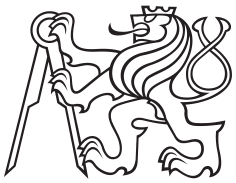
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Adaptace UI/SW založena na stavu uživatele

Bakalářská práce

Vojtěch Svoboda

Vedoucí: Ing. Jiří Šebek
Obor: Softwarové inženýrství
Studijní program: Softwarové technologie a management
Květen 2018

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Svoboda** Jméno: **Vojtěch** Osobní číslo: **434745**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové technologie a management**

Studijní obor: **Softwarové inženýrství**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Adaptace UI/SW založena na stavu uživatele

Název bakalářské práce anglicky:

Adaptive UI/SW based on user?s state

Pokyny pro vypracování:

[1, 2, 3] Návrh adaptivní struktury aplikace, která se mění se na základě kontextových informací, může být řešením obtíží s návrhem efektivního uživatelského rozhraní. Jako kontext bereme stav uživatele, kterému podle potřeby přizpůsobujeme rozhraní. Uživatel se může nacházet v různých psychických stavech, které mohou (ale nemusí) být zapříčiněné emocemi. Cílem je zjistit možnou spojitost mezi emocemi a stavem, dále získat kontextovou informaci o stavu uživatele a využít ji v adaptivní struktuře aplikace.
Cíle práce jsou následující:
1) Zjistěte vztah mezi stavem a emocemi uživatele
2) Prostudujte možnosti adaptivních uživatelských rozhraní a nástrojů pro zachycení emocí / zjištění stavu [1]
3) Rozšiřte adaptivní framework [3]
4) Využijte framework a vytvořte pomocí něj testovací aplikaci, tu otestujte na uživatelích
5) Upravte funkčnost frameworku, aby na základě stavu uživatele měnil uživatelské rozhraní
6) Vyhodnoťte testy

Seznam doporučené literatury:

1. LUNOVA, Anastasiia. Adaptace UI založena na emocích uživatele [online]. Prague, Czech Republic, 2017. Dostupné také z: http://hdl.handle.net/10467/68521. Bachelor thesis. ČVUT.
2. B. Schilit, N. Adams and R. Want, 'Context-Aware Computing Applications,' 1994 First Workshop on Mobile Computing Systems and Applications, Santa Cruz, California, USA, 1994, pp. 85-90. doi: 10.1109/WMCSA.1994.16 keywords: {Application software;Computer networks;Context;Context-aware services;Costs;Embedded computing;Home computing;Mobile computing;Pervasive computing;Portable computers}, URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4624429&isnumber=4624409
3. Šebek, J. - Richta, K.: Usage of Aspect-Oriented programming in Adaptive Application Structure. New Trends in Databases and Information Systems: ADBIS 2016 Short Papers and Workshops, BigDap, DCSA, DC, Prague, Czech Republic, August 28-31, 2016, Proceedings

© ČVUT v Praze, Design: ČVUT v Praze, VIC

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Jiří Šebek, laboratoř inteligentního testování softwaru FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **21.05.2018** Termín odevzdání bakalářské práce: **26.05.2017**

Platnost zadání bakalářské práce: **30.09.2019**

[redacted] [redacted] [redacted]

Ing. Jiří Šebek
podpis vedoucí(ho) práce

podpis vedoucí(ho) ústavu/katedry

prof. Ing. Pavel Ripka, CSc.
podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

23/5/18
Datum převzetí zadání

[redacted]
Podpis studenta

© ČVUT v Praze, Design: ČVUT v Praze, VIC

Poděkování

Chtěl bych zejména poděkovat panu vedoucímu, Ing. Jiřímu Šebkovi, za příjemnou spolupráci, cenné rady, rychlou komunikaci a zejména za neskutečnou trpělivost, kterou se mnou v průběhu vypracovávání této práce měl.

Dále bych chtěl poděkovat rodině a přátelům za pochopení a podporu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

V Praze, 25. května 2018

Abstrakt

Tato bakalářská práce se zabývá problematikou emocí a stavu uživatele a přizpůsobení uživatelského rozhraní tomuto uživateli. V rámci této práce je provedena úprava a rozšíření existující knihovny na adaptaci uživatelského rozhraní v závislosti na emocích uživatele, který ve své bakalářské práci vytvořila A. Lunova. Výsledná knihovna umožňuje detekci stavu uživatele a následnou adaptaci uživatelského rozhraní na základě tohoto stavu. Řešení má za cíl usnadnit a zpříjemnit uživateli užívání aplikace a tím ho také povzbudit k jejímu dalšímu používání. Framework je psaný programovacím jazykem Java a vyvinutý byl v prostředí Android Studio 3.1.2 od JetBrains. Tato práce využívá SDK Affdex od Affectiva.

Klíčová slova: Android, rozhraní, emoce, stav, stav vědomí, návrh, framework, adaptace

Vedoucí: Ing. Jiří Šebek

Abstract

This bachelor thesis deals with problematics of emotions, state of user and ability of user interface to adapt to this user. Within this work is done adjustment and extension of existing library allowing adaptation of user interface depending on the user's emotions, which was created in bachelor thesis written by A. Lunova. Resulting library allows detection of user's state and following adaptation of user interface based on this state. Objective of this solution is to make using of application easier and more comfortable for user, convincing him to keep using this application. Framework is written using programming language Java and developed in Android Studio environment by JetBrains. This thesis is using SDK Affdex by Affectiva.

Keywords: Android, interface, emotions, state, state of consciousness, design, framework, adaptation

Title translation: Adaptive UI/SW based on user's state — Bachelor thesis

Obsah

Obrázky

Tabulky

Kapitola 1

Úvod

V dnešní době, na rozdíl od minulého století, už si téměř nedovedeme představit, že bychom opustili domov bez mobilního telefonu - být dostupný téměř nonstop se stalo novým trendem. Téměř polovina lidí v moderním světě (podle portálu statista.com přesáhne počet uživatelů chytrých telefonů v roce 2018 hranici 2,5 miliardy[9]), snad jen s výjimkou nejstarších a nejmladších generací, vlastní některý z typů chytrých telefonů, přičemž mobilní telefony s operačním systémem Android či iOS z toho tvoří více než 99,5% z čehož si zařízení s některou variací systému Android nárokují více než 85% [8]. Technika postupuje dopředu mílovými kroky každý den a u mobilních zařízení tomu není jinak. Výrobci se snaží poskytovat co možná největší výkon v co nejmenším obalu, nejvýkonnější počítače dostupné na trhu před 20 lety nesahají po hardwarové stránce dnešním telefonům téměř ani po kotníky. Software telefonů se také více a více snaží přizpůsobit jeho uživatelům, ať už je to nabídkou dostupných aplikací, ovládání pomocí hlasu nebo možností všemožných drobných změn. Co však je stále opomíjeno a přehlíženo je dynamické přizpůsobování těchto zařízení tak, aby pro jejich uživatele byly stále pohodlné za všech situací, bez nutnosti vnějších zásahů. Do hardwarových dynamických přizpůsobování nás čekají ještě roky vývoje, ale se současnou technologií už můžeme uživateli adaptovat uživatelské prostředí a aplikace tak, aby uživateli za všech (nebo alespoň ve většině situací) vyhovovaly a splňovaly jeho očekávání.

Emoce a stav uživatele jsou navzájem velmi úzce provázané. Většina stavů jako takových je často způsobena hlavně působením emocí na daného jedince. Samozřejmě emoce nejsou jediným zapříčiněním změny stavu člověka, vliv na to mohou mít například i chemické látky, fyzická námaha či působení okolí.

Cílem této práce je prostudovat závislost emocí na aktuálním stavu uživatele Android zařízení a vytvořit framework, který budou moci vývojáři využít k vytváření aplikací, které se budou dynamicky přizpůsobovat uživatelovu stavu a budou mu lépe vyhovovat. Framework bude snadno rozšiřitelný a upravovatelný, aby si ho vývojář mohl přizpůsobit dle vlastních potřeb.

Kapitola 2

Rešerše

V této části práce se budeme zabývat teorií a měřením emocí, následně teorií stavu člověka. Dále se budeme zabývat uživatelským rozhraním, jeho návrhem a následnou adaptací.

2.1 Emoce

V průběhu staletí se lidstvo snažilo na tuto otázku odpovědět. Koncept základních, nebo primárních, emocí se datuje přinejmenším do prvního století, kdy čínská encyklopedie[18] – Book of Rites[21] – identifikuje sedm základních „pocitů“ (radost, naštvání, smutek, strach, láska, odpor a záliba). Samotné slovo „emoce“ se datuje do roku 1579, kdy bylo převzato z francouzského émouvoir, které v překladu znamená vyvolat, rozvířit nebo vybudit.

Detailněji na tuto otázku psychologové odpověděli v minulém století, a to mnoha způsoby, ale přesto jasná a výstižná odpověď stále chybí. Pro ilustraci, níže je několik definicí emocí, které nám pomohou pochopit, s čím máme tu čest.

R. Sternberg: „Emoce je pocit (feeling) zahrnující fyziologické a behaviorální reakce na interní a externí události.“ [35]

J.S. Nairne: „Emoce je komplex psychologických událostí, které zahrnují směs reakcí: fyziologické reakce (vzrušení), expresivní reakce (postavení těla, výraz tváře ale i vydávané zvuky) a nějaký druh subjektivní zkušenosti (myšlenky, pocity)“ [29]

V knize The Nature of Emotions: Fundamental Questions[24] Paula Ekmana a Richarda J. Davidsona diskutuje mnoho výzkumníků v poli emocí nad jejich definicí a významen.

Jaak Panksepp definuje emoce jako intenzivní vzrušení mozkových systémů, které silně podporují organismus v impulzivním chování.

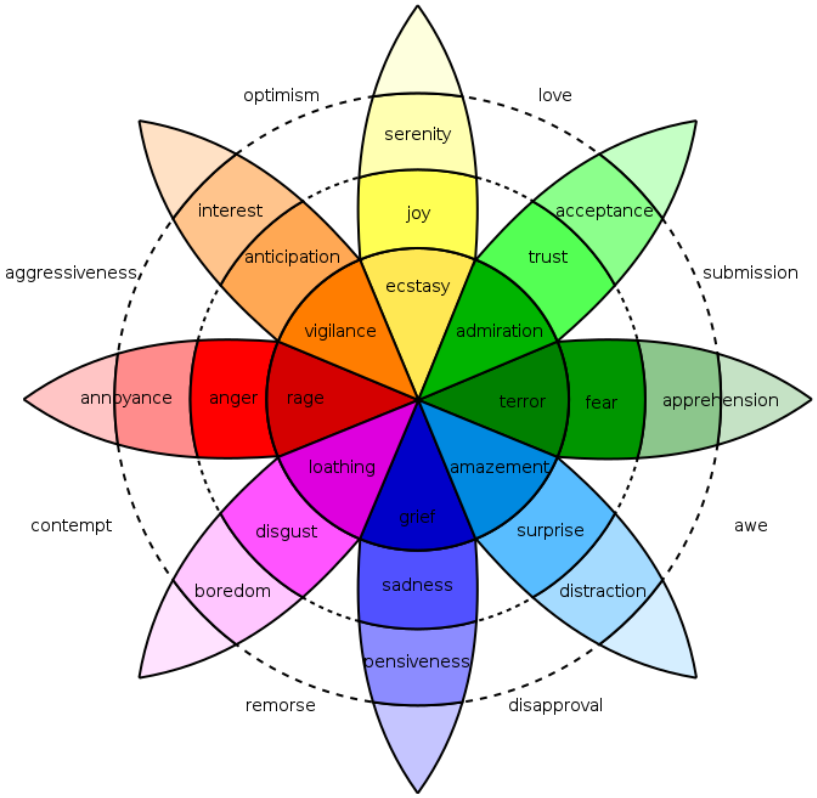
Joseph LeDoux definuje emoce: „Emoce jsou afektivně nabité a subjektivně prožívané stavy vědomí."

Koncept emocí podle Jamese R. Averilla se skládá ze 3 částí – emočních syndromů, emočních stavů a emočních reakcí. Emoční syndrom je to, co myslíme, když mluvíme o vzteku, smutku, strachu a podobně; například syndrom strachu popisuje a předepisuje, co člověk může (nebo by měl) dělat, když je naštvaný. Emoční stav je krátkodobá nálada v reprezentativní reakci na korespondující emoční syndrom. Emoční reakce je sada odpovědí, které jsou projeveny jedincem v emočním stavu (například mimika obličeje, fyziologické změny, chování, subjektivní zkušenosti).

Pro účely této práce však není samotná definice emocí podstatná. Důležité je si pouze uvědomit, že emoce mají velkou roli v našich reakcích na nastalé situace a ponděty.

Ve dvacátém století identifikoval Paul Ekman, přední americký psycholog, šest základních a měřitelných emocí (vztek, nechuť, strach, štěstí, smutek a překvapení). Robert Plutchik těchto základních emocí definoval 8, čtyři páry protikladů (radost-smutek, vztek-strach, důvěra-nedůvěra a překvapení-očekávání).

Tyto základní emoce podle Plutchika mohou být modifikovány a vytvářet tak komplexní, složité emoce. K zobrazení těchto komplexních emocí vytvořil Plutchik „wheel of emotions" (obrázek 2.1), graf, který ukazuje vývoj komplexních emocí ze základních, ať už nárůstem intenzity, nebo kombinací emocí.



Obrázek 2.1: Plutchikovo kolo emocí[31]

Další klasifikaci emocí má na svědomí Human-Machine Interaction Network on Emotion (HUMAINE), která navrhuje pro emoční anotaci a reprezentaci do jazyko (Emotion Annotation and Representation Language, EARL), a klasifikuje 48 emocí, které dělí do 10 skupin.

• Negativní mocné – hněv, rozmrzelost, pohrdání, znechucení, iritace • Negativní neovladatelné – úzkost, rozpaky, strach, bezmoc, bezradnost, starost • Negativní myšlenky – pýcha, nejistota, závist, frustrace, vina, hanba • Negativní pasivní – nuda, zoufalství, zklamání, ublížení, smutek • Neklid – stres, šok, napětí • Pozitivní živé – pobavení, potěšení, radost, nadšení, povznesení, štěstí, slast • Starostné – náklonnost, empatie, přátelskost, láska • Pozitivní myšlenky – odvaha, naděje, pokora, satisfakce, důvěra • Mírně pozitivní – klid, spokojenost, relaxace, úleva, vyrovnanost • Reakční – překvapení, zájem, zdvořilost

2.1.1 Barvy a emoce

Propojením barev a emocí se zabývá psychologie emocí. Základními barvami z psychologického hlediska jsou červená, modrá, žlutá a zelená.

Je obecně známo, že barvy silně ovlivňují naše emoce a pocity. Například červená barva je spojována se vzrušením, žlutá je spojována s radostnými pocity, zelená má relaxační efekt a modrá barva je spojována s pocitem komfortu a bezpečí.

Bylo provedeno velké množství výzkumů na téma barev a emocí, kde účastníci měli přiřazovat emocím barvy, které jim podle jejich mínění odpovídaly.

Jeden z takových výzkumů je byl proveden na univerzitě v Georgii a ve své práci ho popisuje Naz Kaya, Ph.D. Tohoto výzkumu se zúčastnilo téměř sto studentů univerzity, kteří byli požádáni, aby popsali svou emoční odezvu na některé barvy.[26]

Další výzkum, který byl na toto téma prováděn, se zaměřil na přiřazování barev k jednotlivým výrazům některých základních emocí. [16] Příklad výsledku je možné vidět na obrázku 2.2

Obrázek 2.2: Výsledek přiřazení barev pro emoci štěstí[16]

Variace ve výsledcích těchto výzkumů je dána hlavně tím, že na každého člověka působí barvy i emoce jinak. Existují barvy teplé a barvy chladné[2]. Teplé barvy jsou okolo vlnového spektra červené (červená, žlutá, oranžová), tyto barvy mohou evokovat emoce v rozsahu od pocitu bezpečí a tepla až po pocit vzteku či nepřátelství. Studené, nebo chladné barvy se vyskytují ve spektru okolo modré barvy. Tyto barvy mohou navozovat pocit klidu, ale také smutku[19].

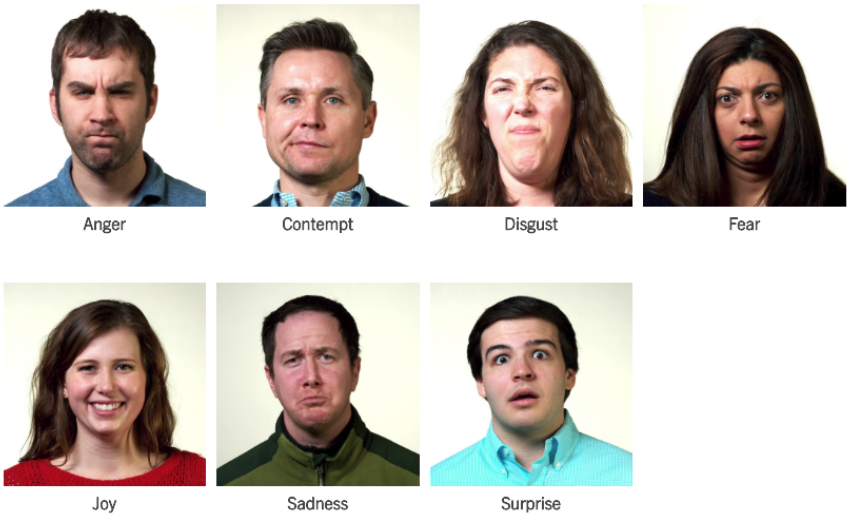
Příklady některých barev a jejich účinků[5]: •červená – pozitivní účinky: síla, teplo, energie, nadšení – negativní účinky: agrese, vzdor •modrá – pozitivní účinky: důvěra, klid, vyrovnanost – negativní účinky: chladnost, nepřátelskost •žlutá – pozitivní účinky: optimismus, sebedůvěra, přátelskost – negativní účinky: strach, deprese, úzkost •zelená – pozitivní účinky: balanc, klid, útěcha – negativní účinky: nuda, stagnace •oranžová – pozitivní účinky: teplo,

komfort, zábava, bezpečí – negativní účinky: frustrace, lehkomyslnost •šedá – pozitivní účinky: psychologická neutralita – negativní účinky: deprese, ztráta sebedůvěry •růžová – pozitivní účinky: láska, teplo, vzrušení – negativní účinky: slabost, utlumení

Podle jednoho výzkumu dokonce instalace modrého osvětlení, místo tradičního bílého/nažloutlého, snížila v dané oblasti kriminalitu na ulicích.[33]

2.1.2 Měření emocí

K detekování a měření emocí budeme používat framework Affdex SDK (Affectiva)[13], který počítá emoce z mimiky tváře uživatele. Affectiva je kompatibilní s mobilními telefony se systémem Android, na které je tato práce zaměřena. Umí detekovat 7 základních emocí (zobrazené na obrázku 2.3), tedy 6 základních emocí definovaných Paulem Ekmanem a k tomu opovržení, a dalších 20 mimických výrazů obličeje (zobrazené v příloze), dále umí detekovat věk, pohlaví, brýle a národnostní příslušnost, ale tyto informace nebudeme využívat.



Obrázek 2.3: Emoce detekované Affdex SDK[12]

Metriky[12], které využívá toto SDK, jsou vyjádřeny pomocí skóre v intervalu od 0 do 100 v závislosti na projevu daného výrazu nebo emoce, kde 0 odpovídá žádnému výrazu a 100 plně vyjádřenému výrazu. Dále měří další 2 metriky, a to zapojení (zapojení obličejových svalů) v intervalu od 0 do 100 a valenci (pozitivní nebo negativní povaha zážitku) v intervalu od -100 do 100.

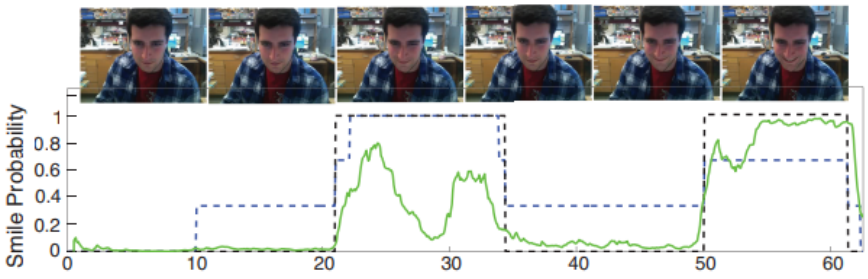
Affectiva pracuje s daty z kamery (v našem případě přední kamery) telefonu, kde detekuje na obličeji 34 základních bodů (popsané na obrázku 2.4), ze kterých určuje výrazy v obličeji a jimi vyjádřené emoce. Mapování emocí je

založeno na EMFACS, které vyvinuli W. V. Friesen a Paul Ekman.

Index	Point on Face	Index	Point on Face
0	Right Top Jaw	17	Inner Right Eye
1	Right Jaw Angle	18	Inner Left Eye
2	Tip of Chin	19	Outer Left Eye
3	Left Jaw Angle	20	Right Lip Corner
4	Left Top Jaw	21	Right Apex Upper Lip
5	Outer Right Brow Corner	22	Upper Lip Center
6	Right Brow Center	23	Left Apex Upper Lip
7	Inner Right Brow Corner	24	Left Lip Corner
8	Inner Left Brow Corner	25	Left Edge Lower Lip
9	Left Brow Center	26	Lower Lip Center
10	Outer Left Brow Corner	27	Right Edge Lower Lip
11	Nose Root	28	Bottom Upper Lip
12	Nose Tip	29	Top Lower Lip
13	Nose Lower Right Boundary	30	Upper Corner Right Eye
14	Nose Bottom Boundary	31	Lower Corner Right Eye
15	Nose Lower Left Boundary	32	Upper Corner Left Eye
16	Outer Right Eye	33	Lower Corner Left Eye

Obrázek 2.4: 34 bodů na obličeji používaných v Affdex SDK k detekci mimiky a emocí[11]

Affdex SDK detekuje pohyby svalů tváře a jejich kombinace, které se označují jako AU (action unit). Na základě přítomnosti těchto AU se určuje pravděpodobnost jednotlivých emocí a výrazů, jak lze vidět na obrázku 2.5 a 2.7. Detekce mimických výrazů a emocí se děje pomocí strojového učení (Affectiva využívá technologie Deep learning). Prvním krokem je extrahovat důležité části obličeje (rty, obočí, oči atd.), dále se analyzují pohyby, tvar a jejich kombinace k určení AU.

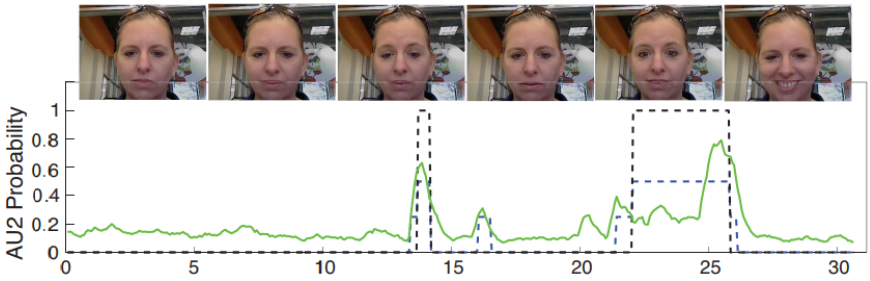


Obrázek 2.5: Určení pravděpodobnosti výrazu Affdex SDK (detekce úsměvu)[14]

AU jsou rozděleny podle oblastí (ROI – region of interest), na které se zaměřují (viz obrázek 2.6). Jednotlivé snímky jsou pak oříznuty podle ROI a pro každou oblast je vypočítán histogram orientovaných gradientů. Výsledným výstupem je váha jednotlivých emocí a mimických výrazů detekovaných Affdex SDK.

AU	Description	Example
AU02	Outer corner of eyebrow raised.	
AU04	Eyebrows drawn medially and down.	
AU09	Upper lip raised and inverted; superior part of the nasolabial furrow deepened; nostril dilated by the medial slip of the muscle.	
AU10	Upper lip puller up (either unilaterally or bilaterally).	
AU12/ smile	Lateral lip corner pull without AU04 or AU09.	
AU14	Dimpler (dimples in the cheeks).	

Obrázek 2.6: Definice a příklady AU[14]



Obrázek 2.7: Určení pravděpodobnosti detekce AU2[14]

2.2 Stav

Pod pojmem stav člověka (uživatele) si lze představit nespočet různých věci. Může nás například napadnout fyzický stav, v jakém se daný člověk nachází, nebo psychické rozpoložení daného člověka, dalším příkladem může být stav denního režimu, v kterém se člověk právě nachází (bdělost, spánek)[22]. Protože se nechceme věnovat pouze jednomu z těchto pojmů, ale raději je obsáhnout všechny, budeme používat jako stav brát stav vědomí uživatele[38], který nejlépe zahrnuje všechny tyto kategorie.

Pro definování stavu uživatele tak, jak ho chceme používat v této práci si nejdříve musíme říci, co je to vědomí.

Vědomí se rozdělit na 2 pojmy. První je primární fenomenální vědomí, které odkazuje k subjektivním zkušenostem. Skládá se ze vzorů subjektivních zkušeností nebo qualií – pocitů, vjemů, emocí, obrazů těla, mentálních obrazů a vnitřních myšlenek.[32] Druhým je reflektivní vědomí. Reflektivní vědomí je závislé na primárním vědomí, protože dále zpracovává informace, které jsou získány v primárním vědomí. Odkazuje na své vlastní akce, reflektuje a dále zpracovává vybraný obsah vědomí. Jinými slovy bere vybraný obsah primárního vědomí pro další zamyšlení. [32] Sebereflexe nebo vědomí vlastního já lze dosáhnout právě skrze reflektivní vědomí.[6]

2.2.1 Stav vědomí (state of consciousness)

Stav vědomí má, jako většina věcí v psychologii, více definic. Přesto tyto definice nejlépe vystihují stav člověka jako takový.

Stanley Krippner definoval stav vědomí jako: „mentální stav, který může být subjektivně rozpoznaný jedincem (nebo objektivním pozorovatelem jedince) jako reprezentující změnu v psychologickém fungování od jedincovo normálního, pohotového a bdělého stavu“.[17]

Patricia Churchland ve své publikaci uvedla: „První v množině prototypických vědomých stavů je řada senzorových vjemů, například vidět ptáka létat, cítit bolest od popáleniny, slyšet policejní sirény.. Za druhé můžeme přidat seznam stavů, které obvykle nejsou mezi senzorovými zážitky. Tento seznam zahrnuje stavy jako pamatování si, co jsme měli k snídani, znalost jízdy na kole, představování si šestinohého psa... Podobně emoční stavy jako cítit strach, zlost, smutek a vzrušení, stejně jako hlad, žízeň, sexuální touhu a mateřskou lásku na tento list patří.“

Problém zde však spočívá v tom, že různé stavy jsou rozeznávány pouze podle různých obsahů vědomí.[32]

Definice stavu vědomí podle pánů Antti Revonsuo, Sakari Kallio & Pilleriin Sikka: „Stav vědomí je definován ne podle náplně nebo vzorů zážitků, ale podle souvislostí, které obklopují kontext, ve kterém se tyto zážitky odehrávají.“ Tato definice je podle autorů nejlepší možnou pro pozdější definici. Mají k tomu 2 důvody. Za prvé, normálnost nebo pozměnění stavu vědomí musí obsahovat jiná kritéria než jen obsah primárního vědomí, a reflektivní úsudek nebo posouzení subjektu založené na obsahu primárního vědomí. Musí zahrnovat referenci k prostředí (nebo přesněji ke vztahům mezi prostředím a vědomím). Termín pozměnění by měl být interpretován jako reprezentativní nebo relační, ne pouze jako fenomenální. Za druhé, stav vědomí zahrnuje něco více než jen vzory subjektivních zkušeností, zahrnuje také nevědomý přirozený

mechanický kontext, ve kterém se vyvolává fenomenální obsah vědomí, nebo nevědomé mechanismy reprezentující v mozku nepřímé vztahy mezi vědomím a prostředím.[32] Normální stav vědomí je takový stav, kde mechanismy v mozku předávají přesné informace o okolním světě a sobě do vědomí.

2.2.2 Pozměněný stav vědomí (Altered state of consciousness, ASC)

Zde už se dostáváme ke stavům, které budeme využívat v této práci. Pozměněný stav vědomí má opět více definic.

Tart (1990) definoval ASC takto: „ASC pro daného jedince je takový, kdy jedinec jasně cítí kvalitativní posun ve struktuře jeho mentálního fungování, to znamená, že cítí nejen kvantitativní posun (více či méně bdělý, lepší či horší představivost, nudnější či ostřejší atd.), ale také to, že některé kvality jeho mentálního procesu jsou jiné.“[32] Farthing (1992) definoval ASC jinak: „ASC může být definován jako dočasná změna v celkové struktuře subjektivních zážitků taková, že dotyčný věří, že jeho mentální fungování je velmi rozdílné od obecných norem jeho normálního bdělého stavu.

Z těchto definic jasně vyplývá, že pokud se člověk nachází v ASC, vnímání okolního světa, sebe sama ale i chování je rozdílné od normálu. To také upravuje nároky uživatele na své okolí, například i mobilní aplikace.

Co způsobuje změnu:

K pozměnění stavu vědomí může dojít z mnoha různých příčin. Nejčastější příčinou ASC jsou emoce, na které je tato práce hodně zaměřena. Dalšími příčinami můžou být chemikálie (alkohol, drogy, medikace), mentální poruchy (schizofrenie, bipolární porucha...), hypnóza, meditace, prostředí, fyzická nemoc či pouze nějaká aktivita.

Hypnóza: hypnózu definoval Kirsch (2005)[20] jako: „stavoví i nestavoví teoretici se shodují, že hypnotické návrhy mohou způsobovat ASC (například subjektivní zkušenost s amnézií, analgesií, nedobrovolnost...).

Chemikálie: drogy[3] – droga je jakákoliv substance (s výjimkou jídla a vody), která, pokud je dána do těla, mění funkci těla buď fyzicky a/nebo psychologicky. Drogy můžou být legální (alkohol, kofein, tabák) nebo nelegální (cannabis, extáze, kokain, heroin).

Psychoaktivní drogy ovlivňují centrální nervovou soustavu a upravují náladu, myšlení a chování uživatele. Efekt drog se mění od člověka k člověku, podle jedincových charakteristik, charakteristik drogy samotné a způsobu a prostředí ve kterém je droga podána. Mohou být rozděleny do 4 kategorií:

halucinogeny (mění vnímání a mohou způsobovat halucinace – LSD, houbičky), depresory (snižují ostražitost zpomalením aktivity centrální nervové soustavy – heroin, alkohol, analgetika), simulanty (zvyšují stav vzrušení v těle zvýšením aktivity mozku – kofein, nikotin, amfetamin) a ostatní (mají vlastnosti více než 1 z předchozích kategorií – cannabis má všechny 3).

Psychická porucha: psychické poruchy[4] (také známé jako mentální poruchy) odkazují k nenormálním změnám mysli, které vedou k chování, které může silně narušit každodenní fungování.

Symptomy a příznaky těchto poruch se liší podle typu poruchy. Typicky, symptomy poruch chování nebo nálady jsou častější, chronické a oslabující. Můžou vážně narušit schopnost fungovat ve společnosti. Psychologické poruchy mohou občas působit jako jiná nemoc nebo nemít žádný identifikovatelný důvod. Symptomy mohou zahrnovat: užívaní drog, násilí, nepřátelskost, časté změny nálady, deprese, izolace, vztek, halucinace a přeludy. Psychické poruchy se klasifikují do následujících kategorií: přizpůsobovací poruchy, úzkostlivé poruchy, dětské poruchy, disociativní poruchy, poruchy jedení, předstírací poruchy, poruchy kontroly impulzivního chování, poruchy nálady, osobnostní poruchy, psychotické poruchy, sexuální poruchy a poruchy indentity pohlaví, spánkové poruchy a somatoformní poruchy.

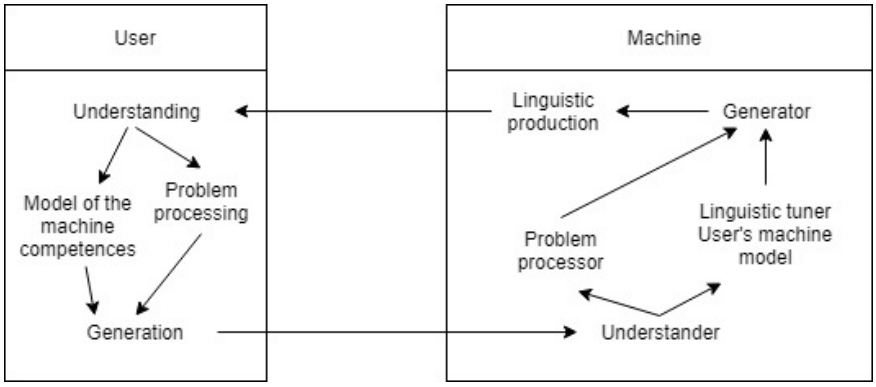
Stavy:

• Opilost • Únava • Úzkost (panika) • Deprese • Štěstí • Extáze (nadšení, vytržení) • Mánie • Roztěkanost • Naštvání • Zuřivost • Frustrace • Šok • Nemoc • Neutrální (normální)

Takovýchto stavů může být vysoký počet. Pro účely této práce budeme dále rozvádět jen několik z nich.

2.3 Uživatelské rozhraní

Uživatelské rozhraní je vše, co je navrženo pro interakci člověka se systémem[7]. Může se zde jednat o hardwarové komponenty ale také o způsob, jakým uživatel komunikuje s aplikací, viz obrázek 2.8.



Obrázek 2.8: Model interakce mezi člověkem a počítačem [15]

Rychlý vývoj z velkých počítačů na malá početná zařízení dramaticky snížil cenu a zapříčinil dramatický technologický pokrok. Cena výpočetního výkonu se milionkrát snížila za poslední desetiletí, zatímco spolehlivost a rychlost se zvýšila. Protože nyní prodejci nabízejí podobnou systémovou funkčnost, přesunul se hlavní cíl pokroku k interakci člověka se systémem.

Existují 3 hlavní důvody:

1) Nárůst populace uživatelů – nových, technicky netrénovaných lidí. Cílem designu uživatelského rozhraní je navrhnout rozhraní, které bude pro co největší procento lidí intuitivní a přijatelné. Návrhy musí být validovány přes pilotní studie a testy přijatelnosti (acceptance test).

2) Narůstající organizační závislost na interaktivních systémech – zatímco automatizace výplatních pásek či účetních funkcí byla původně podporována jako šetřivá výhoda, moderní interaktivní systémy jsou středem funkčnosti mnoha organizací. Například rezervační systémy jsou často kritickým ukazatelem úspěchu či dokonce přežití organizace. V dnešní době už chyby, dlouhé čekání na odpověď, pomalý výkon či zmatený operátor nejsou přijatelné. Cílem je zlepšení uživatelského přístupu. Pro vytěžované aplikace i 10% redukce času na transakci může vést k 10% snížení nákladů na obsluhu, terminály či hardware.

3) Nutnost minimálního počtu chyb v životně důležitých aplikacích – aplikace jako ovládání nukleárního reaktoru, kontrola vzdušné dopravy, zdravotní intenzivní péče či vysílání policejních a hasičských jednotek vyžadují naprosto minimální počty chyb a pohotovostní podmínky, které eliminují luxus konzultace s manuály nebo experty. Návrháři se snaží o prezentaci komplexních informací velmi rychle a zřetelně. Výrobci těchto systémů vědí, že chyby v jejich návrhu mohou být jedním z faktorů nějaké katastrofy.

Hlavní problémy při návrhu rozhraní: • Příkazový jazyk vs menu • Čas odpovědi a display rate (rychlost zobrazení výsledku) • Formulace systémových hlášek • Online tutoriály, vysvětlivky a zprávy • Hardware

Jak tyto problémy odstranit: • Participační návrh (participatory desing) • Specifikační metody • Softwarové implementační nástroje • Pilotní studie a testy přijatelnosti • Další vývoj návrhu založený na zpětné vazbě od uživatelů

Podle Shneidermana [34] lze nalézt spojitosti, mezi spokojeným uživatelem a systémem. Hlavními rysy takového systému by byly: • Snadná viditelnost objektu zájmu • Rychlé zpětné akce • Vyměnění komplexních příkazových jazyků za přímou manipulaci s objektem zájmu

Existují dva způsoby návrhu UI:

1. Statický – rozhraní si zachovává stálou strukturu chování ve vztahu k uživateli
2. Adaptivní – rozhraní se dynamicky přizpůsobuje uživatelskému modelu chování

2.4 Adaptace

Problematika adaptace uživatelského rozhraní je ve způsobu adaptace a v důvodech, proč adaptovat:

Adaptace může být vyvolána nedostatečnými (nebo naopak rozšířenými) právy uživatele v systému – například zde by docházelo ke skrývání či odkrývání určitých položek nebo funkcí, které z bezpečnostních důvodů musí zůstat některým uživatelům skryté. Může se také jednat o rozšířené funkce systému či aplikace, za které by si autor chtěl nechat zaplatit navíc.

Další možností adaptace je přizpůsobení uživatelského rozhraní na základě používání objektů aplikace – změna pořadí položek v navigaci či uspořádání tlačítek na panelu nástrojů.

Poslední adaptací, kterou zde budeme zmiňovat jsou kosmetické úpravy rozhraní – zde můžeme mluvit o úpravě barev, fontu nebo tvarů a velikostí.

Kapitola 3

Související práce

Tato práce volně rozvíjí a navazuje na bakalářskou práci A. Lunova – Adaptace UI založena na emocích uživatele[27]. Dále využívá poznatky z práce Affective computing[30], která se zabývá detekcí emocí. Stav únavy se vyskytuje v mnoha pracích, zejména se řeší detekce únavy řidiče v dopravním prostředku[36][10][25]. Nejvíce je třeba asi zmínit práci Fatigue detection based on the distance of eyelid [23]] od autorů Wenhui Dong a Xiaojuan Wu, která se zabývá detekcí únavy v závislosti na zavření očního víčka. Tato práce ve své implementační části také využívá myšlenky popsané v bakalářské práci N. Mishchenko – Aspektově orientovaný vývoj adaptivní struktury aplikace pro Java EE aplikace[28].

Kapitola 4

Návrh

V této práci se zaměříme na navržení detekce a adaptace pouze některých stavů z těch, které jsou uvedené v kapitole 2.2.2). Protože cílem práce je vytvořit dále snadno rozšiřitelný framework, bude nám těchto několik stavů stačit jako reprezentativní vzorek pro vývojáře aplikací, kteří v případě potřeby budou moci nějaký stav doimplementovat. Mnou navrhované základní stavy, které si budeme dále rozvádět jsou – únava, frustrace, naštvání, zuřivost, radost (štěstí), nadšení, deprese, úzkost a neutrální stav.

Spokojenosti uživatelů nelze dosáhnout 100%, protože každý člověk má jiné vnímání a jinak na něho působí emoce. Proto se při volbě adaptací musíme zaměřit na obecně známá fakta, která by měla platit pro většinu uživatelů. Na tuto první možnost se v této práci zaměříme.

Další možností je iterativní vývoj adaptací tak, abychom dosáhli maximálního procenta spokojených uživatelů.

Poslední možností je nechat každého konkrétního uživatele si vytvořit či přizpůsobit adaptace pro každý stav tak, aby mu při budoucím používání aplikace vyhovovaly.

Adaptovat budeme na základě stavu tyto části uživatelského rozhraní: • Pozadí – barva

• Font – barva, velikost, styl

• Položky (tlačítka, input text area) – barva, velikost, tvar, pořadí, uspořádání

Primárně se v návrhu zaměříme na barvu. Barevné schéma má za cíl pozitivně ovlivnit stav uživatele.

4.1 Stavy

Barevné palety pro jednotlivé stavy jsou navrženy na základě informací obsažených v kapitole 2.1.1 tak, aby pozitivně ovlivnily uživatele, například naštvaného uživatele chceme uklidnit, proto převládá v paletě modrá barva, smutného uživatele chceme povzbudit, proto využíváme barvy ze spektra červené.

Neutrální stav

Neutrální stav je základní stav. Použitý je v případě, že při detekci nezjistíme žádný z námi detekovatelných a využívaných stavů.

Položka adaptace	ilustrace	R	G	B	HEX
Pozadí		238	201	177	EEC9B1
Písmo		71	87	111	47576F
Akcent		162	127	104	A27F68

Tabulka 4.1: Paleta barev pro neutrální stav

Únava

Zvětšení položek. Zde je cílem adaptace uživatele dostat do bdělého stavu a zároveň mu zjednodušit práci s aplikací.

Položka adaptace	ilustrace	R	G	B	HEX
Pozadí		255	51	51	FF3333
Písmo		255	255	255	FFFFFF
Akcent		255	242	41	FFF229

Tabulka 4.2: Paleta barev pro stav únava

Frustrace

Položka adaptace	ilustrace	R	G	B	HEX
Pozadí		114	189	232	72BDE8
Písmo		71	83	181	4753B5
Akcent		116	117	130	747582

Tabulka 4.3: Paleta barev pro stav frustrace

Naštvání

Zaoblení položek.

Položka adaptace	ilustrace	R	G	B	HEX
Pozadí		114	189	232	72BDE8
Písmo		71	83	181	4753B5
Akcent		116	117	130	747582

Tabulka 4.4: Paleta barev pro stav naštvání

Zuřivost

Zvětšení, zaoblení položek.

Položka adaptace	ilustrace	R	G	B	HEX
Pozadí		114	189	232	72BDE8
Písmo		71	83	181	4753B5
Akcent		116	117	130	747582

Tabulka 4.5: Paleta barev pro stav zuřivost

Radost

Položka adaptace	ilustrace	R	G	B	HEX
Pozadí		0	204	0	00CC00
Písmo		255	28	28	FF1C1C
Akcent		252	252	28	FCFC1C

Tabulka 4.6: Paleta barev pro stav radost

Nadšení

Zvětšení položek.

Položka adaptace	ilustrace	R	G	B	HEX
Pozadí		0	204	0	00CC00
Písmo		255	28	28	FF1C1C
Akcent		252	252	28	FCFC1C

Tabulka 4.7: Paleta barev pro stav nadšení

Deprese

Položka adaptace	ilustrace	R	G	B	HEX
Pozadí		242	124	56	F27C38
Písmo		248	248	248	F8F8F8
Akcent		242	29	29	F21D1D

Tabulka 4.8: Paleta barev pro stav deprese

Úzkost

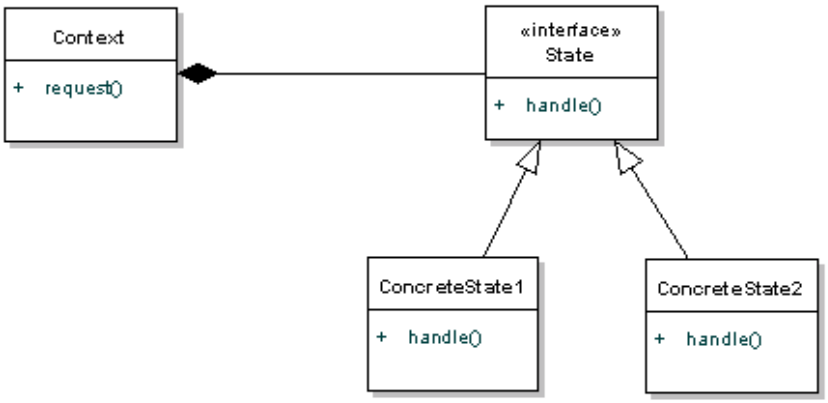
Zvětšení položek.

Položka adaptace	ilustrace	R	G	B	HEX
Pozadí		242	124	56	F27C38
Písmo		248	248	248	F8F8F8
Akcent		242	29	29	F21D1D

Tabulka 4.9: Paleta barev pro stav úzkosti

Použití stavů

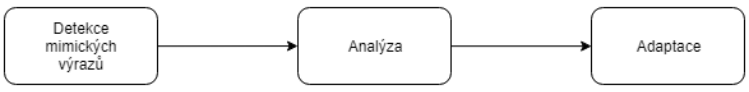
Pro jednotlivé stavy zavedeme ve vlastní implementaci vlastní třídy. K tomu nám poslouží návrhový vzor State, kde objekt mění své chování v závislosti na stavu. Každý stav si ve své třídě bude držet hodnoty pro adaptaci a klíčové hodnoty pro svou detekci. Použití tohoto návrhového vzoru nám umožní snadno přidávat a upravovat položky adaptace, jakož i modifikovat detekční hodnoty.



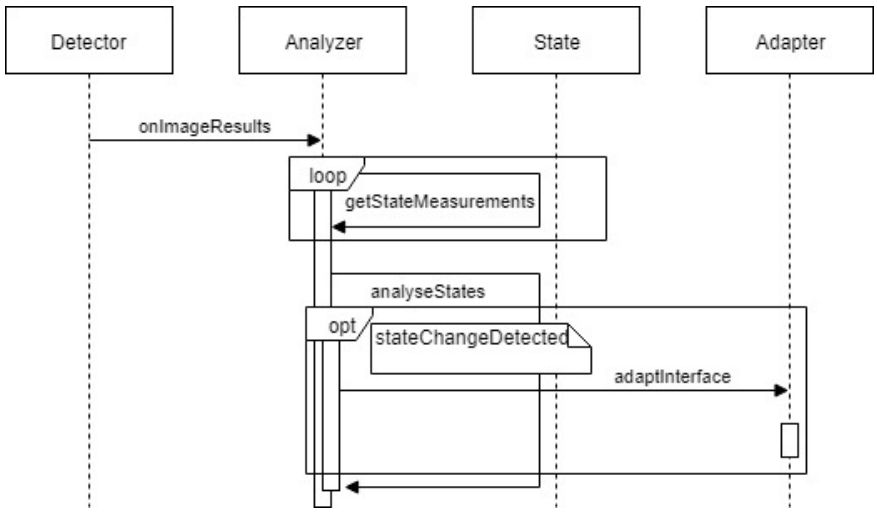
Obrázek 4.1: Ukázka, jak vypadá návrhový vzor state [37]

4.2 Funkce knihovny

Základními funkcemi knihovny jsou sběr dat, analýza sesbíraných dat a následná adaptace uživatelského rozhraní podle stavu uživatele (Obrázek 4.2). Sběr dat se provádí pomocí Affdex SDK přes přední kameru, kde se detekují jednotlivé mimické výrazy obličeje uživatele. V analýze rozhodujeme, zda nastala změna stavu uživatele a který ze aktivních stavů odpovídá uživatelovu stavu. Adaptace se provádí na základě hodnot uložených ve stavu v případě, že nastala změna stavu. Podrobnější průběh procesů je zobrazen na obrázku (Obrázek 4.3).



Obrázek 4.2: Obecný přehled procesů knihovny



Obrázek 4.3: Sekvenční diagram procesů knihovny

4.2.1 Sběr dat

O sběr dat se stará Affdex SDK od Affectiva. Pomocí přední kamery zmapuje obličej uživatele a vrátí hodnoty pro jednotlivé detekované emoce a výrazy. Tyto hodnoty dále budeme zpracovávat v analýze dat.

4.2.2 Analýza dat

Prvním krokem analýzy je zpracování sesbíraných dat. Z detekovaných hodnot, které dodá Affdex SDK, vybereme pouze ty, které potřebujeme k detekci stavů. Každý ze stavů má definovány jednu nebo dvě hodnoty, které slouží k jeho určení. Všechny hodnoty pro aktivní stavy zapíšeme do kolekce Map<K,V> s klíčem, který odpovídá názvu naměřené hodnoty. Kolekce Map<String, float> nám umožní snadný přístup k hodnotám pro jednotlivé stavy v další fázi analýzy.

Druhým krokem je test aktivních stavů. Naměřené hodnoty se porovnají s prahovými hodnotami každého stavu. Pokud žádný ze stavů neodpovídá aktuálnímu uživatelovu stavu (tedy žádné hodnoty nepřekročily nastavené prahy), použijeme neutrální (základní) stav.

Posledním krokem analýzy je kontrola, zda zvolený stav už není aplikovaný v aplikaci. Tato kontrola se provádí z důvodu vysokého počtu snímků od Affdex SDK, abychom neadaptovali rozhraní stále dokola na hodnoty stejného stavu. Pokud tato kontrola projde v pořádku, můžeme pokračovat na adaptaci uživatelského rozhraní.

4.2.3 Adaptace

Adaptace se provádí v případě, že se v analýze vyhodnotil jiný než aktuální stav. Prvním krokem je zjištění náležitostí adaptace ze zvoleného stavu. Každý ze stavů obsahuje informace o tom, jakou adaptaci rozhraní provádíme. Základem je změna barevné palety, které jsou u jednotlivých stavů popsány výše v kapitole 4.1. Dále je potřeba zjistit objekty, které budeme adaptovat. Následně se provedou potřebné změny na základě dat získaných ze stavu. Popsáno na obrázku 4.4

Obrázek 4.4: Znázornění adaptace

4.2.4 Zmapování struktury aplikace

Aplikace jsou obvykle složené z více aktivit (stránek), které mívají vzájemně odlišnou funkci či zobrazují různá data. Abychom adaptaci mohli použít pro celou aplikace (nebo její část), budeme potřebovat znát její strukturu.

K mapování struktury aplikace využijeme datovou strukturu stromu, která nám umožní popsat aplikaci od kořene, až po její nejkrajnější body aplikace. Kořenem zde bude hlavní/úvodní stránka aplikace, ze které se uživatel postupně dostává do ostatních částí aplikace (navigace), které mohou obsahovat další podčásti aplikace jako takové.

Každý bod navigace lze ve stromě reprezentovat jako uzel (Node). Zavedeme si tedy třídu Node, která bude uchovávat informace o rodiči, potomcích a informaci k vlastní identifikaci.

Každý tento uzel, tedy každá stránka aplikace, má svůj vlastní obsah a můžou se lišit i objekty, které stránka obsahuje.

Abychom s těmito objekty mohli pracovat, a tedy je i adaptovat, je nutné si je namapovat. Protože k identifikaci objektu stačí jeho vlastní ID, můžeme využít jednoduchý seznam (List), který zároveň umožňuje snadný a rychlý

přístup k těmto datům. Zavedeme tedy proměnnou typu List<Integer>.

Další proměnnou, kterou je potřeba si zavést pro následnou implementaci je proměnná typu View, která nám umožní zjištění výše uvedených dat.

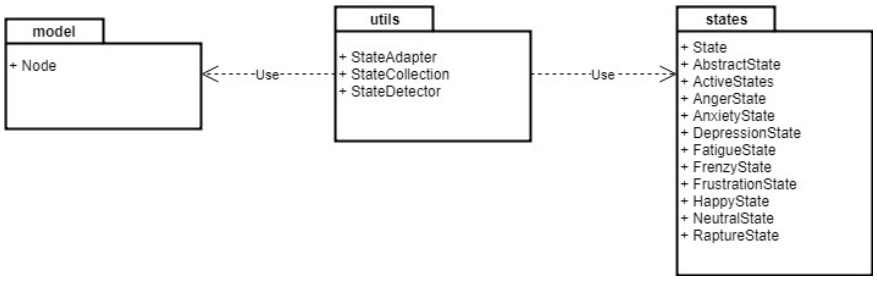
Kapitola 5

Implementace

V této kapitole probereme postup implementace projektu.

5.1 Struktura projektu

V kapitole analýza jsme navrhli jednotlivé části projektu. Celý projekt se bude skládat ze tří logicky rozdělených modulů zobrazených v následujícím diagramu 5.1.



Obrázek 5.1: Package Diagram projektu

Balík „model“ obsahuje třídu, která slouží k zmapování struktury aplikace do modelu stromu. Každá instance třídy Node obsahuje data o jednotlivých položkách (aktivitách) aplikace.

Balík „utils“ obsahuje třídy, které slouží k detekování stavu a adaptaci rozhraní. Tento balík obsahuje hlavní funkční třídy knihovny a slouží také ke komunikaci s aplikací.

Balík „states“ obsahuje rozhraní a třídy, které drží hodnoty a informace o stavech a které určují, co se bude dít v případě detekce určitého stavu.

5.2 Popis modelu aplikace

S ohledem na možnou složitost aplikace je nejprve potřeba zjistit, jakou strukturu daná aplikace má. Struktura bude zjišťovaná pomocí menu (navigace), kde každá z položek zpravidla odpovídá jedné aktivitě. Z toho důvodu je ale nutné, aby vývojář sdílel s knihovnou proměnnou Menu.

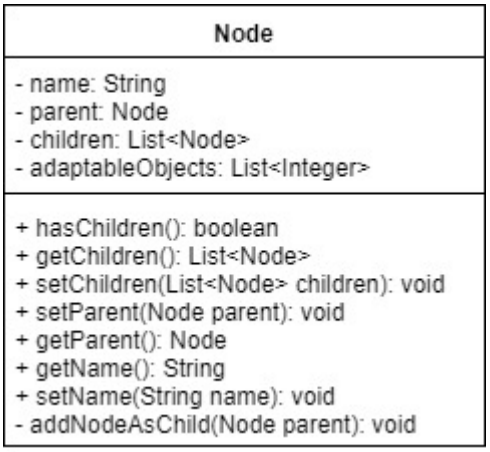
Po zavolání metody createApplicationStructure() s argumenty Activity, Menu a View se vytvoří kořenový Node (root), který reprezentuje úvodní aktivitu aplikace. Dalším krokem je analýza dotyčného xml souboru, který obsahuje informace o dalších položkách menu, v případě, že existují.

Dále je nutné zjistit objekty, které se vyskytují na daném Node (aktivitě). Nejprve musíme zjistit ID těchto objektů. To zjistíme pomocí proměnné typu ViewGroup příslušné aktivity zavoláním metody getChildAt(int id). Tato metoda nám vrátí objekt typu View, ze kterého dědí objekty, se kterými přichází uživatel Android systému do kontaktu. Příkladem takovýchto objetků mohou být tlačítka (Button, RadioButton), text (TextView), zaškrtávací box (CheckBox) nebo ukazatele průběhu (ProgressBar) a mnoho dalších tříd, které mohou být nalezeny v oficiální dokumentaci Androidu [1].

Vývojář má možnost si vybrat, které typy objektů chce modifikovat. V tom případě je ale nutné specifikovat všechny tyto typy objektů. Tuto možnost využije tak, že při volání metody createApplicationStructure(), přidá jako poslední argument (či více argumentů) název třídy vybraného objektu. Název třídy musí být převeden do typu String, k tomu slouží standardní metoda toString().

V případě, že chce vývojář mít proměnlivé všechny objekty, vynechá poslední argument při volání createApplicationStructure(). V tomto případě se defaultně budou adaptovat všechny typy objektů dané aktivity.

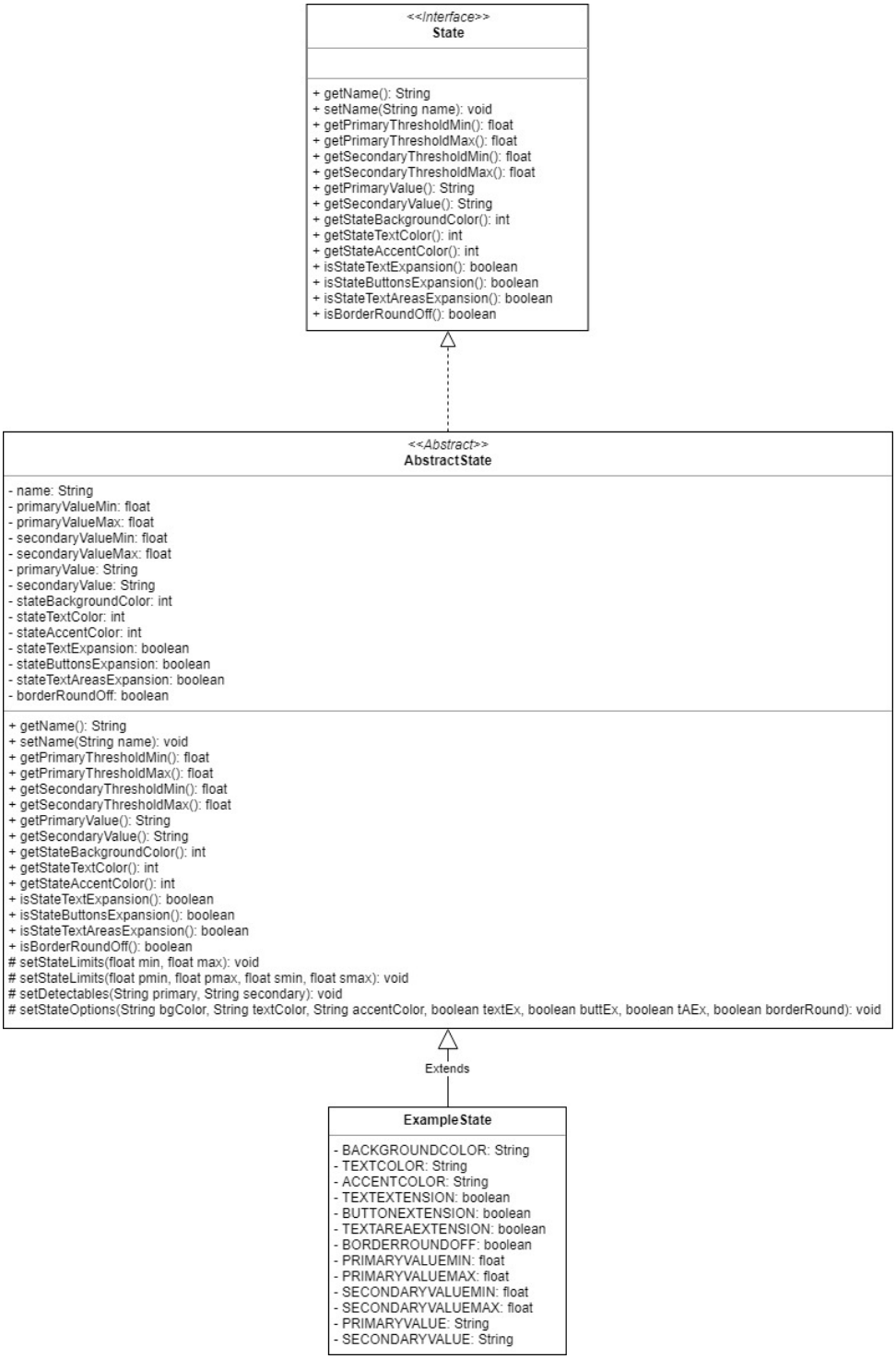
Třída Node pro popis struktury aplikace bude vypadat následovně diagram 5.2:



Obrázek 5.2: Třída pro sestavení modelu aplikace

5.3 Popis stavů aplikace

Jak už bylo popsáno v kapitole návrhové části, pro popis a práci se stavy byl zvolen návrhový vzor State. Chování jednotlivých objektů v závislosti na různých stavech se vzájemně liší, proto má každý stav svou vlastní třídu s informacemi, které ho popisují a které obsahují informace pro adaptaci. Celý popis je na následujícím diagramu 5.3



Obrázek 5.3: Třídy pro popis stavů s příkladovou třídou ExampleState

Třída AbstractState:

- String name – název stavu

• float primaryValueMin, primaryValueMax – stanovené hranice primární emoce nebo výrazu použitého pro detekci stavu

• float secondaryValueMin, secondaryValueMax – stanovené hranice sekundární emoce nebo výrazu použitého pro detekci stavu

• String primaryValeu, secondaryValue – názvy primární a sekundární emoce či výrazu použitého pro detekci stavu

• final int defaultBackgroundColor, defaultTextColor, defaultAccentColor – defaultní hodnoty barev pro adaptaci – použité v případě, že stav nemá definované vlastní barvy

• final int stateBackgroundColor, stateTextColor, stateAccentColor – hodnoty barev pro adaptaci

• boolean stateTextExpansion, stateButtonsExpansion, stateTextAreasExpansion, borderRoundOff – hodnoty popisující, zda se má měnit velikost či tvar některých objektů

Jednotlivé třídy stavů pak drží vlastní hodnoty pro popis adaptace a detekce.

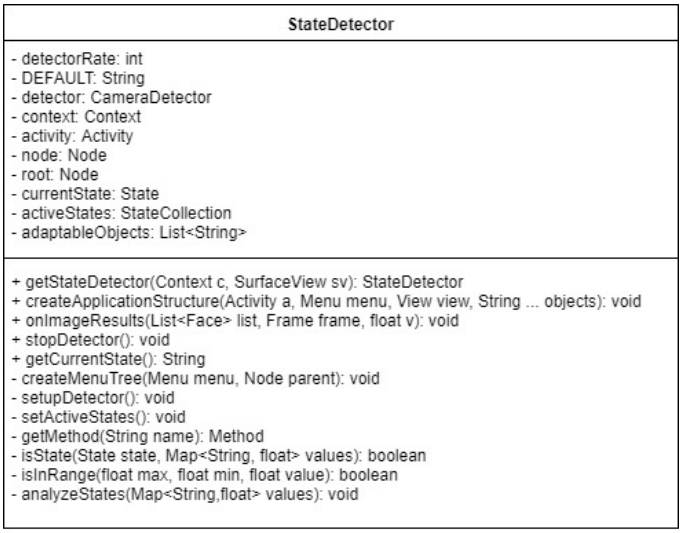
5.4 Třídy pro analýzu a adaptaci

Sběr dat se provádí přes Affdex SDK od Affectiva. Pro práci s tímto SDK a přístupu k datům je potřeba třída, která bude implementovat Detector.ImageListener a přepisovat metodu onImageResults() jejíž deklaraci lze vidět na výpisu 5.1. V této metodě se přistupuje k obdrženým datům z detektoru a další práce s těmito daty.

```
public void onImageResults(List<Face> list, Frame frame, float v) {
```

Listing 5.1: Deklarace metody onImageResults

K přístupu k datům z Affdex SDK máme třídu StateDetector, která se zároveň stará o zpracování těchto dat a určení stavu, ve kterém se uživatel nachází, na základě dat získaných právě z Affdex SDK.



Obrázek 5.4: Třída StateDetector pro analýzu stavů

Popis některých proměnných třídy:

• int detectorRate – hodnota určující rychlost snímkování kamery v Affdex SDK

• Node root – root Node aplikace – úvodní stránka

• State currentState – současný detekovaný stav

• StateCollection activeStates – aktivní stavy, používané k adaptaci uživatelského rozhraní

• List<String> adaptableObjects – seznam objektů k adaptování

Popis některých metod třídy:

• StateDetector getStateDetector() – získání instance třídy StateDetector, pokud neexistuje tak její vytvoření

• void createApplicationStructure() – zmapování struktury aplikace, získání objektů k adaptaci

• void onImageResults() – přepisovaná metoda z Affdex SDK, slouží k získání dat z detektoru. Získá z aktivních stavů informaci o tom, které hodnoty má zpracovávat a volá metodu analyzeStates().

• void createMenuTree() – privátní pomocná metoda pro vytvoření struktury aplikace

• void setActiveStates() – privátní pomocná metoda sloužící k inicializaci základních stavů

• Method getMethod() – privátní pomocná metoda sloužící k získání metody, která bude získávat hodnoty z Affdex SDK pro jednotlivé stavy

• boolean isState() – privátní pomocná metoda určující zda je daný stav aktuálně platný

• void analyzeStates() – privátní metoda sloužící k analýze aktuálního stavu uživatele, volá si pomocnou metodu isState(). V případě, že zjistí změněný stav volá adaptaci.

Před začátkem detekování se musí spustit detektor Affdex SDK, ten spouštíme v konstruktoru při vytváření instance třídy StateDetector voláním metody 5.2 setupDetector().

```
private void setupDetector(){
   detector = new CameraDetector(context,
       CameraDetector.CameraType.CAMERA_FRONT, cameraPreview);
   if(detector == null){
      System.err.println("Error in setting up Detector");
      return;
   }
   detector.setDetectAllEmotions(true);
   detector.setDetectAllExpressions(true);
   detector.setImageListener(this);
   detector.setMaxProcessRate(detectorRate); //The maximum
       processing rate
   detector.start();
}
```

Listing 5.2: Nastavení detektoru Affdex SDK

V metodě onImageResults při detekování tváře zpracujeme hodnoty, které určují jednotlivé stavy. Pro neutrální stav žádné hodnoty neměříme, používáme ho pouze v případě, že žádný stav neodpovídá naměřeným hodnotám.

```
Face face = list.get(0);
Map<String, Float> values = new HashMap<>();
for (State state : activeStates.getStatesCollection()) {
   if(state.getName().equalsIgnoreCase(DEFAULT)) {
      System.out.println("NEUTRAL STATE DETECTED, skipping");
      continue;
   } else {
      String primary = state.getPrimaryValue();
      String secondary = state.getSecondaryValue();
      float primaryValue = 0;
      final Method primaryMethod = getMethod(primary);
      if(primaryMethod == null) {
      System.err.println("NO METHOD FOR STATE "+state.getName() +
          "FOUND.");
      continue;
   }
   try {
```

```
      if(primary.startsWith("emotions")) {
         primaryValue = (float) primaryMethod.invoke(face.emotions);
      } else {
         primaryValue = (float)
             primaryMethod.invoke(face.expressions);
      }
   } catch (IllegalAccessException e) {
      e.printStackTrace();
   } catch (InvocationTargetException e) {
      e.printStackTrace();
   };
   values.put(primary,primaryValue);
```

Listing 5.3: Získání primární emoce či výrazu pro detekci pro jednotlivé stavy

```
if(secondary != "") {
   float secondaryValue = 0;
   final Method secondaryMethod = getMethod(secondary);
   if(secondaryMethod == null) {
      continue;
   }
   try {
      if(secondary.startsWith("emotions")) {
         secondaryValue = (float)
             secondaryMethod.invoke(face.emotions);
      } else {
         secondaryValue = (float)
             secondaryMethod.invoke(face.expressions);
      }
   } catch (IllegalAccessException e) {
      e.printStackTrace();
   } catch (InvocationTargetException e) {
      e.printStackTrace();
   }
   values.put(secondary,secondaryValue);
}
```

Listing 5.4: Získání sekundární emoce či výrazu pro detekci pro jednotlivé stavy – v případě že existuje

Testujeme jednotlivé stavy vůči naměřeným hodnotám. Neutrální stav netestujeme, používáme ho pouze v případě, že žádný jiný stav neodpovídá naměřeným hodnotám. V případě, že detekovaný stav odpovídá stavu, který je v současné době aplikovaný, adaptaci neprovádíme.

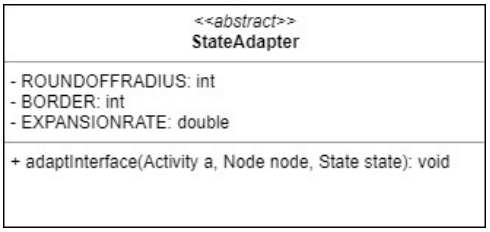
```
private void analyzeStates(Map<String, Float> values) throws
    ClassNotFoundException, NoSuchMethodException,
    InvocationTargetException, InstantiationException,
    IllegalAccessException {
   Collection<State> states = activeStates.getStatesCollection();
   State oldCurrentState = currentState;
   boolean stateFound = false;
```

```
    if(activeStates != null) {
       for (State state : states) {
          if(state.getName().equals(DEFAULT)) {
             System.out.println("NEUTRAL STATE TEST, skipping");
             continue;
          }
          if(isState(state, values)) {
             currentState = state;
             stateFound = true;
             break;
          }
       }
    }
    if(!stateFound) {
       currentState = activeStates.getState(DEFAULT);
       stateFound = true;
    }
    if(stateFound && !oldCurrentState.equals(currentState)) {
       StateAdapter.adaptInterface(activity, node, currentState);
    }
 }
```

Listing 5.5: Metoda pro analýzu stavů

Instanci této třídy také vývojář volá v aplikaci pro využití knihovny, konkrétně metody getStateDetector() a createApplicationStructure().

Po analýze, pokud je to nutné, je potřeba adaptovat uživatelské rozhraní na hodnoty odpovídající vybranému stavu. K tomu slouží abstraktní třída StateAdapter. Pro adaptaci je nutné vědět, které objekty budeme adaptovat. Co konkrétně potřebujeme vědět je ID těchto objektů. Tyto ID objektů jsou uloženy ve stromu aplikace, tedy v instancích třídy Node. Jednotlivá View (tedy objekty k adaptaci) si vyvoláme pomocí volání metody findViewById() na instanci aktivity. Posledním krokem je pak aplikace adaptačních informací jednotlivé objekty. Tyto procesy se provádí ve metodě adaptInterface(), která je veřejně dostupná a přijímá tři argumenty typu Activity, Node a State.



Obrázek 5.5: Abstraktní třída StateAdapter pro adaptaci uživatelského rozhraní

Kapitola 6

Testování

Testování funkčnosti jsme provedli použitím demo aplikace, kde jsme zkoumali přizpůsobení aplikace jednotlivým nadefinovaným stavům.

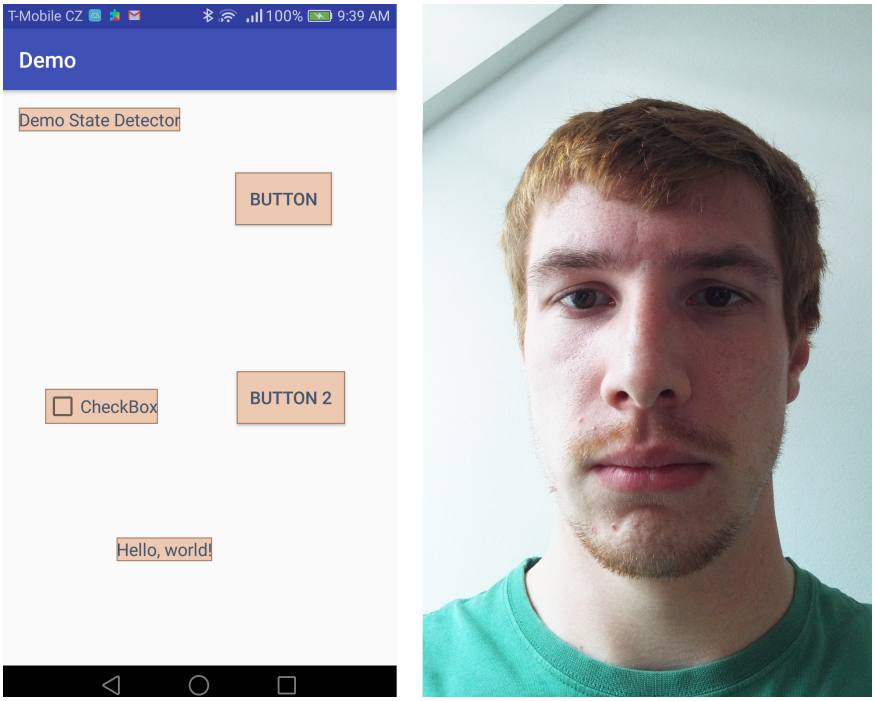
Postup testování byl následující:

1) instalace aplikace v debug modu

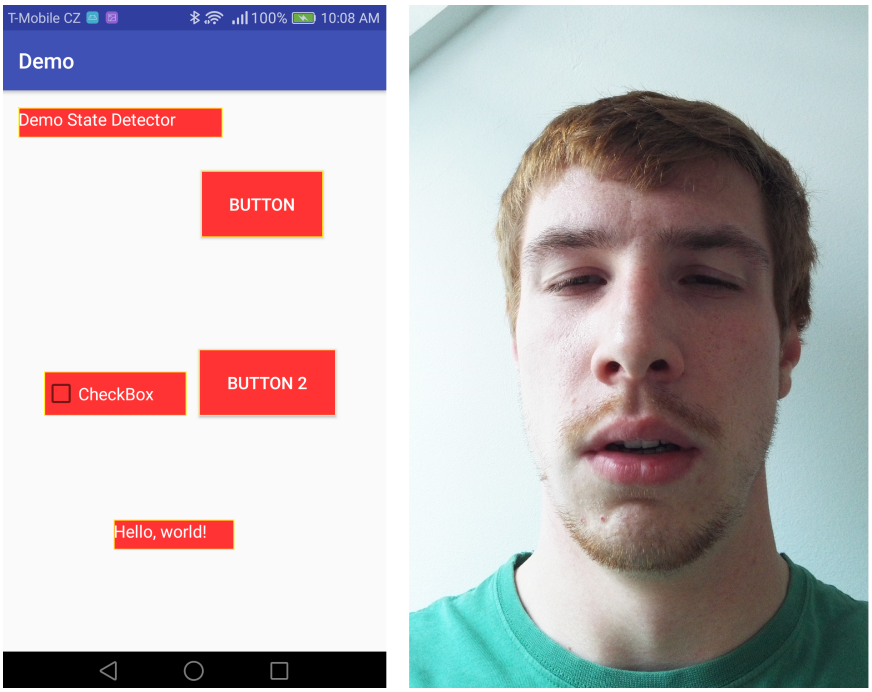
2) test detekce obličeje

3) test detekce jednotlivých stavů a úspěšné adaptace UI

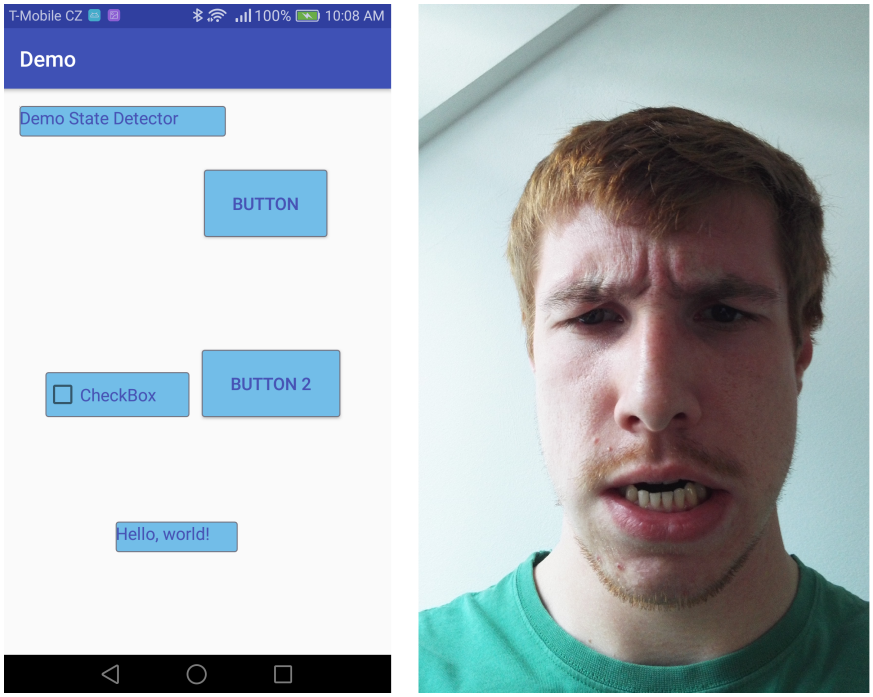
výsledky:



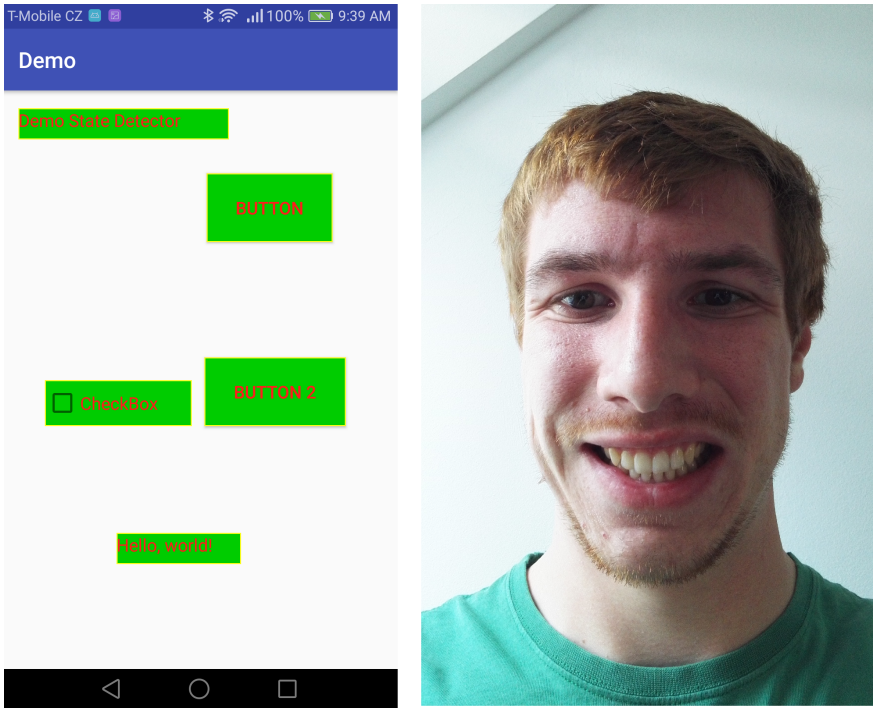
Obrázek 6.1: Výsledek demo app pro test neutrálního stavu



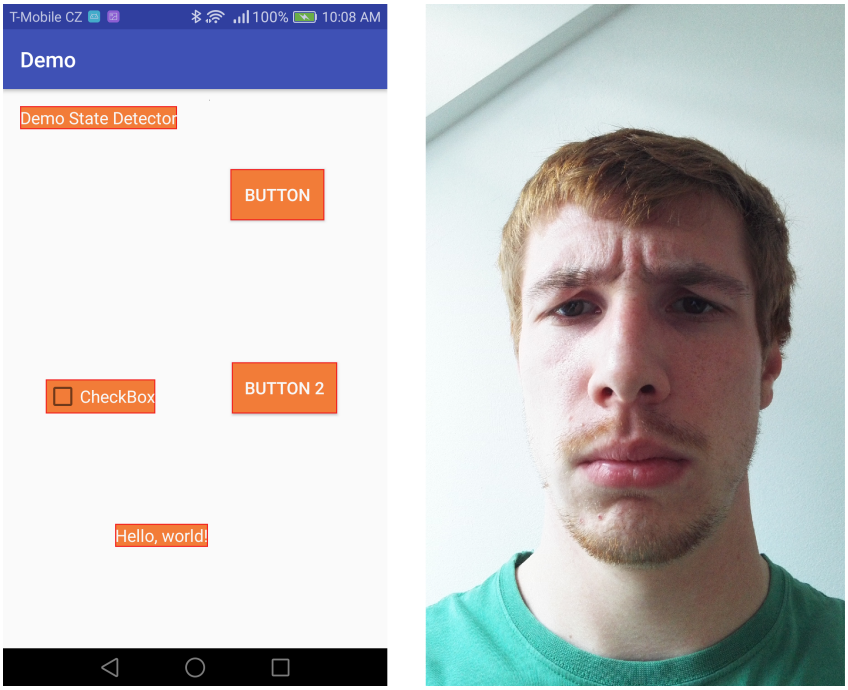
Obrázek 6.2: Výsledek demo app pro test stavu únavy



Obrázek 6.3: Výsledek demo app pro test stavu zuřivosti



Obrázek 6.4: Výsledek demo app pro test stavu nadšení



Obrázek 6.5: Výsledek demo app pro test stavu deprese

Test instalace a spuštění úspěšný. Test detekce obličeje úspěšný. Test detekce jednotlivých stavů a následné adaptace UI úspěšný.

6.1 Návrh dalších testů

Další testování pro zpřesnění detekce stavů a pro vylepšení knihovny v další práci. Navrhujeme následující testování. V první fázi budeme zkoumat přesnost detekce a určení stavů. Ve druhé fázi budeme zkoumat navrhované adaptace a spokojenost uživatelů s nimi. Základem tohoto testování s uživateli je dostatečný vzorek uživatelů.

V první fázi je tedy potřeba sesbírat dostatečné množství dat, abychom ověřili správnost a případně mohli upravit hranice stavů. Znamená to tedy, že zpětná vazba od testovacích uživatelů nebude pouze odpověď „ano/ne“ na dotaz, zda se v daném stavu nacházel, ale také budeme chtít získat informaci, v jakém, jiném než detekovaném, stavu se daný uživatel nacházel.

Sběr dat od testovacích uživatelů by probíhal pomocí testovací aplikace – jednoduché aplikace implementované pouze za tímto účelem. V této aplikaci by neprobíhala žádná adaptace, pouze by se sbíraly a odesílaly data. Aplikace musí mít obsahovat následující funkce:

- Detekce stavu uživatele

- Zobrazení detekovaného stavu
- Potvrzení/odmítnutí navrhovaného stavu
- V případě odmítnutí stavu – výběr, případně specifikace jiného stavu
- Uložení hodnot z detekčního systému (Affdex SDK)
- Přiřazení a uložení hodnot zadaných testovacím uživatelem
- Volitelně: Odeslání hodnot z detekčního systému a hodnot zadaných uživatelem k vývojáři

Druhá fáze testování, tedy uživatelská spokojenost, je velmi obtížně testovatelná, jednak kvůli velké variaci v jednotlivých aplikacích a za druhé kvůli subjektivitě a preferencích uživatelů. Proto zde navrhujeme testování provést v jednotlivých aplikacích vývojářů standardní formou UX testování.

Kapitola 7

Instalace

V této kapitole se nachází informace o hardwarových a softwarových požadavcích ke zprovoznění knihovny a postup jakým připojit knihovnu k aplikaci.

7.1 Požadavky

Minimální hardwarové a softwarové požadavky pro práci s knihovnou:

- Operační systém Android ve verzi 4.4 (KitKat)
- Přítomnost přední kamery
- Aplikace využívá Java ve verzi 8
- V základním layoutu aplikace (standardně activity_main.xml) vytvořený CameraPreview alespoň v minimalistickém provedení

```
<SurfaceView
   android:id="@+id/cameraPreview"
   android:layout_width="1px"
   android:layout_height="1px"
   android:background="@android:color/transparent"
   />
```

Listing 7.1: Vytvoření minimalistického CameraPreview

7.2 Instalace

Postup zpřístupnění knihovny aplikaci:

1) Dodat knihovnu (soubor classes.jar) do složky „<aplication_name>/app/libs“

2) Do souboru build.gradle (Project: <aplication_name>) dodat do allprojects -> repositories Maven Affectivy:

```
maven {
url "http://maven.affectiva.com"
}
```

Listing 7.2: Součást souboru build.gradle (Project)

3) Do souboru build.gradle (Module: app) dodat do dependecies:

```
compile 'com.affectiva.android:affdexsdk:3.+'
compile 'com.android.support:appcompat-v7:24.2.1'
```

Listing 7.3: Součást souboru build.gradle (app)

4) Do souboru AndroidManifest.xml přidat následující:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
...

   <uses-permission
       android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
   <uses-permission android:name="android.permission.INTERNET" />
   <uses-permission
       android:name="android.permission.ACCESS_NETWORK_STATE" />
   <uses-permission android:name="android.permission.CAMERA" />

   <application
tools:replace="android:allowBackup,android:label"
```

Listing 7.4: Součást souboru AndroidManifest.xml

5) V souboru MainActivity.java deklarovat proměnnou typu StateDetector

```
public class MainActivity extends AppCompatActivity {

   private StateDetector stateDetector;

...
```

Listing 7.5: Deklarování proměnné typu StateDetector

a. v metodě onCreate() inicializovat proměnnou typu StateDetector

```
@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);
```

```
stateDetector = StateDetector.getStateDetector(this,
    (SurfaceView) findViewById(R.id.cameraPreview));
```

Listing 7.6: Doplnění metody onCreate()

b. v metode onCreateOptionsMenu() zavolat na proměnné typu StateDetector metodu createApplicationStructure()

```
@Override
public boolean onCreateOptionsMenu (Menu menu) {
   stateDetector.createApplicationStructure(this, menu,
       getLayoutInflater().inflate(R.layout.activity_main, null));
```

Listing 7.7: Příklad volání createApplicationStructure()

c. v metodě onStop() zavolat na proměnné typu StateDetector metodu stopDetector()

```
@Override
public void onStop(){
   super.onStop();
   stateDetector.stopDetector();
}
```

Listing 7.8: Příklad volání stopDetector()

6) Při instalaci aplikace je nutné povolit aplikaci přistupovat k přední kameře

Kapitola 8

Závěr

Cílem práce bylo projít problematiku emocí a stavu uživatele a zjistit, jaký vztah mezi nimi je. Dále probrat problematiku adaptace uživatelského rozhraní v závislosti na stavu uživatele. V souvislosti s emoční stránkou uživatele byla prozkoumána i závislost mezi barvami a emocemi, což umožnilo navrhnout barevnou paletu pro adaptaci rozhraní tak, aby působila na uživatele pozitivním dojmem. Výsledná knihovna by měla přinést spokojenost jak uživateli, kterému se díky ní přizpůsobí aplikace stavu a pozitivně ho ovlivní, tak vývojáři, který díky ní získá spokojenějšího uživatele.

V úvodní části práce je zkoumána teorie emocí a jejich měření z mimiky uživatele pomocí Affdex SDK. Dále je zkoumána teorie stavu uživatele a jeho změn. V neposlední řadě se zabývá problematikou uživatelského rozhraní a adaptace.

Druhá část práce se zabývá návrhem knihovny pro adaptaci uživatelského rozhraní v závislosti na stavu uživatele a následnou implementací tohoto návrhu. Poslední část se zabývá návrhem testování s uživateli pro zdokonalení knihovny. Výsledkem této práce je knihovna, která je snadno rozšiřitelná a upravitelná, co se týká stavů a případných dalších adaptací, a která umožňuje Android aplikaci snadnou adaptaci v závislosti na detekovaném stavu uživatele.

Literatura

[1] Android - view. https://developer.android.com/reference/android/view/View. [Online, cit. 20.05.2018].

[2] Color Psychology: The Emotional Effects of Colors. http://www.arttherapyblog.com/online/color-psychology-psychologica-effects-of-colors/. [Online; cit. 16.05.2018].

[3] Drug Aware - Getting The Facts. https://drugaware.com.au/getting-the-facts/. [Online; cit. 11.04.2018].

[4] Psychological Disorders. http://www.psychone.net/psychological-disorders.php. [Online; cit. 15.04.2018].

[5] Psychological Properties Of Colours. http://www.colour-affects.co.uk/psychological-properties-of-colours. [Online; cit. 16.05.2018].

[6] Psychology dictionary. https://psychologydictionary.org/reflective-consciousness/. [Online, cit. 20.05.2018].

[7] user interface (UI). https://searchmicroservices.techtarget.com/definition/user-interface-UI, 2016. [Online; cit. 18.04.2018].

[8] Global market share held by the leading smartphone operating systems in sales to end users from 1st quarter 2009 to 2nd quarter 2017. https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/, 2018. [Online; cit. 22.03.2018].

[9] Number of smartphone users worldwide from 2014 to 2020 (in billions). https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/, 2018. [Online; cit. 22.03.2018].

[10] M. Abulkhair, A. H. Alsahli, K. M. Taleb, F. M. Bahran, H. A. Alzahrani, and L. F. Ibrahim. Mobile platform detect and alerts system for driver fatigue. *Procedia Computer Science*, 62:555–564, 2015.

[11] Affectiva. Affectiva - facial landmarks. `https://knowledge.affectiva.com/v3.2/docs/facial-landmarks-1`, note =.

[12] Affectiva. Affectiva - metrics. `https://developer.affectiva.com/metrics/`, note =.

[13] Affectiva. Affectiva developer homepage. `https://developer.affectiva.com/`, note =.

[14] Affectiva. Affectiva-mit facial expression dataset (am-fed): Naturalistic and spontaneous facial expressions collected in-the-wild. `http://www.affectiva.com/wp-content/uploads/2017/03/Affectiva-MIT-Facial-Expression-Dataset-AMFED-Naturalistic-and-Spontaneous-pdf` note =.

[15] R. Amalberti, P. Falzon, and N. Carbonell. User representations of computer systems in human-computer speech interaction. *International Journal of Man-Machine Studies*, 38(4):547–566, 1993.

[16] I. C. Association. `http://www.aic-color.org/journal/v1/jaic_v1_01.pdf`. [Online, cit. 16.05.2018].

[17] C. C. Bell. States of consciousness. *Journal of the National Medical Association*, 72(4):331–334, 1980.

[18] N. M. Burton. What are basic emotions? `https://www.psychologytoday.com/us/blog/hide-and-seek/201601/what-are-basic-emotions`, 2016. [Online, cit. 20.05.2018].

[19] K. Cherry. Color Psychology: Does It Affect How You Feel? `https://www.verywellmind.com/color-psychology-2795824`, 2018. [Online; cit. 16.05.2018].

[20] C. Christensen and N. Gwozdziewycz. Revision of the apa division 30 definition of hypnosis. *American Journal of Clinical Hypnosis*, 57(4):448–451, 2015. PMID: 25928784.

[21] Confucius. *The Book of Rites (Li Ji): English-Chinese Version.*

[22] B. Dainton. Temporal consciousness. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, fall 2017 edition, 2017.

[23] W. Dong and X. Wu. Fatigue detection based on the distance of eyelid. In *Proceedings of 2005 IEEE International Workshop on VLSI Design and Video Technology, 2005.*, pages 365–368, May 2005.

[24] P. Ekman and R. J. Davidson. *The Nature of Emotion: Fundamental Questions*. 1st edition, 1994.

[25] J. He, S. Roberson, B. Fields, J. Peng, and S. Cielocha. Fatigue Detection using Smartphones. `https://www.omicsonline.org/open-access/fatigue-detection-using-smartphones-2165-7556.1000120.php?aid=21520`, 2013. [Online; cit. 11.03.2018].

[26] N. Kaya and H. H. Epps. Relationship between color and emotion: A study of college students. 38, 09 2004.

[27] A. Lunova. *Adaptace UI založená na emocích uživatele*. Bachelor thesis, Department of Computer Science, CTU FEE, 2017. Leadership Ing. Jiří Šebek.

[28] N. Mishchenko. *Aspektově orientovaný vývoj adaptivní struktury aplikace pro Java EE aplikace*. Bachelor thesis, Department of Computer Science, CTU FEE, 2017. Leadership Ing. Jiří Šebek.

[29] J. S. Nairne. *Psychology: The Adaptive Mind*. Wadsworth Publishing Co Inc, 1999.

[30] R. W. Pickard. *Affective Computing (MIT Press)*. MIT Press, 1997.

[31] R. Plutchik. Plutchik's wheel of emotions. `https://upload.wikimedia.org/wikipedia/commons/c/ce/Plutchik-wheel.svg`.

[32] A. Revonsuo, S. Kallio, and P. Sikka. What is an altered state of consciousness? *Philosophical Psychology*, 22(2):187–204, 2009.

[33] Y. Shimbun. Blue streetlights believed to prevent suicides, street crime. `https://www.seattletimes.com/nation-world/blue-streetlights-believed-to-prevent-suicides-street-crime/`, 2008. [Online, cit. 16.05.2018].

[34] B. SHNEIDERMAN. The future of interactive systems and the emergence of direct manipulation. *Behaviour & Information Technology*, 1(3):237–256, 1982.

[35] R. Sternberg. *In Search of the Human Mind*. Wadsworth Publishing, 2nd edition, 1998.

[36] T. Sundelin, M. Lekander, G. Kecklund, E. J. W. Van Someren, A. Olsson, and J. Axelsson. Cues of fatigue: Effects of sleep deprivation on facial appearance. *Sleep*, 36(9):1355–1360, 2013.

[37] J. Surgue. State pattern tutorial with java examples. `https://dzone.com/articles/design-patterns-state`. [Online, cit. 01.05.2018].

[38] R. Van Gulick. Consciousness. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, spring 2018 edition, 2018.

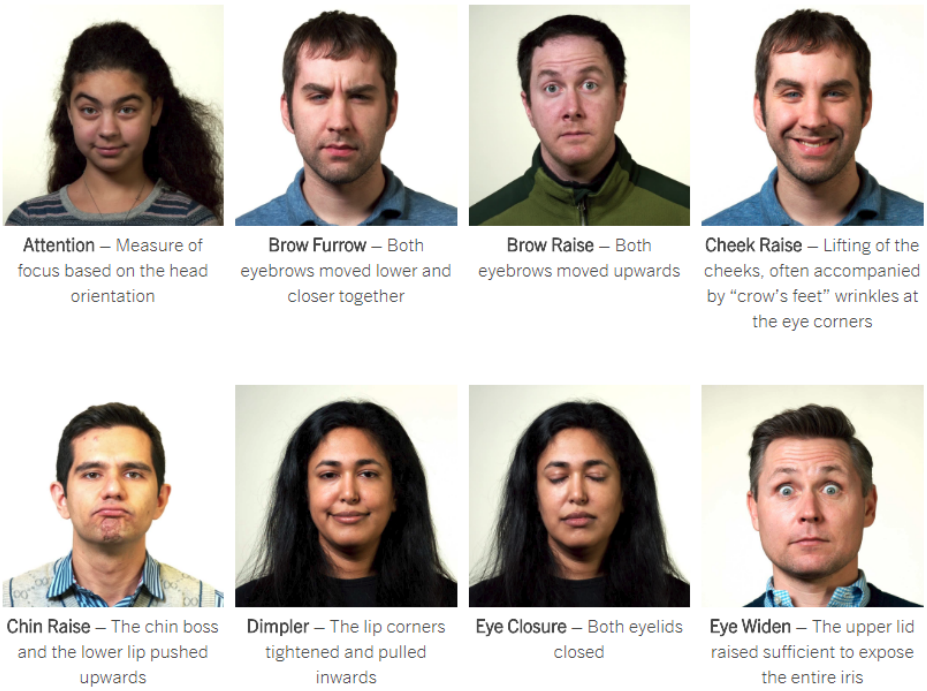
Příloha A

Seznam použitých zkratek

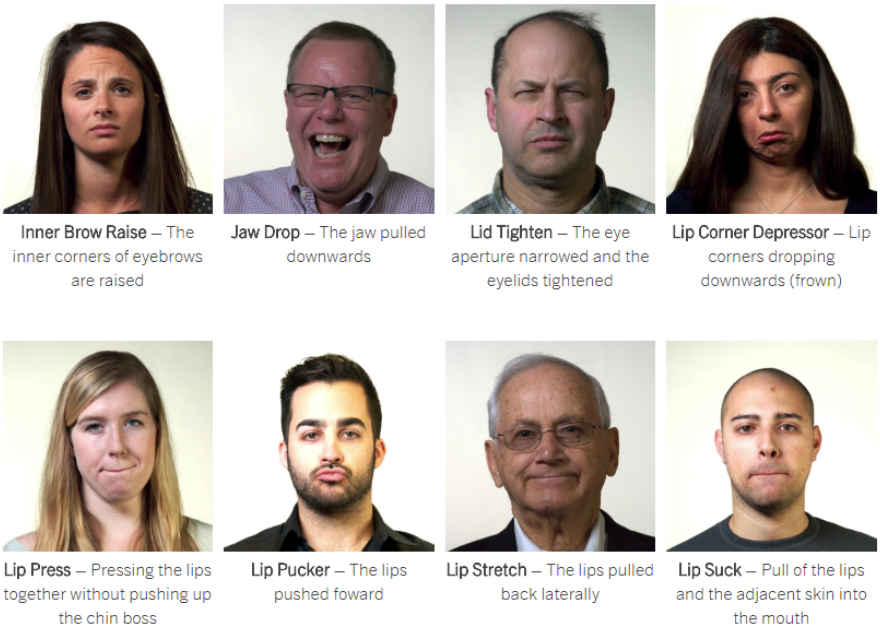
UI	User Interface = uživatelské rozhraní
SW	Software
SDK	Software Development Kit = sada nástrojů pro vývoj
ASC	Altered State of Consciousness = pozměněný stav vědomí
HUMAINE	Human-Machine Interaction Network on Emotion
EARL	Emotion Annotation and Representation Language
EMFACS	Emotional Facial Action Coding System
AU	Action Unit
ROI	Region of Interest

Příloha B

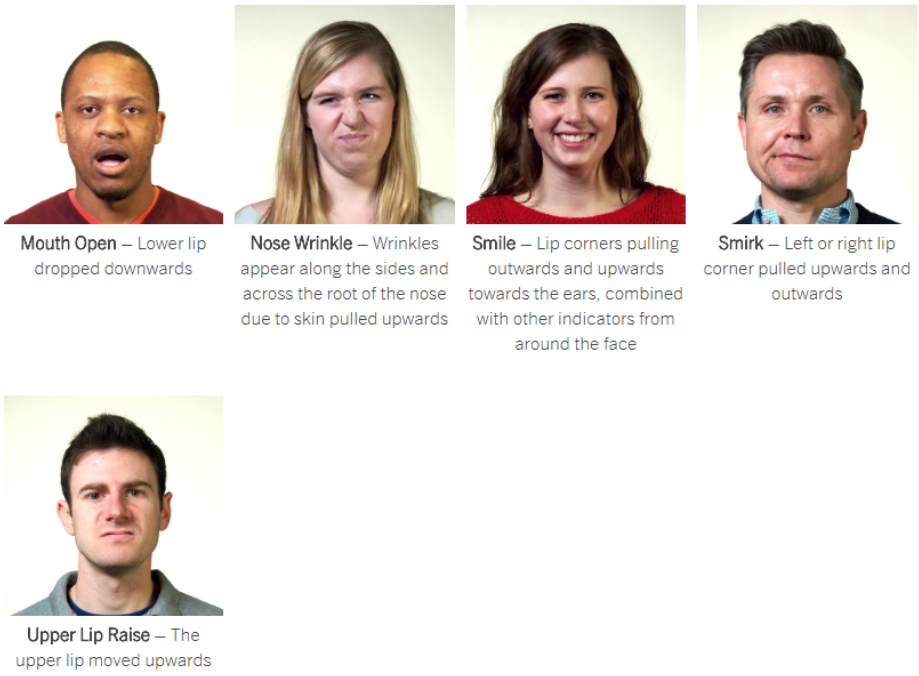
Detekované mimické výrazy



Obrázek B.1: Detekované mimické výrazy Affdex SDK 1/3 [12]



Obrázek B.2: Detekované mimické výrazy Affdex SDK 2/3 [12]



Obrázek B.3: Detekované mimické výrazy Affdex SDK 3/3 [12]

Příloha C

Obsah přiloženého CD

- UserStateAdaptLibrary.zip - projekt implementace knihovny
- Demo.zip - demo aplikace pro vyzkoušení knihovny
- zadani.pdf - zadání bakalářské práce
- bakalarska-prace.pdf - bakalářská práce
- bp-tex.zip - bakalářská práce v TeX formě ...