (vytvorenie), digitalizácia a uloženie obrazu v počítači.
2. Predspracovanie.
3. Segmentácia obrazu na objekty.
4. Popis objektov.
5. Klasifikácia objektov (porozumenie obsahu).

Obraz je uložený ako matica prirodzených čísel. Prvku obrazu sa hovorí obrazový element alebo pixel (z angl. *picture element*). Jedná sa o nedeliteľnú jednotku.(29)

Predspracovanie obrazu predstavuje odstránenie nežiadúcich javov (ako napríklad šum), alebo naopak môže zvýrazniť informácie ktoré sú relevantné (napríklad zvýraznenie hrán).[20]

Ďalším krokom je segmentácia, ktorá má za úlohu nájsť objekty v obraze. Za objekty môžeme považovať tie časti obrazu, ktoré sú pre nás z hľadiska ďalšieho spracovania obrazu zaujímavé.[19]

Popis nájdených objektov je štvrtou časťou spracovania. To, aké vlastnosti budeme popisovať, záleží od hľadaných objektov. Jeden z najjednoduchších popisov je veľkosť objektu, teda počet zodpovedajúcich obrazových bodov (pixelov) objektu v obraze.

Posledným krokom spracovania je klasifikácia. V jednoduchom prípade ide o klasifikáciu objektov podľa ich veľkosti, obvodu, pomeru strán alebo iných vopred známych tried. Pri zložitejších prípadoch sa už ale dostávame do vyššej úrovne počítačového videnia.[19]

3.1 Reprezentácia obrazu

Pre prácu s obrazom je samozrejme potrebné obraz a jeho jasové hodnoty reprezentovať vhodným spôsobom. V [21] sa definuje obraz ako spojitá funkcia dvoch premenných $f(x, y)$, kde f značí hodnotu jasu a (x, y) sú súradnice určujúcu pozíciu v obraze. Hodnotou tejto funkcie môže byť jedno číslo, čo sa vyskytuje najmä v šedotónových obrazoch. Pre farebné obrazy potrebujeme čísel viacero, napríklad pre klasický model RGB potrebujeme tri zložky jasu pre červenú, zelenú a modrú (t. j. tri funkcie).

3.2 Predspracovanie obrazu

„Predspracovanie je spoločný názov pre operácie s obrazom na nízkej úrovni abstrakcie“.[19]

Tieto metódy slúžia k zlepšeniu obrazu z hľadiska ďalšieho spracovania. Použité úpravy sa líšia podľa toho, či s výsledným obrazom pracuje človek, alebo slúži na automatické spracovanie. V priebehu predspracovania nezískavame žiadnu novú informáciu, informácie len potláčame alebo zvýrazňujeme.[19]

Pôvodný obraz s hodnotami jasu funkcie $f(x, y)$ je transformovaný do nového s hodnotami jasov funkcie $g(x, y)$. Táto transformácia T sa dá vyjadriť vzťahom:

$$g(x, y) = T[f(x, y)].$$

T sa môže tiež označiť ako operátor, ktorý je aplikovaný na obraz.[21]

V tejto kapitole zhrniem používané metódy spracovanie obrazu. Najdôležitejšou pre moju prácu je detekcia hrán.

3.2.1 Bodové jasové transformácie

Medzi najjednoduchšie operácia patria bodové jasové transformácie. Ide o zmenu intenzity jasu f v každom jednotlivom bode (x, y) na hodnotu g. Funkcia môže byť určená ešte pred spracovaním obrazu (napr. negatív) alebo sa určí počas spracovania (napr. ekvalizácia histogramu). Operácie pritom môžu využívať informácie o globálnych vlastnostiach obrazu.[21] Príklad takejto transformácie je už vyššie spomenutý negatív alebo ekvalizácia histogramu.

3.2.2 Geometrické transformácie

Geometrické transformácie sa používajú na opravenie obrazu, ak obsahuje zdeformované tvary, alebo ak chceme obrázok zámerne zdeformovať. Medzi najjednoduchšie transformácie patrí posunutie, otočenie alebo skosenie.

[19] definuje geometrickú transformáciu T_g plošného obrazu ako vektorovú funkciu, ktorá transformuje bod (x, y) na bod (x_1, y_1), kde pre zložky platí:

$$x_1 = T_x(x, y), y_1 = T_y(x, y),$$

kde T_x a T_y sú transformačné vzťahy.

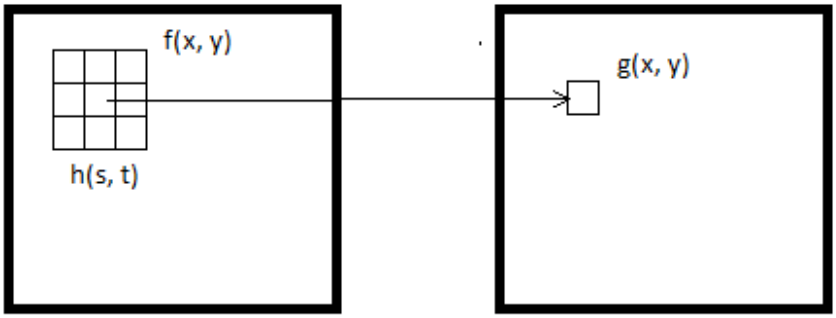
3.2.3 Filtrácie

Lokálne metódy využívajú pre výpočet jasu bodu lokálne okolie daného bodu vo vstupnom obraze. Časti lokálneho predspracovania sa tiež hovorí filtrácia (z teórie signálov). Pri tomto postupe sa ako vstup neberie len jediný bod (x, y), ale aj jeho určité okolie, ako na obr. 3.1. Najčastejšie je filtrácia vykonávaná na princípe tzv. diskrétnej konvolúcie.[19]

Pri konvolúcii je dôležité dobre si zvoliť filtračné jadro (angl. kernel). Ide o maticu nepárnej veľkosti (napr. 3x3, 5x5), aby bola jej stredová pozícia jasne určená. Maska sa posúva po obraze. Pri každom posunutí masky sa každá hodnota v matici vynásobí s príslušnou hodnotou jasovej funkcie v obraze. Súčet všetkých hodnôt sa následne zapíše do bodu výsledného obrazu ktorý zodpovedá stredu jadra konvolúcie a maska sa posunie ďalej po obraze.[20]

Dnes sú u k dispozícii vysoko optimalizované implementácie konvolúcie pre spracovanie obrazu, napr. pre Intel procesory, ktoré používajú známe nástroj ako napr. Matlab a OpenCV.

Podľa cieľu sa filtrácia dá rozdeliť do dvoch skupín – vyhladzovacie metódy a gradientné operácie. Ako sa bude daná metóda správať záleží hlavne na voľbe konvolučného jadra.



Obr. 3.1: Operácie uvažujúce okolie[21]

Vyhladzovacie metódy slúžia na potlačenie vyšších jasových frekvencií obrazu. Slúžia hlavne k potlačeniu šumu, ale ich vedľajším účinkom je aj potlačenie náhlych zmien jasovej funkcie – čo znamená potlačenie ostrých čiar a hrán.[19]

Gradientné operácie zase spôsobujú ostrenie obrazu, t. j. zdôrazňujú vyššie jasové frekvencie. Výsledkom tejto operácie je zvýraznenie hrán a ostrých čiar v obraze, ale zase vedľajším efektom je zvýraznenie šumu.[19]

3.2.3.1 Vyhladzovacie metódy

Medzi vyhladzovacie metódy patrí napríklad vyhladzovanie priemerovaním, Gaussovo rozostrenie alebo mediánová filtrácia.

Priemerovanie patrí k najjednoduchším metódam. Ide o konvolúciu, kde je výslednou hodnotou priemer všetkých okolitých bodov v obraze. Konvolučné jadro je zobrazené na obr. 3.2.

Pri Gaussovom vyhladzovaní sa konvolučné jadro zmení tak, že koeficienty bližšie ku stredu majú väčšiu váhu a zodpovedajú hodnotám na Gaussovej krivke ako v obr. 3.2.

Ako už názov naznačuje, mediánová filtrácia zase počíta medián okolitých hodnôt bodu. Medián je prostredný prvok v usporiadanej postupnosti hodnôt. Táto metóda vykazuje dobré výsledky v špecifických prípadoch šumu, ako napríklad šum typu „korenie a soľ“.[21]

3.2.3.2 Gradientné operácie

Hrana v obraze je vlastnosť obrazového elementu a jeho okolia. Prítomnosť hrany nám indikuje miesto v obraze, kde sa prudko mení hodnota jasu obra-

$\frac{1}{273}$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

$\frac{1}{25}$

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

Obr. 3.2: Ukážka konvolučných jadier pre Gaussovo vyhladzovanie (vľavo) a vyhladzovanie priemerovaním (vpravo)

zovej funkcie $f(x, y)$ - body s veľkým gradientom sa považujú za hrany. Na detegovanie hrán sa používajú hranové operátory.[22]

Výsledkom prvej derivácie obrazu je gradient. Gradient je vektorová veličina a je určená veľkosťou a smerom rastu (jasu) a využíva sa ako informácia pri hľadaní hrán. Na princípe gradientu (prvej derivácie) je založených niekoľko hranových operátorov.

Prewittovej operátor využíva konvolučné jadrá z obrázku 3.3. Prvé jadro (G_x) deteguje zmeny v smere x a druhé (G_y) deteguje zmeny v smere y. Sobelov operátor upravuje tieto hodnoty tak, aby tie, čo sú bližšie pri strede, boli zdôraznené. Ich jadrá sa dajú otáčať po 45 stupňov ako v obrázku 3.3 a tým zvýrazňovať hrany v rôznych smeroch.[21]

Prewitt

Vertical

-1	0	1
-1	0	1
-1	0	1

Horizontal

-1	-1	-1
0	0	0
1	1	1

Sobel

Vertical

-1	0	1
-2	0	2
-1	0	1

Horizontal

-1	-2	-1
0	0	0
1	2	1

Obr. 3.3: Konvolučné jadrá Prewittovej a Sobela v dvoch smeroch[23]

Druhá derivácia predstavuje rýchlosť zmeny jasu v danom bode. Laplaceov operátor aproximuje druhú deriváciu. Jeho operátor je necitlivý voči otočeniu a udáva len veľkosť hrany a nie jej smer. Konvolučné jadro Laplaceovho operátora je ukázané na obrázku 3.4.[21]

Ako jeden z najlepších hranových detektorov sa označuje Cannyho hranový detektor.[22] Detektor má tri kritéria. Prvé zaisťuje, aby boli zazna-

0	1	0
1	-4	1
0	1	0

Obr. 3.4: Konvolučné jadro - Laplaceov operátor[23]

menané všetky významné hrany. Lokalizačné kritérium zase zodpovedá, aby rozdiel medzi skutočnou a nájdenou pozíciou hrany bol minimálny. Tretie kritérium zaisťuje aby tá istá hrana obrazu nebola zaznamenaná viackrát. Detektor hľadá hrany na základe druhej derivácie a jej priechodu nulou a výsledkom je veľkosť a smer hrany.[19]

3.2.4 Morfologické transformácie

Matematická morfológia tvorí pomerne samostatnú oblasť v rámci analýzy obrazu. Morfologické transformácie sú realizované ako relácia obrazu s jej bodovou podmnožinou, ktorej sa tiež hovorí štruktúrny element. Táto podmnožina môže mať rôzny tvar. Dá sa predstaviť, že morfologickou transformáciou je systematický pohyb štruktúrneho elementu po obraze. Výsledok relácie medzi obrazom a štruktúrnym elementom je zapísaný do bodu obrazu ekvivalentnému počiatku štruktúrneho elementu.[19]

3.2.4.1 Dilatácia a erózia

Dilatácia skladá body množín pomocou vektorového súčtu. Má to za následok zväčšenie popredia obrazu. Používa sa to hlavne na zaplnenie malých dier a úzkych zálivov.

Opačný efekt má erózia. Táto transformácia počíta s rozdielom množín obrazu a elementu. Následkom erózie zanikajú malé objekty a väčšie objekty sa zmenšujú. V spracovaní obrazu slúži hlave na oddelenie objektov spojených tenkými čiarami.[19]

3.2.4.2 Otvorenie a uzavretie

Kombináciou predošlých dvoch transformácií vznikajú ďalšie morfologické transformácie - otvorenie a uzavretie. Výsledkom je obraz, ktorý obsahuje menej detailov a je jednoduchší.

Otvorenie tvorí erózia nasledovaná dilatáciou. Naopak uzavretie je dilatácia nasledovaná eróziou. Obe transformácie nám odstránie malé detaily v obraze, pričom celkový tvar objektov sa nezmení. Otvorenie oddelí objekty spo-

jené úzkou líniou, čím zjednoduší štruktúru objektov. Uzavrenie zase spojí objekty blízko seba a zaplní malé diery.[19]

3.3 Segmentácia obrazu

Segmentácia obrazu nám rozdeľuje obraz na časti, ktoré nás zaujímajú, a ich rozlíšenie od pozadia. Patria sem metódy, ktoré sa v obraze snažia nájsť objekty potrebné na ďalšie spracovanie alebo analýzu obrazu. Segmentácia a správny popis objektov patrí medzi najzložitejšie úlohy spracovania obrazu. Metódy segmentácia sa dajú rozdeliť do dvoch skupín. V prvej skupine sú algoritmy, ktoré hľadajú oblasti podľa nejakého kritéria podobnosti. Patria sem algoritmy založené na postupnom rozdeľovaní alebo narastaní oblastí. Druhou skupinou sú algoritmy, ktorý pracujú na základe nájdených hrán, čo je napríklad využitie Houghovej transformácie.[21]

3.3.1 Prahovanie (Thresholding)

Prahovanie rozdeľuje obraz na popredie a pozadie na základe zvoleného prahu T. Prah nám udáva minimálnu hodnotu jasovej funkcie, ktorú môže mať popredie. Výsledkom prahovania je teda binárny (dvojúrovňový) obraz, a jeho jasovú funkciu $g(x, y)$ môžeme definovať takto:

$$g(x, y) = 1 pref(x, y) >= T$$
$$g(x, y) = 0 pref(x, y) < T.$$

Stanovenie prahu je teda pre túto metódu kľúčové. Existujú metódy automatického určovania prahu, ktoré ale vyžadajú dobré oddelenie jasu popredia od pozadia. Využívajú sa aj metódy lokálneho prahovania, ktoré určia samostatnú hodnotu prahu pre jednotlivé oblasti obrazu.[19]

3.3.2 Rozdeľovanie a spojovanie oblastí (Region splitting and merging)

Pri tejto metóde je obraz postupne rozdeľovaný na predom určené oblasti (väčšinou štvorce). Každá oblasť je následne skontrolovaná, či splňuje podmienku – „kritérium rovnorodosti“. Toto kritérium môže byť rôzne – vychádza napr. zo strednej hodnoty jasu, štatistických testov, alebo sa testuje farebný odtieň oblasti apod. Ak sa podmienka splnila, oblasť sa ďalej nedelí. V opačnom prípade sa opäť delí a postup sa opakuje.[21]

3.3.3 Narastanie oblastí (Region growing)

Narastanie oblastí je metóda na opačnom princípe ako predošlá. Na začiatku sa určia štartovacie body podľa určitých podmienok (t. j. pixely s danými

vlastnosťami). Oblasť sa následne zväčšuje, t. j. sú k nej podľa definovaných vlastností pridávané ďalšie a ďalšie body. Keď sa už ďalšie body pridať nedajú, oblasť je kompletná.[21]

3.3.4 Houghova transformácia

Pôvodne bola Houghova transformácia elegantné riešenie ako v obraze nájsť priamky a úsečky. Neskôr bola rozšírená na vyhľadávanie ľubovoľného tvaru, najčastejšie sa používa na nájdenie kruhu alebo elipsy.[21]

3.4 Popis objektov

Predošlé kroky spracovania obrazu nám ako výstup poskytnú obraz rozložený na určité oblasti. K porozumeniu obrazovým dátam potrebujeme ešte tieto oblasti rozpoznať. Rozpoznávanie znamená exaktný popis oblasti tak, aby mohol byť predložený klasifikátoru.[19]

Spôsobom rozpoznávania je mnoho: pozdĺžnosť, kruhovosť, podpisy (*signatures*), reťazové kódy (*chain codes*), momenty a iné. Popíšem niektoré jednoduché metódy podľa [21].

Pravouhlosť a pozdĺžnosť sa dá jednoducho vypočíta opísaním objektu najmenším možným obdĺžnikom. Ak uvažujeme dlhšiu stranu opísaného obdĺžnika označenú a, kratšiu b, pozdĺžnosť L môžeme definovať ako:

$$L = \frac{a}{b}.$$

Pravouhlosť R sa definuje ako:

$$R = \frac{N}{S},$$

kde S je plocha opísaného obdĺžnika a N je plocha objektu.

Kruhovosť C vieme taktiež jednoducho spočítať ako:

$$C = \frac{l^2}{N},$$

kde N je plocha objektu a l je dĺžka hranice objektu.

Kapitola 4

Nástroje počítačového videnia

V tejto kapitole som sa snažil zamerať hlavne na nástroje podporujúce jazyk Java, keďže môj nástroj programujem v ňom. Podarilo sa mi nájsť nasledujúce.

4.1 ImageMagick

Image Magick je slobodný a otvorený (open-source) softvér na vytváranie, upravovanie a konvertovanie obrázkov. Autorom je firma ImageMagick Studio LLC. Vie čítať a zapisovať z viac ako 200 formátov vrátane PNG, JPEG, JPEG-2000, GIF, TIFF, DPX, EXR, WebP, Postscript, PDF, a SVG. Najnovšia verzia je 7.0.5 a podporuje Linux, Windows, Mac OS, iOS a Android.[24]

Funkcionalita knižnice je prístupná cez príkazový riadok alebo v programe cez rozhranie (API). Image Magick má vytvorené API pre 16 programovacích jazykov, každé nesie iný názov. Medzi ne patria napríklad MagickWand (pre jazyk C), Magick++ (pre C++), JMagick (Java), Magick.NET (.NET), IMagick (PHP), PythonMagick (Python), a iné.

Nástroj síce vyniká s prácou s formátmi, ale z oblasti počítačového videnia má oproti ostatným nástrojom menšiu funkcionalitu. Dokáže vykonať niektoré jasové transformácie, morfologické operácia, geometrické transformácie a Fourierovu transformáciu. Okrem toho implementované algoritmy majú väčšinou menej vstupných argumentov na ich ovládanie.[24]

Dokumentácia Magic++ a JMagick je len zoznam funkcií a parametrov. Parametre aj ked ich je minimálne, sú často krát málo vysvetlené alebo aj nevysvetlené. Funkcie sú len krátko popísane. Pri funkciách, ktoré si vyžadujú aj spôsob ich implementácie, tento spôsob nie je vôbec zdokumentovaný.

4.2 ImageJ

ImageJ je otvorená (open-source) knižnica čisto pre jazyk Java od Wayna Rasbanda. Ponúka možnosti spracovania, úpravy a vylepšenia obrazu. V [25]

seba uvádzajú ako najrýchlejšiu čisto Java knižnicu pre spracovanie obrazu s filtráciou 40 milónov pixelov za sekundu. Podporuje formáty TIFF (nekomprimovaný), GIF, JPEG, BMP, PNG, PGM a FITS.

Nástroj tiež podporuje základné funkcie spracovania obrazu, ako sú manipulácia kontrastu, vyhladzovanie, ostrenie, detekcia hrán. Ovláda tiež geometrické transformácie. ImageJ tak isto umožňuje pridávať ďalšie moduly s prídavnými funkcionalitami.[25]

4.3 BoofCV

Autorom BoofCV je Peter Abeles. BoofCV je voľne dostupná knižnica v jazyku Java pre počítačové videnie a robotiku. Podporuje len jazyk Java a teda je multiplatformová.

Knižnica je rozdelená do viacerých balíkov: *image processing* (spracovanie obrazu), *features* (črty), *geometric vision*, *calibration* (kalibrácia), *recognition* (rozpoznávanie), *visualize* (vizualizácia) a *IO*.

Image processing obsahuje používané metódy pre spracovanie obrazu, ktoré priamo pracujú z pixelmi obrazu. *Features* sa venuje algoritmom na extrakciu čŕt. *Calibration* obsahuje metódy kalibrácie kamery. *Recognition* sa venuje rozpoznávaniu zložitých objektov. *Visualize* má funkcie pre zobrazuje nájdených čŕt a *IO* sa venuje vstupom a výstupom pre rôzne dátové štruktúry.[26]

4.4 OpenCV

OpenCV je najpopulárnejšia voľne dostupná knižnica pre počítačové videnie a spracovanie obrazu. Je k dispozícii pod BSD licenciou, a tým je dostupná pre akademická a komerčné účely. Obsahuje viac ako 500 implementovaných algoritmov pre analýzu obrazu a videa. Tento nástroj bol navrhnutý hlavne na výpočtovú efektivitu a jeho algoritmy sú dobre optimalizované. Je napísaný v C++, čo tiež zlepšuje výkon oproti knižiciam v Jave. Podporuje jazyky C, Python a Java a je kompatibilný s väčšinou operačných systémamov: Windows, Linux, Mac OS, iOS a Android.[27]

OpenCV je vyvíjaná od roku 1999 firmou Intel a na jej vývoji sa zúčastnilo veľké množstvo ľudí. Hlavná komisia sa skladá zo 7 ľudí, profesorov z rôznych univerzít.[28]

OpenCV podporuje 14 obrazových formátov: BMP, JPEG, JPG, PNG, TIFF, RAS, DIB, JPE, JP2, PBM, PGM, PPM, SR, TIF. Je potrebné ale poznamenať, že nie všetky formáty sú podporované všetkými operačnými systémami.

Štruktúra OpenCV pre jazyk Java je zložená z viacerých modulov, kde každý plní inú funkciu. Popíšem ich podľa [27].

Core je „jadro“ OpenCV. Obsahuje základné dátové štruktúry, ako napríklad *Mat* (viacrozmerné pole hlavne na reprezentáciu obrazu), *Rect* (trieda pre reprezentáciu obdĺžnika) a základné funkcie, ktoré využívajú ostatné moduly.

Imgproc slúži na spracovanie obrazu. Obsahujé funkcie pre morfologické operácie, prahovanie, hranové operátory (Sobel, Laplace, Canny), Houghovu transformáciu, geometrické transformácie, jasové operácie, a ďalšie iné.

Highgui je rozhranie pre prácu s video a obrazovými súbormi v rôznych formátoch (hlavne načítanie a ukladanie). Modul tiež umožňuje vytvorenie jednoduchého užívateľského rozhrania. Novšie verzie OpenCV (od 3.0.0) pre Javu tento modul neobsahujú a jeho funkcionalita bola rozdelná do modulov Imgproc a Video.

Video obsahuje funkcie pre analýzu videa.

Features2d je hlavný modul na detekovanie čŕt (Feature detection) a popis objektov (Descriptors).

Objdetect obsahuje funkcie na detekciu preddefinovaných objektov, ako napr. tváre, oči, autá, ľudia a iné.

Calib3d slúži na rekonštrukciu 3D scény a kalibráciu (Camera calibration).

4.4.1 Základné štruktúry

OpenCV používa na prácu s obrazom triedu *Mat*. Mat sa používa hlavne na ukladanie matíc a šedotónych alebo farebných obrazov. Je to vlastne n-rozmerné pole s jedným alebo viacerými kanálmi. Napríklad pre šedotónové obrazy sa používa pole s jedným kanálom pre odtieň šedej.

Pre definovanie typu instancie Mat sú preddefinované hodnoty z triedy *CvType*, napríklad *CV_8UC1* znamená jeden 8-bitový kanál.

Na uloženie bodu sa používajú triedy *Point* resp. *Point3*, ktoré slúžia na ukladanie bodu v 2-rozmernom resp. 3-rozmernom súradnicovom systéme určenom osymi x, y a z.

Trieda *Rect* v sebe uchováva informácie o obdlížniku, ktore zložky x a y a sú súradnice bodu najbližšieho k počiatku súradnicového systému a *width* a *height* sú šírka (veľkosť obdĺžnika podľa osi x) a výška (podľa osi y).

Typ *Scalar* sa používa ako 4-prvkový vektor na ukladanie farby pixelov, napríklad pri RGB sa použijú prvé tri hodnoty pre definovanie hodnoty jasu zložiek červenej, zelenej a modrej farby.

Size je jednoduchá trieda pre ukladanie veľkosti obrazu alebo obdĺžnika. Obsahuje hodnoty *width* (šírka) a *height* (výška).

4.4.2 Filtrovanie

OpenCV má implementované morfologické operácie - eróziu, dilatáciu, otvorenie aj uzavretie. Na eróziu a dilatáciu slúžia funkcie *erode* a *dilate*, otvorenie a uzavretie sa vykonáva metódou *morphologyEx* s parametrom *MORPH_OPEN* pre otvorenie a *MORPH_CLOSE* pre uzavretie. Všetky tieto funkcie vyžadujú ako parameter štruktúrny element, ktorý sa vytvorí funkciou *getStructuringElement* - ktorý vie vytvoriť rôzny typy (štvorec, kríž, elipsa) a veľkosti štruktúrnych elementov podľa parametrov.

Ďalej obsahuje funkcie Gaussovo vyhladzovanie, ktorá má v OpenCV názov *gausianBlur*, Cannyho hranový detektor, volaný príkazom *Canny* s parametrami minimálneho a maximálneho prahu pre určenie hrany, a *cvtColor*, ktorá slúži konvertovanie obrazu z jedného farebného spektra do iného (definované parametrom, napríklad *COLOR_BGR2GRAY* znamená zmenu z BGR spektra do šedotónového).

4.4.3 Segmentácia obrazu

Na segmentáciu obrazu v mojom nástroji som spočiatku používal Houghovu transformáciu, pre detekciu čiar a kruhov. V OpenCV na to slúžia funkcie *houghLines* a *houghCircles*. Avšak zistil som, že OpenCV obsahuje zaujímavý a rýchly algoritmus na segmentáciu čiar a kriviek z obrazu. Je to funkcia *findContours*, ktorá obsahuje implementáciu algoritmu nasledovania hranice v binárnom obraze, ktorý vytvorili Suzuki a Abe a popísali v [29]. Výstupom funkcie je pole prvkov, kde každý prvok je jeden obrys nájdený algoritmom. Obrys je typu *MatOfPoint*, v ktorom sú body, ktoré určujú hranicu. Hranice je popísaná úsečkami a výstupné body sú teda koncové body úsečiek, ktoré tvoria spojitú hranicu.

4.5 Zhodnotenie nástrojov

Všetky nástroje z rešerše sú voľne dostupné. Najobsiahlejšiu funkcionalitu z popísaných nástrojov má jednoznačne OpenCV. ImageMagick síce podporuje najviac formátov, ale zaostáva funkcionalitou a jeho dokumentácia je nedostačujúca. Rýchlosťou napreduje OpenCV a ImageJ, avšak v mojej práci nebude na rozdiele rýchlosti až tak záležať, pretože sa venujem len detekcii základných tvarov. Ak by som porovnával popularitu nástrojov, tak vedie OpenCV. Väčšina prác a diskusií, na ktoré som narazil sa odkazuje práve na tento nástroj.

Kapitola 5

Implementovaný nástroj

Záverčnou časťou mojej práce je implmentácia nástroja, ktorý bude vedieť s pomocou spracovania obrazu a počítačového videnia detegovať prvky diagramu a následne zhodnotiť procesný diagram zo získaných informácií. V predošlej kapitole sú zhrnuté nástroje počítačového videnia - ImageMagick, BoofCV, ImageJ a OpenCV. Podľa zadania tejto práce som si mal vybrať jeden z nástrojov, ktorý použijem pri implementácii. Na základe zistených poznatkov z predošlých kapitol a konzultáciou s vedúcim práce, som sa rozhodol nástroj implementovať pomocou knižnice OpenCV - najnovšou verziou *3.2.0*. OpenCV napreduje funkcionalitou aj popularitou. Implementuje najväčšie množstvo algoritmov, má veľkú užívateľskú základňu a je stále vyvíjaný.

Ako vstup budem načítať diagramy vytvorené v BPMN notácii. BPMN je dnes najpoužívanejší štandard pre procesné diagramy. V ďalšej sekcii vysvetlím metódy použité v mojej implementácii a názorne ich ukážem.

5.1 Postup pri implementácii

Notáciu BPMN som popisoval v sekcii 1.3.4. Hlavné prvky, ktoré chcem v diagrame nájsť sú aktivity, udalosti, brány, a sekvenčné toky. Považujem ich za hlavné prvky diagramu a poskytujú najdôležitejšie informácie. Implementáciu som prispôsobil diagramom exportovaným z programu Bizagi Modeler, ktoré som používal aj na testovanie nástroja.

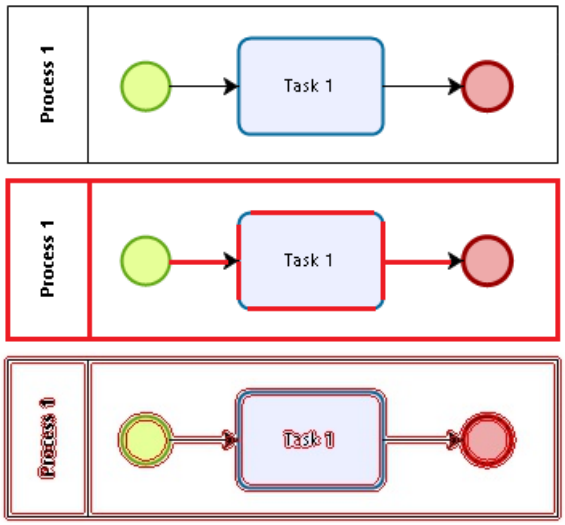
5.1.1 Aktivity

Na nájdenie aktivít je potrebné detegovať obdĺžnik s oblými rohmi.

Prvotná implementácia spočívala vo využívaní Houghovej transformácie. Pomocou jej implementácia v podobe funkcie *houghLines* v OpenCV som našiel hrany obdĺžnika. Hlavný problém bol, že funcia slúžila na detekciu rovných čiar, a zaoblené rohy vynechávala. Výstup *houghLines* je naznačený na obrázku 5.1. Predĺžil som úsečky, počítal priesečníky úsečiek a tým hľadal 4

úsečky ktoré mohli tvoriť obdĺžnik. Avšak stále som nemal informáciu o či je medzi nimi zaoblený roh. Navyše v špeciálnych prípadoch mohol sekvenčný tok pôsobiť ako aktivita.

Rozhodol som sa preto použiť metódu *findContours*, ktorá opíše hranicu objektov. Kedže sekvenčný tok spája väčšinu prvkov, diagram je spojitý. *FindContours* teda opíše vonkajšiu hranicu všetkých prvkov spojených sekvenčným tokom. Vnútro aktivít je farebne odlíšené, takže funkcia nájde aj vnútornú hranicu. Výstup funkcie *findContours* je naznačený na obrázku 5.1. Na nájdenie hraníc som použil práve vnútornú hranicu aktivity.



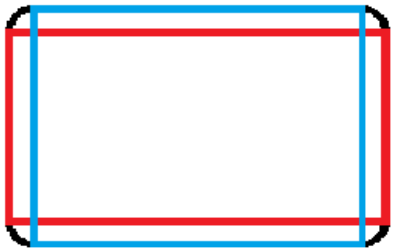
Obr. 5.1: Výstup po vyznačení nájdených oblastí po funkciách *houghLines* a *findContours*. Prvý je originál, potom Houghova transformácia čiar a posledný je algoritmus nasledovania hranice.

Na to, aby som našiel kontúry je potrebné najprv aplikovať Cannyho detektor hrán, ktorý má ako výstup binárny obrázok s hranami v popredí. Hrany následne zväčším pomocou dilatácie, pre lepšiu detekciu kontúr.

Aby som popísal len tie kontúry, ktoré sú hranicami aktivít, musel som nájsť správne vlastnosti, ktoré ich odlíšia od iných tvarov. Najprv aplikujem na kontúru metódu *approxPolyDB*. Metóda implementuje Douglasov-Peuckerov algoritmus, ktorý aproximuje tvar jednoduchším tvarom s menej bodmi. Tým dosiahnem že sa body pri zaoblenom rohu spoja do jedného. Ak po tejto operácii bude mať kontúra 4 body, mohla by byť aktivitou, ale môže to byť aj iný tvar, ktorý sa skladá zo 4 skupín vrcholov pri sebe.

Ďalej je potrebné, aby som nebral v úvahu bazény a dráhy, ktoré majú podobu obdĺžnikov. Kontúre opíšem obdĺžnik a kružnicu, a skontrolujem či vrcholy obdĺžnika sú mimo kružnice - ak nie sú, nemôže to byť aktivita, pretože zaoblené rohy aktivity spôsobia, že rohy opísaného obdĺžnika budú ležať mimo

kružnice. Nakoniec opísanému obdĺžniku zmenším šírku, resp. výšku tak, aby nezasahoval do zaoblených rohov (ako na obrázku 5.2) a skontrolujem, či patrí kontúre. Ak tieto obdĺžniky patria kontúre, tak to znamená, že daný tvar môžem označiť za aktivitu.



Obr. 5.2: Aproximácia aktivity pomocou zmenšenia opísaného obdĺžnika.

5.1.2 Udalosti

Udalosti majú v diagrame tvar kružnice. Na nájdenie kruhu som použil Houghovu transformáciu - funkciu *houghCircles*. Ako som už popísal v kapitole 1.3.4, stredné aktivity sú značené dvomi kružnicami. Vnútorná je o niečo menšia a má rovnaký stred ako vonkajšia. Funkcii *houghCircles* vieme predať parametre, ktoré nám nájdu kružnicu s určitým polomerom. Diagram prechádzam viackrát, aby som našiel prípadné vnútorné kružnice a zistil, či je udalosť stredná.

5.1.3 Brány

Brány sú zobrazené štvorcom, ktorý je otočený o 45 stupňov. Na jeho detekciu som použil taktiež metódu *findContours*, na nájdené kontúry zase metódu *minAreaRect* ktorá opíše kontúre najmenší možný obĺžnik (môže byť aj rotovaný). Na to, aby bola kontúra brána musí splniť tieto podmienky:

- Opísaný obdĺžnik musí byť približne štvorec.
- Opísaný obdĺžnik musí byť rotovaný o 45 stupňov.
- Obsah kontúry a opísaného obdĺžnika musí byť približne rovnaký.
- Uhlopriečka obdĺžnika a polomer opísanej kružnice je približne rovnaký.
- Po aproximovaní tvaru musí mať kontúra 4 vrcholy.

Posudzovaním približnej veľkosti sa vyhnem problémom pri prípadnom posune pixelov pri práci s obrazom. Väčšinou je to 2-6 pixelov podľa veľkosti nájdenej kontúry.

5.2 Popis nástroja

Hlavnú funkcionalitu aplikácie tvorí trieda *ElementDetector.java*, ktorá detekuje aktivity, udalosti aj brány pre triedu *Analyzer.java*, ktorá vypočíta hodnoty daných mier. V nasledujúcej ukážke kódu by som chcel popísať funkciu *detectActivities*, ktorá nachádza v diagrame aktivity.

```
1  public Vector<MatOfPoint> detectActivities() {
2     Vector<MatOfPoint> activities = new Vector<MatOfPoint>();
3     Mat temp = new Mat();
4     Imgproc.Canny(image, temp, 50, 150); //Cannyho hranovy detektor
5     Imgproc.dilate(temp, temp, Imgproc.getStructuringElement(Imgproc.MORPH_CROSS,
          new Size(3.0,3.0))); //dilatacia
6     List<MatOfPoint> contours = new Vector<MatOfPoint>();
7     Imgproc.findContours(temp, contours, new Mat(), Imgproc.RETR_LIST,
          Imgproc.CHAIN_APPROX_SIMPLE); //alg. nasledovania hranice
8     for (int i = 0; i < contours.size(); i++) {
9        MatOfPoint2f approx = new MatOfPoint2f();
10       MatOfPoint2f cont = new MatOfPoint2f(contours.get(i).toArray());
11          Imgproc.approxPolyDP(cont, approx, CORNER_LENGTH, true); //aproximacia
12          if(approx.toList().size() != 4)
13             continue; //po aproximacii nema 4 vrcholy
14       Rect bound = Imgproc.boundingRect(contours.get(i)); //opisanie obdlznika
15       Point mid = new Point();
16        float[] radius = new float[1];
17        Imgproc.minEnclosingCircle(new MatOfPoint2f(contours.get(i).toArray()),
             mid, radius);
18          if(isInside(new Circle(mid, radius[0]), new Point(bound.x, bound.y),
               bound.width, bound.height) != -1)
19             continue; //rohy opisaneho obdlznika nie su mimo kontury
20          if(isInside(cont, new Point(bound.x+CORNER_LENGTH, bound.y),
               bound.width-CORNER_LENGTH, bound.height) != 0)
21             continue; //opisany obldznik zmenseny v~sirke nie je v~konture
22          if(isInside(cont, new Point(bound.x, bound.y+CORNER_LENGTH),
               bound.width, bound.height-CORNER_LENGTH) != 0)
23             continue; //opisany obldznik zmenseny vo vyske nie je v~konture
24          activities.add(contours.get(i));
25    }
26    return activities;
27 }
```

Metóda pracuje so šedotónovým obrázkom (premenná *image*), na ktorý aplikuje Cannyho hranový detektor a dilatáciu nájdených hrán. Následne vyhľadáva v obrázku kontúry, ktoré prechádza vo for cykle. Je potrebné ich previesť z *MatOfPoint* na *MatOfPoint2f*, pretože funkcia aproximácie kontúr počíta s 32-bitovými číslami typu float (*MatOfPoint* zase s int). Po aproximácii sa skontrolu, či obsahuje 4 vrcholy. Ďalším príkazom sa opíše okolo kontúry obdĺžnik a kružnica. Nasleduje séria príkazov if, kde sa skontrolujú podmienky uvedené v sekcii 5.1.1. Využíva sa pri tom mnou vytvorená funkcia *isInside*, ktorá vracia -1 ak sú všetky body obdĺžnika mimo kontúry (resp. kružnice), 0

ak sú na jej hrane, 1 ak sú vnútri. Ak sú podmineky splnené kontúra sa pridá medzi aktivity.

Ďalšou ukážkou by som chcel popísať nachádzanie aktivít v diagrame.

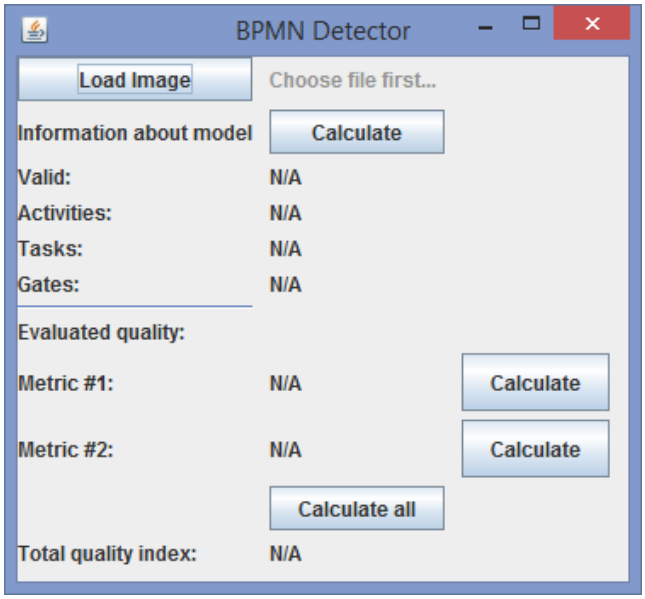
```
public Vector<MatOfPoint> detectGates() {
   Vector<MatOfPoint> gates = new Vector<MatOfPoint>();
   Mat temp = new Mat();
   Imgproc.Canny(image, temp, 50, 150); //Cannyho hranovy detektor
   Imgproc.dilate(temp, temp, Imgproc.getStructuringElement(Imgproc.MORPH_CROSS,
       new Size(3.0,3.0))); //dilatacia
   List<MatOfPoint> contours = new Vector<MatOfPoint>();
   Imgproc.findContours(temp, contours, new Mat(), Imgproc.RETR_LIST,
       Imgproc.CHAIN_APPROX_SIMPLE); //alg. nasledovania hranice
   for(int i=0; i<contours.size(); i++) {
      MatOfPoint2f approx = new MatOfPoint2f();
      MatOfPoint2f cont = new MatOfPoint2f(contours.get(i).toArray());
      Imgproc.approxPolyDP(cont, approx, 3, true); //aproximacia
        if(approx.toList().size() != 4)
          continue; //po aproximacii viac ako 4 vrcholy
        RotatedRect rotated = Imgproc.minAreaRect(cont); //opisanie rotovaneho
            obdlznika
        Point mid = new Point(); //stred opisanej kruznice
        float[] radius = new float[1]; //polomer
        Imgproc.minEnclosingCircle(cont, mid, radius); //opisanie kruznice
        double diag = Math.sqrt(Math.pow(rotated.size.width,
            2)+Math.pow(rotated.size.height, 2));
        double area = Imgproc.contourArea(contours.get(i));
        if(area < 50)
          continue; //min. velkost
        if(Math.abs(rotated.size.width - rotated.size.height) > 4)
          continue; //je pribl. stvorec
        if(Math.abs(area - rotated.size.width*rotated.size.height) > 30)
          continue; //pribl. rovnaky obsah obdlznika a kontury
        if(rotated.angle != -45 && rotated.angle != 45)
          continue; //otocenie o~45 st.
        if(Math.abs(radius[0]*2 - diag) > 4))
          continue; //uhlopriecka a priemer pribl. rovnake
        gates.add(contours.get(i));
   }
   return gates;
}
```

Spracovanie obrazu je rovnaké ako v predošlej funkcii. Tak isto sa skontroluje počet vrcholov po aproximácii. Avšak teraz pracujem s opísaným rotovaným obdĺžnikom, vypočítam jeho uhlopriečku a oblasť kontúry. Potom skontrolujem podmienky pre detekciu brány spomenuté v sekcii 5.1.3.

Výstupy funkcií idú do triedy *Analyzer.java*, ktorá obsahuje funckie *calcNOA* a *calcNOG* na vypočítanie jednoduchých mier podľa počtu aktivít a rozhodovacích brán. Aplikácia umožňuje vypočítať len jednoduché miery kvality. Vypočíta miery na základe počtu aktivít a počtu brán - všetky dosahujú hodnôt od 0 po 10, kde 10 znamená, že model má príliš veľa elementov. Na zá-

klade týchto mier spočíta celkovú kvalitu programu jednoduchým priemerom týchto hodnôt.

Aplikácii má vytvorené aj užívateľské rozhranie, ktoré pozostáva z jednoduchých operácií. Dokáže načítať obrázok zo súboru, následne zistiť základné informácie o modeli, ako sú počet udalostí, aktivít a brán, a vypočítať hodnoty mier kvality. Ukážka grafického rozhrania je na obrázku 5.3.



Obr. 5.3: Užívateľské rozhranie aplikácie.

Grafické rozhranie je vytvorené v Java Swing za pomoci IntelliJ IDEA (vygenerovanie tlačítiek a napísov).

5.3 Testovanie

Na testovanie nástroja som použil modely exportované z programu Bizagi Modeler vo formáte PNG. Inšpiroval som sa modelmi z [**?**], aby som vytvoril zložitejšie diagramy.

Prvý diagram je jednoduchší a zobrazuje výkon taxislužby (príloha B). Diagram zachytáva troch účastníkov - zákazníka, kanceláriu a garáž. Zákazník žiada kanceláriu o službu. Ak je žiadosť schválená, pridelí sa mu auto, ak nie, proces je ukončený neschválenou žiadosťou. Pri úspešnom pridelení sa vykoná transport a proces sa ukončí. V opačnom prípade sa žiadosť vráti do časti schvaľovania. Výstup môjho programu je zhrnutý v tabuľke 5.1.

Druhý diagram, zobrazený v prílohe C, je o niečo zložitejší. Ukazuje proces podávania žiadosti o prácu úchadzačmi a proces príjímania uchádzačov firmou. Uchádzač vytvára žiadosť o prácu a následne ju zasiela firme, ktorá ju

vyhodnotí. Na základe toho procesy buď končia, alebo pokračujú pohovorom. Po pohovore môže firma vytvoriť uchádzačovi ponuku alebo ho odmietnuť, čím končí proces. Pri kladnej odpovedi uchádzač zhodnotí ponuku a opäť buď prijme alebo nie. V obidvoch prípadoch už finálne končia obidva procesy. Výstup môjho programu je zhrnutý v tabuľke 5.2.

Prvok	**Reálny počet**	**Zaznamenaný počet**
Aktivity	6	6
Udalosti	3	3
Brány	4	4

Tabuľka 5.1: Výstup nástroja pri diagrame „objednanie taxislužby“.

Prvok	**Reálny počet**	**Zaznamenaný počet**
Aktivity	15	15
Udalosti	10	10
Brány	3	3

Tabuľka 5.2: Výstup nástroja pri diagrame „žiadosť o zamestnanie“.

Aplikácia správne zaznamenala všetky prvky čo mala.

5.4 Zhodnotenie implementácie

Môjmu nástroju sa úspešne darí detegovať prvky diagramu a následne vypočítať základné miery kvality. Keďže program nedeteguje tok v procese, miery sú vypočítané len na základe počtu prvkov. Tieto miery teda nemajú veľkú váhu. Pre zložitejšie miery by bolo potrebné brať v úvahu aj napr. sekvenčný tok, tok správ, rozhodovanie sa v modeli alebo iné vlastnosti. Skúšal som pár implementácii detegovania sekvenčného toku, avšak neúspešne. Najväčší problém pri detegovaní toku je, že šípka, ktorá ho zobrazuje, môže meniť smer a prekrývať sa s inými šípkami.

Záver

Cieľom mojej bakalárskej práce bolo zhrnúť používané štandardy pri modelovaní procesov, zoznámiť sa s mierami kvality procesných modelov a nástrojami, ktoré ich počítajú. Taktiež som mal zosumarizovať známe nástroje počítačového videnia, ktoré sa dajú využiť pri detekcii symbolov procesného diagramu. V záverečnej časti som dostal za úlohu navrhnúť a implementovať nástroj, ktorý dokáže rozpoznať prvky procesného diagramu a diagram následne zhodnotiť.

Dané ciele sa mi z väčšej časti podarilo splniť. V úvode teorickej časti definujem pojmy proces, procesný model a procesné riadenie. Pokračujem popísaním používaných štandardov procesných modelov, kde som sa zameral hlavne na grafické notácie. V ďalšej časti som sa oboznámil s hodnotením kvality procesného modelu, používanými mierami a nástrojmi, ktoré tieto miery počítajú. V mojom hľadaní sa mi nepodarilo nájsť nástroj, ktorý by proces hodnotil po základe grafickej stránky diagramu, ako to robím v tejto práci. Pre môj účel bolo potrebné sa taktiež oboznámiť s pojmom počítačové videnie a metódami, ktoré sa v ňom používajú. Tie som popísal v 3. kapitole práce. Touto kapitolou som nadviazal na rešerš existujúcich nástrojov implementujúcich metódy z oblasti počítačového videnia. Zameral som sa hlavne na nástroje podporujúce jazyk Java - ImageMagick, ImageJ, BoofCV a OpenCV.

Pre praktickú časť som sa rozhodol použiť knižnicu OpenCV a rozpoznávať diagramy štandardu BPMN. Navrhol som rozpoznávanie prvkov diagramu - aktivít, udalostí a brán. V poslednej časti práce popisujem mnou vytvorený nástroj pre účel detekcie elementov diagramu. Aplikácii sa úspešne darí rozpoznávať základné prvky diagramu a vypočítavať miery podľa ich počtu. Zdrojové kódy sú umiestnené v priloženom CD.

Môj nástroj je základným využitím počítačového videnia na hodnotenie procesných diagramov. Táto oblasť je určite rozšíriteľná a dala by sa zlepšiť, čím by som sa rád v budúcnosti zaoberal.

Literatúra

[1] Management Mania: Podnikový proces (Business process). [online]. 2016, [cit. 2017-5-1]. Dostupné z: https://managementmania.com/sk/business-process-podnikovy-proces

[2] Řepa, V.: *Procesně řízená organizace.* GRADA Publishing, a.s., 2012, ISBN 978-80-247-4128-4.

[3] Carda, A.; Kunstová, R.: *Workflow - Řízení firemních procesu.* GRADA Publishing, spol. s.r.o., 2001, ISBN 80-247-0200-2.

[4] Muller, M.: Business process management (první část). *IT Systems*, ročník 10, Október 2007.

[5] Karvaš, O.: *Využitie modelovania podnikových procesov pri zefektívnení informačneho systému organizácie.* Diplomová práce, Bankovní institut vysoká škola Praha, 2015.

[6] Fasbinder, M.: Business process standards, Part 1: An introduction. Technická zpráva, Október 2007.

[7] Khan, R. M.: What standards really matter for BPM. Technická zpráva, CEO, Ultimus Inc., Máj 2005.

[8] Dumas, M.; van der Aalst, W.; ter Hofstede, A. T.: *Process-aware information systems: Bridging People and Software Through Process Technology.* A John Willey & Sons, inc., 2005, ISBN 978-0-471-66306-5.

[9] Lucidchart: UML Activity Diagram Symbols & Notation. [online]. 2016, [cit. 2017-5-8]. Dostupné z: https://www.lucidchart.com/pages/uml-activity-diagram-symbols-meaning

[10] van der Aalst, W.: Formalization and verification of event-driven process chains. *Information and Software Technology*, ročník 41, č. 10,

1999: s. 639 – 650, ISSN 0950-5849, doi:https://doi.org/10.1016/S0950-5849(99)00016-6. Dostupné z: http://www.sciencedirect.com/science/article/pii/S0950584999000166

[11] Business Process Model and Notation, v2.0. Technická zpráva, Object Management Group, Január 2011. Dostupné z: http://www.omg.org/spec/BPMN/2.0

[12] Chinosi, M.; Trombetta, A.: BPMN: An introduction to the standard. *Computer Standards & Interfaces*, ročník 34, č. 1, 2012: s. 124 – 134, ISSN 0920-5489, doi:https://doi.org/10.1016/j.csi.2011.06.002. Dostupné z: http://www.sciencedirect.com/science/article/pii/S0920548911000766

[13] Khlif, W.; Zaaboub, N.; Ben-Abdallah, H.: Coupling metrics for business process modeling. *WSEAS Transactions on Computers*, ročník 9, 2010, ISSN 11092750.

[14] Vanderfeesten, I.; Cardoso, J.; Mendling, J.; aj.: *Quality Metrics for Business Process Models.* Lighthouse Point, Florida, USA: Computers in Industry, 2007, s. 179–190.

[15] Hronza, R.; Pavlíček, J.; Mach, R.; aj.: Míry kvality v procesním modelování. *Acta Informatica Pragensia*, ročník 4, č. 1, 2015, ISSN 1805-4951.

[16] Mach, R.: *Návrh a tvorba nástroje pro optimalizaci procesu na základe analýzy BPM modelu.* Diplomová práce, České vysoké učení v Praze, Fakulta informačních technologií, 2015.

[17] Vanderfeesten, I.; Reijers, H. A.; van der Aalst, W. M.: Evaluating workflow process designs using cohesion and coupling metrics. *Computers in Industry*, ročník 59, č. 5, 2008: s. 420 – 437, ISSN 0166-3615, doi:https://doi.org/10.1016/j.compind.2007.12.007. Dostupné z: http://www.sciencedirect.com/science/article/pii/S0166361507001807

[18] Makni, L.; Khlif, W.; Haddar, N.; aj.: A Tool for Evaluating the Quality of Business Process Models Overview on current metrics for BPM.

[19] Hlaváč, V.; Šonka, M.: *Počítačové vidění.* GRADA a.s., 1992, ISBN 80-85424-67-3.

[20] Šikudová, E.; Černeková, Z.; Benešová, V.; aj.: *Počítačové videnie. Detekcia a rozpoznávanie objektov.* Wikina, první vydání, 2011, ISBN 978-80-87925-07-2.

[21] Dobeš, M.: *Zpracovaní obrazu a algoritmy v C#.* Nakladatelství BEN, 2008, ISBN 978-80-7300-233-6.

[22] Šefčík, M.: *Vyhladzovanie a gradientné operácie na obrazoch.* Diplomová práce, Slovenská technická univerzita v Bratislave. Materiálovotechnologická fakulta so sídlom v Trnave; Ústav aplikovanej informatiky, automatizácie a matematiky., 2011.

[23] what-when-how: Neighborhood Processing Part 3. Dostupné z: http://what-when-how.com/introduction-to-video-and-image-processing/neighborhood-processing-introduction-to-video-and-image-processing-part-3/

[24] ImageMagick: Convert, Edit or Compose Bitmap Images. Dostupné z: https://www.imagemagick.org/script/index.php

[25] ImageJ. Dostupné z: https://imagej.net/Welcome

[26] Manual - BoofCV. Dostupné z: https://boofcv.org/index.php?title=Manual

[27] Dalal, J.; Patel, S.: *Instant OpenCV Starter.* Packt Publishing, 2013. Dostupné z: https://ebookcentral.proquest.com/lib/techlib-ebooks/detail.action?docID=1215012

[28] OpenCV Contributors. Dostupné z: http://code.opencv.org/projects/opencv/wiki/Contributors

[29] Suzuki, S.; be, K.: Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, ročník 30, č. 1, 1985: s. 32 – 46, ISSN 0734-189X, doi: http://dx.doi.org/10.1016/0734-189X(85)90016-7. Dostupné z: http://www.sciencedirect.com/science/article/pii/0734189X85900167

Dodatok **A**

Zoznam použitých skratiek

API Application programming interface

ARIS Architecture of Integrated Information Systems

BMP Bitmap image

BPM Business process management

BPMN Business Process Modelling Notation

CFC Control-Flow Complexity

DIB Device Independent Bitmap

DPX Digital Picture Exchange

EPC Event-driven process chain

EXR Extra Space Storage

FITS Flexible Image Transport System

GIF Graphics Interchange Format

JPEG Joint Photographic Experts Group

JPG Joint Photographic Group

LOC Lines of code

OMG Object Management Group

PBM Portable Bit Map

PDF Portable Document Format

PGM Probabilistic Graphical model

PNG Portable Network Graphic

PPM Portable Pixel Map

RGB Red-green-blue color model

SAP Systems, Applications and Products

SR Sun Raster

SVG Scalable Vector Graphics

SWAP Simplified Workflow Access Protocol

TIFF Tagged Image File Format

UML Unified Modelling Language

WfMC Workflow Management Coalition

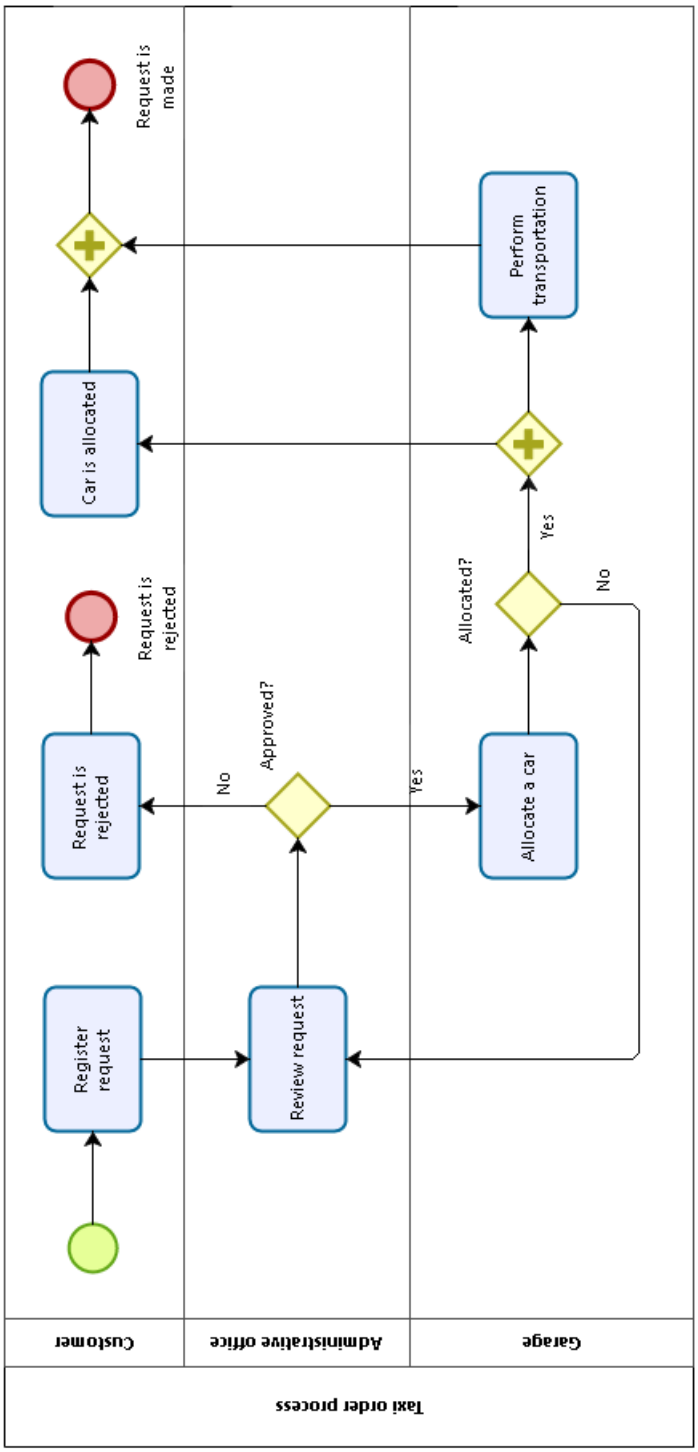
XMI XML Metadata Interchange

XML Extensible markup language

XPDL XML Process Definition Language

Dodatok B

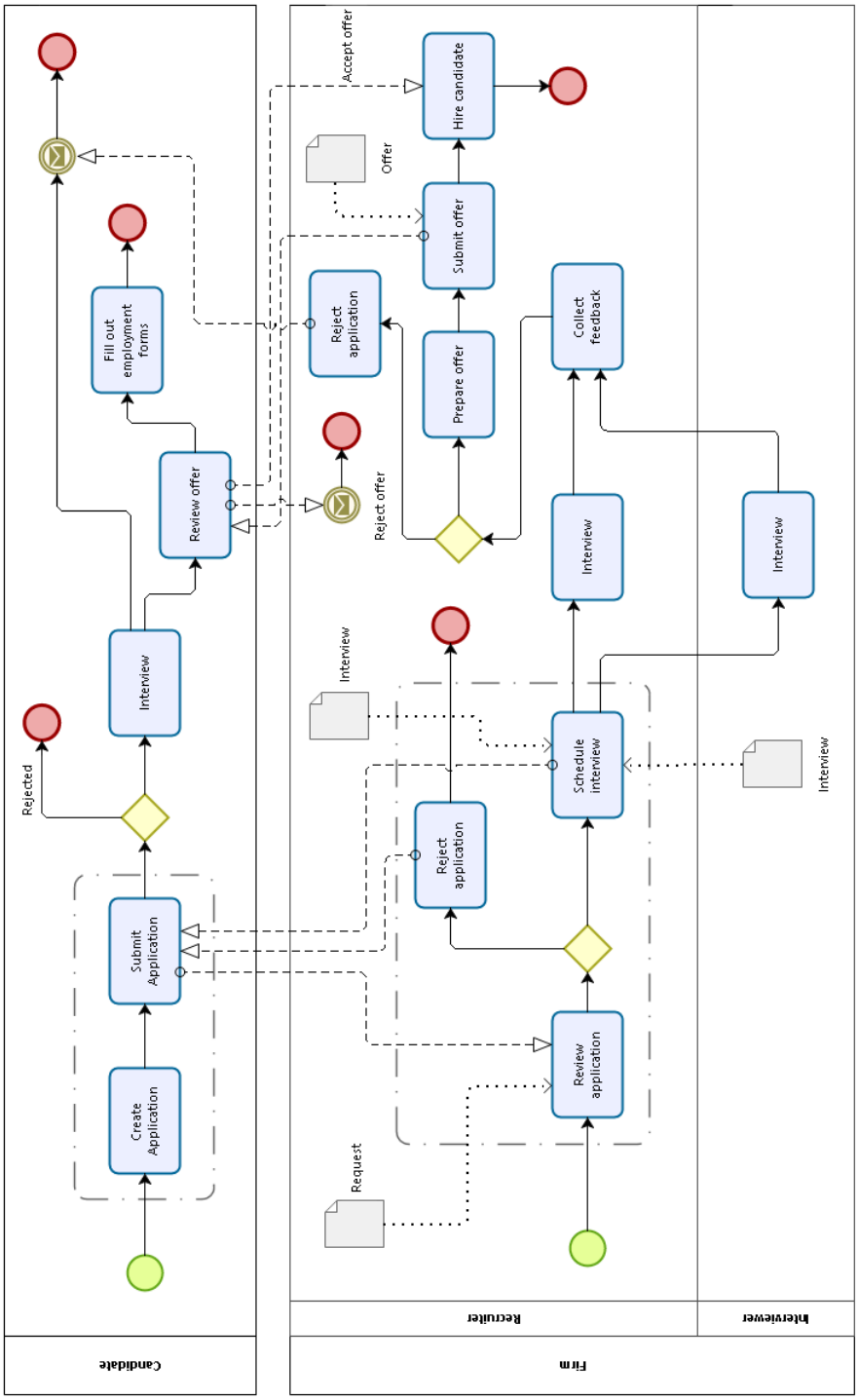
Proces „objednanie taxislužby“



Obr. B.1: Objednanie taxislužby.

Dodatok C

Proces „žiadosť o zamestnanie“



Obr. C.1: Žiadosť o zamestnanie.

Dodatok D

Obsah priloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `jar` ..... adresár so spustiteľnou formou implementácie
- `src`
  - `impl` ..... zdrojové kódy implementácie
- `thesis-src` ..... zdrojová forma práce vo formáte LaTeX
  - `img` ..... obrázky používané v práci
- `thesis-pdf` ..... text práce
  - `BP_Oliver_Findra_2017.pdf` ..... text práce vo formáte PDF