Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava 4

# Emotion Log

## Tímový projekt

## Dokumentácia k riadeniu projektu

Bc. Michal Biroš
Bc. Tomáš Caban
Bc. Tomáš Kunka
Bc. Filip Staňo
Bc. Tomáš Lekeň
Bc. Milan Martinkovič
Bc. Bálint Szilva

Vedúci tímového projektu: Ing. Martin Labaj
Ak. rok: 2012/13

# Obsah

# 1 Úvod

Táto dokumentácia obsahuje dokumenty k riadeniu projektu Emotion log. Tento projekt je riešený tímom EMO pod vedením Ing. Martina Labaja v zložení Bc. Michal Biroš, Bc. Tomáš Caban, Bc. Tomáš Kunka, Bc. Filip Staňo, Bc. Tomáš Lekeň, Bc. Milan Martinkovič a Bc. Bálint Szilva.

Druhá kapitola tohto dokumentu obsahuje ponuku tímu na vypracovanie troch vybratých tém, preferenciu tém a rozvrh tímu v zimnom semestri. Obsahom tretej kapitoly je dlhodobý plán a krátkodobé plány viažuce sa na jednotlivé šprinty. V štvrtej kapitole sú uvedené úlohy členov tímu pri riešení projektu. Piata kapitola obsahuje zápisnice z jednotlivých stretnutí tímu. V šiestej kapitole sú uvedené metodiky, použité pri vývoji produktu.

# 2 Ponuka

## *2.1 Členovia tímu*

**Bc. Tomáš Kunka**
Bakalárske štúdium ukončil v študijnom odbore Informatika na FIIT STU. Má skúsenosti s programovacími jazykmi C, C++ a Java. Počas štúdia pracoval s mapovými technológiami OpenStreetMap a Google Maps, ktoré použil aj pri implementácii svojej bakalárskej práce – Pamiatky na mapách. Pri jej vypracovaní sa zdokonalil v práci s HTML, JavaScriptom, a MySQL.

**Bc. Tomáš Lekeň**
Bakalárske štúdium ukončil na FIIT STU v Bratislave v študijnom programe Informatika. Počas bakalárskeho štúdia pracoval vo firme, kde mal možnosť spolupracovať pri vývoji niekoľkých informačných systémov, vyvíjaných ako webové aplikácie. Taktiež mal možnosť vyvíjať aplikácie pre mobilnú platformu Android. Vďaka štúdiu a práci získal vedomosti aj praktické skúsenosti v oblasti algoritmizácie aj vývoja softvéru. Z technológií sa zdokonalil najmä v programovacích jazykoch C/C++, Java, JavaScript a databázovej technológii PostgreSQL.

**Bc. Michal Biroš**
Bakalalárske štúdium absolvoval na FIIT STU v Bratislave v odbore Informatika. Počas štúdia pracoval vo firme, kde vyvíjal webový informačný systém. Vďaka tomu získal skúsenosti so správou verzií softvérových produktov, návrhom databáz, ale aj v programovaní s využitím návrhového vzoru MVC. Ovláda programovacie jazyky Java, C#, PHP, Javascript a MySQL.

**Bc. Tomáš Caban**
Bakalárske štúdium ukončil na FIIT STU v Bratislave v odbore Informatika. Jeho bakalárska téma sa zaoberala tvorbou mashup aplikácií, pri ktorej získal znalosti z oblasti webových aplikácií. Popri štúdiu pracuje v oblasti telekomunikácií, kde získal praktické skúsenosti z oblasti programovania a testovania informačných systémov. Jeho hlavným jazykom je Java, ovláda tiež C/C++, PHP, ActionScript, XML, HTML. Z databázových systémov ma praktické skúsenosti s PostgreSQL, Oracle a MySQL.

**Bc. Milan Martinkovič**
Absolvoval bakalárske štúdium na FIIT STU v Bratislave v odbore Informatika. Medzi jeho hlavné záujmy v oblasti informačných technológií patria najmä webové aplikácie o čom svedčí aj fakt, že svoju bakalársku prácu realizoval ako webovú aplikáciu a momentálne je popri škole aj v zamestnaní, ktoré je orientované na webové aplikácie. Stredobodom jeho záujmu je však aj práca so systémami, ktoré nemusia byť založené len na webových technológiách. Ovláda programovacie jazyky C, Java, PHP, MySQL ale aj JavaScript či značkovací jazyk HTML.

**Bc. Filip Staňo**

Vyštudoval bakalársky stupeň štúdia na FIIT STU v Bratislave v štúdijnom odbore Informatika. Ovládaprogramovacie jazyky C, C#, JavaScript, PHP(základy), taktiež jazyky HTML, CSS. Má pracovné skúsenosti s webovým dizajnom a databázovým systémom. Vo voľnom čase sa venuje hlavne práci s .NET a taktiež webovým aplikáciam. Pri bakalárskej práci sa venoval vyhľadávaniu notového zápisu.

**Bc. Bálint Szilva**

Bakalárske štúdium absolvoval na FIIT STU v Bratislave v študijnom odbore Informatika. Bakalársku prácu vypracoval na tému “Mobilná aplikácia pre počítanie pravdepodobnosti výhry v Pokri”. Počas štúdia získal hlbšie vedomosti s programovacími jazykmi C, C++, Java, PHP, HTML, CSS, Javascript a s databázou MySQL. Vo voľnom čase sa venuje webovým technológiám a grafike.

## *2.2 Odporúčanie pre inteligentnú TV*

### 2.2.1 Motivácia

Táto téma nás oslovila pretože, vidíme jej prínos pre obyčajných ľudí. So zvyšovaním množstva programov, seriálov a filmov je neustále ťažšie si vybrať čo chceme pozerať. Niekedy strávime viac času prepínaním programov ako ich samotným sledovaním. Často premeškáme programy a filmy s našimi obľúbenými hercami, ktoré by nás mohli zaujať. Každý človek je iný, preto mu rôzne hodnotenia nepovedia o programe to podstatné. Film s vysokým hodnotením na IMDB nemusí byť vhodný pre každého a v každej chvíli. Nálada, duševný a fyzický stav môžu ovplyvniť výber filmu viac ako odporúčania cudzích ľudí. A naopak film môže zhoršiť, alebo zlepšiť našu náladu. Taktiež filmy, ktoré si vybrali naši priatelia sa budú pravdepodobne páčiť aj nám. Existuje množstvo faktorov, ktoré ovplyvňujú náš názor na film. Napríklad obľúbení herci, režiséri, rok výroby, žáner, náš vek a krajina pôvodu filmu. História videných filmov dokáže napovedať niečo bližšie o danom človeku a určiť jeho preferencie v budúcnosti. Výzvou je pre nás aj určenie preferencií netradičným spôsobom – minihra, test. Hudobný žáner a hudba ako taká vyvoláva v ľuďoch rôzne emócie. Preto aj filmy s určitým hudobným štýlom sú pre niekoho prioritou. Viaceré filmy a seriály sú preslávené svojimi zvučkami a soundtrackmi. Pri tejto téme nás zaujala aj možnosť jej prepojenia s internetom. Ako hlavné zlepšenia vidíme ušetrenie času a zjednodušenie výberu multimediálneho obsahu.

### 2.2.2 Koncepcia riešenia

Pri riešení tohto projektu by sme chceli vytvoriť systém, ktorý by používateľom poskytoval relevantné odporúčania filmov, seriálov a televíznych kanálov na základe viacerých faktorov.

Hlavnou časťou systému by mala byť serverová aplikácia, ktorá by bola schopná zbierať a vyhodnocovať údaje z rôznych zariadení a platform. Používatelia by pre získanie odporúčania mohli použiť aplikácie vo svojich mobilných telefónoch, tabletoch, počítačoch, ale aj aplikácie v televízore. Filmy a seriály by si používatelia mohli pozrieť presne na tých istých zariadeniach, čo by sme mohli docieliť využitím napríklad služieb ako Netflix, AppleTV, Youtube a ďalších. Po dopozeraní, respektíve prerušení filmu by sa tak aplikácia mohla diváka opýtať, ako sa mu to páčilo alebo prečo vypol prehrávanie. Tým by sa jednoduchým a nenásilným spôsobom dostali do systému údaje, ktoré by používateľ v iných odporúčacích systémoch s veľkou pravdepodobnosťou zabudol vyplniť.

Používateľ by mal možnosť prepojiť svoj účet na Facebooku, Googli alebo Twitteri s účtom v odporúčacom systéme. Medzi faktory pre výber odporúčaných filmov by sa tak dal zahrnúť aj sociálny faktor. Ak by si používateľ zmenil stav na Facebooku zo zadaného na slobodný, mohli by mu z odporúčaní zmiznúť romantické filmy, ktoré by určite iba prehĺbili jeho depresiu.

Systém by bol vďaka jednotnému API jednoducho rozšíriteľný nielen o ďalšie služby, ale aj prídavné zariadenia. Napríklad, pomocou kamery by bolo možné zistiť, či sú v miestnosti deti a z odporúčaní by boli vyradené horory alebo iné neprístupné programy.

## *2.3 Odhaľovanie emocionálneho stavu používateľa*

### 2.3.1 Motivácia

Práca na počítači je v dnešnej dobe každodennou realitou. Čoraz väčší dôraz sa kladie na dodržiavanie nesplniteľných termínov odovzdania prác, v dôsledku čoho môžu byť používatelia frustrovaní a robiť pri práci s počítačom čoraz viac chýb. Preto je viac ako vhodné aplikovať systém, ktorý by bol schopný včas identifikovať zvyšujúce sa napätie, prípadne klesajúcu produktivitu práce používateľa a vhodným spôsobom sa pokúsiť o nápravu tohto stavu.

Tento projekt je pre nás zaujímavý najmä z hľadiska vlastných skúseností s prácou na počítači, ktorá si vo väčšine prípadov vyžadovala vysokú mieru sústredenia. Práve na základe týchto skúseností by bolo pre nás lákavé vypracovať systém, ktorý by sa pokúsil rozoznať a odbúrať tieto stavy a dopomôcť používateľovi udržiavať pracovné tempo a prijateľnú náladu. Zaujímavou je táto téma aj vďaka tomu, že interaguje s používateľom ako ľudskou bytosťou, tz. pokúša sa rozoznať a určiť emočný stav a následne sa snaží mu na základe tohto emočného rozpoloženia prispôsobiť. Túto tému vnímame ako prácu na rozvoji umelej inteligencie vo vzťahu používateľ – počítač, fascinujúca je práve myšlienka prispôsobenia počítača používateľovi. Ďalšou výraznou výhodou je široká použiteľnosť takéhoto systému vďaka faktu, že by mohol byť použitý pri rôznych činnostiach s počítačom, ktoré si vyžadujú určitú mieru sústredenia, od bežnej práce s textovým editorom až po implementovanie rozsiahlych informačných systémov.

### 2.3.2 Koncepcia riešenia

Cez zisťovacie médium, ktorým by bola klávesnica a myš, by bolo možné zisťovať aktuálny stav používateľa napríklad meraním frekvencie výskytu chýb alebo množstvom práce, ktorú dokázal vyprodukovať za určitý čas. Dôležitým aspektom môže byť aj štýl písania, ktorý by bol charakterizovaný dĺžkou stlačení jednotlivých kláves a časom písania jednotlivých slov a podobne. Taktiež by bolo možné rozoznať aktuálny stav používateľa na základe práce s myšou, kde by sa sledovala rýchlosť stláčania, prípadne pohybu.

Na začiatok potrebujeme získať informácie o tom, ako používateľ reaguje v bežnom stave, tz. aké ma štandardné tempo písania slov, stláčania myši a podobne. Na základe nezhôd v týchto jednotkách by sme následne dokázali vyhodnotiť určitý neštandardný stav v správaní, napríklad časté mazanie písmen prostredníctvom klávesy Backspace, prípadne na základe nejakého slovníku by sme dokázali rozoznať zvýšenie výskytu gramatických chýb pri písaní textov.

Tieto stavy by sme chceli riešiť viacerými spôsobmi. V prvom rade by sme používateľovi navrhli pravidelné ukladanie vykonanej práce, pretože práve na toto by mohol v podobných situáciach zabúdať. Taktiež v prípade výskytu gramatických chýb, by sme používateľovi poskytli možné správne tvary slov, ktoré by súviseli s daným slovom. V prípade častého výskytu chýb by sme používateľovi navrhli menšiu prestávku. Do nášho riešenia by sme mimo iné chceli zakomponovať aj akýsi relaxačný mód, ktorý by bolo možné spustiť po dosiahnutí kritického emočného stavu. Tento mód by mal pomôcť k zlepšeniu aktuálneho

Ponuka stavu používateľa formou zobrazenia rôzneho multimediálneho obsahu ako napríklad hudba, obrázky, vtipy a podobne. Tento mód si bude môcť používateľ taktiež nakonfigurovať, aby sme dosiahli, že pokus o uvoľnenie atmosféry bude blízky jeho vkusu.

Rozoznávanie emočného stavu používateľa by bolo možné realizovať aj na základe snímania tváre používateľa pomocou kamery. Na začiatku by bol používateľ odfotený v bežnom stave. Následne by sa určili kritické body jeho tváre v bežnom stave. Týmito bodmi by boli najmä obočie a pery, na základe ktorých dokážeme rozoznať hnev, beznádej, či radosť. Čiastočne by sa dalo vychádzať zo systému FACS (http://en.wikipedia.org/wiki/Facial_Action_Coding_System).

Hnev by sme mohli reprezentovať takto:

http://persuasive.net/wp-content/uploads/2009/6/microexpressions-anger.jpg

Na tomto obrázku môžme evidovať smútok, prípadne beznádej, napríklad keď sa používateľ nedokáže ďalej „pohnúť“ pri práci na projekte.

http://xfinity.comcast.net/blogs/tv/files/2009/04/use_sadness.jpg

Oba spomenuté prístupy k zisťovaniu emočného stavu používateľa sú veľmi zaujímavé a každý z nich má svoje výhody a nevýhody. Pri možnosti voľby by sme však mali prioritný záujem o prácu so vstupmi cez klávesnicu a myš.

## *2.4 FIIT Kinect*

### 2.4.1 Motivácia

Používateľov fascinujú netradičné a inovatívne formy ovládania a zadávania príkazov, ako sú napríklad ovládanie hlasom alebo gestami. Práve ovládanie gestami považujeme za zaujímavý a jednoduchý spôsob ovládania s perspektívnou budúcnosťou. Jeho hlavnou výhodou je možnosť ovládať najrôznejšie zariadenia intuitívne, bez nutnosti používať veľké množstvo ovládačov. Taktiež umožňuje vykonávať viacero operácií naraz, vykonaním jediného gesta, čo dovoľuje definovať používateľom často vykonávané alebo obľúbené akcie. Ovládanie pomocou gest tiež predstavuje určité zjednodušenie pre hendikepovaných ľudí.

Práve senzor Kinect disponuje kamerou a infračervenými hĺbkovými senzormi, umožňujúcimi vytvárať hĺbkovú mapu, dôležitú pre rozpoznávanie vykonávaných gest. Funkcia rozpoznávania tváre umožní každému požívateľovi prispôsobiť si jeho vlastnú množinu gest.

Počítačovú grafiku považujeme za zaujímavú oblasť informatiky, o ktorej by sme si chceli prehĺbiť poznatky, k čomu nám táto téma môže významne pomôcť. Taktiež Kinect je podľa nás veľmi zaujímavý projekt, ktorý prináša nové možnosti v oblasti ovládania počítačov, ktoré je možné rozšíriť aj na iné zariadenia, čomu by sme sa radi v tomto projekte venovali.

### 2.4.2 Koncepcia riešenia

Nevyhnutným krokom bude oboznámenie sa všetkých členov tímu v dostatočnej miere so všetkými technológiami, nevyhnutnými pre vývoj aplikácií pre senzor Kinect. Taktiež bude potrebné analyzovať existujúce riešenia, do akej miery je možné využiť už existujúce softvérové moduly, a čo bude nutné implementovať od začiatku. Pri hľadaní informácií k tejto téme sme našli niekoľko knižníc, ktoré umožňujú vývoj aplikácií pre senzor Kinect (CL NUI, OpenCV, OpenKinect), takže bude dôležité analyzovať ich funkcie, možnosti a zvoliť vhodné technológie.

Hlavnou časťou riešenia bude systém, ktorý bude umožňovať definovanie a následné rozpoznávanie gest používateľa, ktoré potom bude možné použiť na ovládanie rôznych elektrických prístrojov v domácnosti. V tejto časti bude dôležité implementovať detekciu a rozpoznávanie pohybu používateľa z hĺbkovej mapy získanej pomocou senzora Kinect.

Ako jednu časť riešenia navrhujeme využiť rozpoznávanie tváre na možnosť personalizovať ovládacie gestá pre rôznych používateľov aplikácie. Toto umožní definovať každému používateľovi gestá podľa jeho preferencie, pričom systém jednotlivým gestám priradí správny príkaz na základe identifikácie používateľa rozpoznaného podľa tváre. Vďaka tomu odpadá nutnosť prihlasovať sa pre identifikáciu používateľa.

Ďalšia časť riešenia sa bude zaoberať možnosťami priradenia definovaných gest používateľa akciám pre ovládanie rôznych elektrických spotrebičov v domácnosti. Systém bude umožňovať jednoduchými intuitívnymi gestami vykonávať aj viacero operácií naraz. Používateľ tak bude môcť definovať napríklad svoje obľúbené či často vykonávané akcie, ale aj ovládať viacero prístrojov naraz. Používateľ by mal mať na výber čo najväčšie množstvo

prístrojov a v tejto fáze bude dôležité analyzovať možnosti a nástroje pre komunikáciu s rôznymi zariadeniami v domácnosti prostredníctvom Bluetooth, WiFi či IrDA. Pri hľadaní informácií k tomuto projektu sme zistili, že výrobcovia SMART televízií poskytujú API pre prístup k funkciám TV, rovnako je možné pristupovať k funkciám mobilných zariadení s operačnými systémami Android a iOS.

## *2.5 Príloha A – Preferencia tém*

1. Inteligentná TV
2. Emócie
3. Kinect
4. Demonštrácia
5. Hra
6. Offline Web
7. Veda
8. Robocup

## *2.6 Príloha B – Rozvrh tímu*

| Deň | Meno | 7:00-7:50 | 8:00-8:50 | 9:00-9:50 | 10:00-10:50 | 11:00-11:50 | 12:00-12:50 | 13:00-13:50 | 14:00-14:50 | 15:00-15:50 | 16:00-16:50 | 17:00-17:50 | 18:00-18:50 | 19:00-19:50 | 20:00-20:50 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PON | Filip Staňo | | VINF | | | | | | | | TP1 | | VSS | | |
| | Tomáš Kunka | | | | | | | | | | | | | | |
| | Milan Martinkovič | | | | | | | | | | | | | | |
| | Tomáš Caban | | | | | | | | PDBT | | | | | | |
| | Tomáš Lekeň | | | | | | | | | | | | | | |
| | Michal Biroš | | | | | | | | | | | | VIS | | |
| | Bálint Szilva | | | | | | | | | | | | | | |
| UT | Filip Staňo | KOD | | | | | | | | MSI | | MSI | | MSI | |
| | Tomáš Kunka | | | | | | | | | | | | | | |
| | Milan Martinkovič | | | | | | | | | | | | | | |
| | Tomáš Caban | | | | | | | | | | | | | | |
| | Tomáš Lekeň | | | | | | | | | | | | | | |
| | Michal Biroš | | | | | | | | | | | | | | |
| | Bálint Szilva | | | | | | | | | | | | | | |
| STR | Filip Staňo | | | | | | | | | | | | | | |
| | Tomáš Kunka | | | | | | | | | | | | | | |
| | Milan Martinkovič | | | | | | | | | | | | | | |
| | Tomáš Caban | | | | | | | | | | | | | | |
| | Tomáš Lekeň | | | | | | | | | | | | | | |
| | Michal Biroš | | | | | | | | | | | | | | |
| | Bálint Szilva | | | | | | | | | | | | | | |
| ŠT | Filip Staňo | KOD | | | | | | | | ASS | | | | | |
| | Tomáš Kunka | | | | | | | | | | | | | | |
| | Milan Martinkovič | | | | | | | | | | | | | | |
| | Tomáš Caban | | | | | | | | | | | | | | |
| | Tomáš Lekeň | | | | | | | | | | | | | | |
| | Michal Biroš | | | | | | | | | | | | | | |
| | Bálint Szilva | | | | | | | SOGAM | | | | | SOGAM | | |
| PIA | Filip Staňo | | | | | | | | | | | | | | |
| | Tomáš Kunka | | | | | | | | | | | | | | |
| | Milan Martinkovič | | | | | | | | | | | | | | |
| | Tomáš Caban | | | | PDBT | | | | | | | | | | |
| | Tomáš Lekeň | | | | | | | | | | | | | | |
| | Michal Biroš | | | | | | | | | | | | | | |
| | Bálint Szilva | | | | | | | | | | | | | | |

# 3 Plán

Nakoľko sa pri vývoji nášho projektu riadime metodológiou SCRUM, tak aj podoba nasledovného plánu, ktorý uvádzame v tejto kapitole je taktiež touto agilnou metódou ovplyvnený. Princíp SCRUM sa používa najmä na projekty, pri ktorých je plánovanie dopredu problematickou úlohou. Plánovanie sa teda zameriava na menšie časové úseky, v ktorých sú nadchádzajúce úlohy oveľa transparentnejšie a tým pádom je ich možné aj ľahšie naplánovať. Tieto časové úseky predstavujú jednotlivé šprinty a ich priebeh sa plánuje na špeciálnych stretnutiach, ktoré sa uskutočňujú na samom začiatku jednotlivých šprintov. Dĺžka trvania jedného šprintu sú dva týždne. Samozrejme aj pri takomto prístupe k riadeniu projektu je potrebné mať predstavu o dôležitých míľnikoch, ktoré nás v priebehu vývoja softvéru budú čakať.

Na základe vyššie uvedených slov preto uvádzame jednak úlohy, ktoré boli naplánované pre jednotlivé šprinty ale aj dôležité body vývoja nášho projektu, ktoré ešte len nastanú z dlhodobého hľadiska.

## *3.1 Termíny šprintov*

Nasledujúca tabuľka popisuje dvojtýždňové intervaly, v ktorých prebiehajú jednotlivé šprinty.

| **Číslo šprintu** | **Termín šprintu** |
|---|---|
| 1. | 17.10-31.10 |
| 2. | 31.10-14.11 |
| 3. | 14.11-28.11 |
| 4. | 28.11-12.12 |

## *3.2 Plány šprintov*

V tejto sekcii uvádzame naplánované úlohy pre jednotlivé šprinty.

### 3.2.1 Šprint 1 (Audi)

Témou prvého šprintu bolo vyriešenie úloh nevyhnutných pre prácu na projekte. Jednalo sa najmä o analýzu jednotlivých riešení, ktoré by mohli mať súvis s našou témou a mohli by sme z nich čerpať, poprípade ich do nášho riešenia zakomponovať. Ďalšou témou úloh bolo zabezpečenie rôznych náležitostí ako napríklad inštalácia webového servera, založenie projektu v nástroji Redmine a podobne.

| **Úloha** | **Riešiteľ** | **Začiatok** |
|---|---|---|
| Nainštalovať server | Bálint Szilva | 23.10.2012 |

| | | |
|---|---|---|
| Vytvorenie plagátu | Michal Biroš | 17.10.2012 |
| Vytvorenie a nasadenie webovej stránky tímu | Bálint Szilva | 17.10.2012 |
| Vytvorenie projektu v Redmine | Bálint Szilva | 17.10.2012 |
| Analyzovať riešenia na sledovanie emócií | Filip Staňo | 17.10.2012 |
| Analyzovať možnosti licencie FaceAPI | Filip Staňo | 17.10.2012 |
| Analyzovať možnosti strojového učenia | Tomáš Lekeň | 17.10.2012 |
| Analyzovať sledovanie zvuku cez mikrofón | Tomáš Caban | 17.10.2012 |
| Analyzovať sledovanie pohľadu používateľa a natočenia tváre | Milan Martinkovič | 17.10.2012 |
| Analyzovať logger | Michal Biroš | 17.10.2012 |
| Analyzovať odporúčania pre jednotlivé emočné stavy | Tomáš Kunka | 17.10.2012 |
| Spracovať obraz pomocou face.com | Bálint Szilva | 24.10.2012 |

### 3.2.2 Šprint 2 (Bentley)

V druhom šprinte sme sa zameriavali na praktické aplikovanie analyzovaných riešení. Predmetom šprintu boli aj náležitosti ako tvorba dokumentácie, ktorá súvisy s jedným z míľnikov v nasledovnej kapitole. Riešila sa aj konfigurácia systému na spravovanie verzií riešení, ktorým je MS team foundation server.

| **Úloha** | **Riešiteľ** | **Začiatok** |
|---|---|---|
| Získavať obraz z kamery | Milan Martinkovič | 31.10.2012 |
| Spracovať obraz pomocou študentského projektu HED | Filip Staňo, Tomáš Kunka | 31.10.2012 |
| Konfigurácia TFS a Webu | Bálint Szilva | 31.10.2012 |
| Tvorba dokumentácie | Všetci | 31.10.2012 |
| Skompilovanie programu na spracovanie zvuku | Tomáš Caban | 31.10.2012 |
| Inštalácia služby do IIS | Michal Biroš | 31.10.2012 |
| Reprezentácia údajov z kamery | Tomáš Lekeň | 31.10.2012 |
| Príprava údajov na odosielanie | Tomáš Caban | 31.10.2012 |
| Preposielanie dát | Michal Biroš | 31.10.2012 |

## *3.3 Dlhodobý plán*

Náš dlhodobý plán sa skladá z dôležitých míľnikov, ktoré budú vývoj nášho projektu sprevádzať, ako aj z cieľov, ktoré sme si naplánovali splniť.

### 3.3.1 Zimný semester

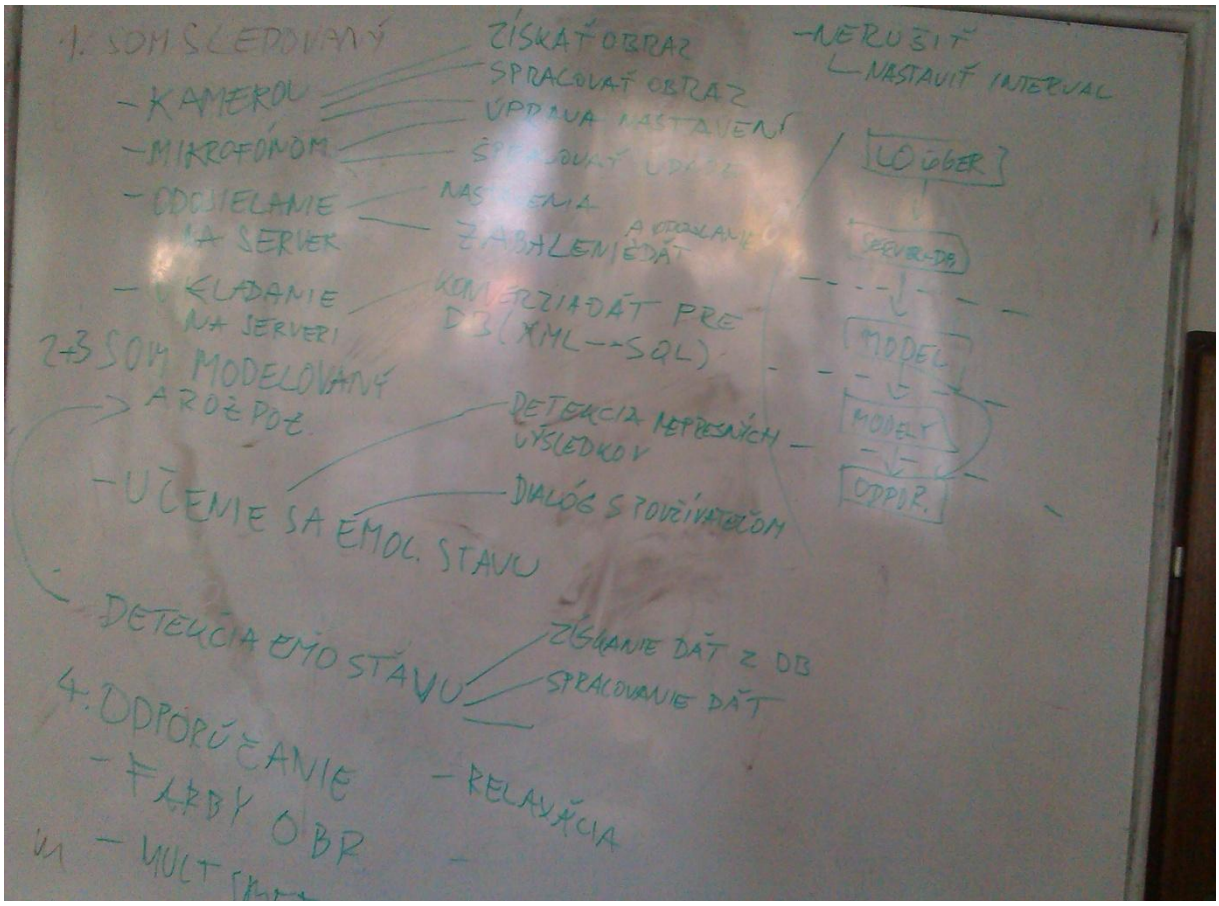
| Popis | Typ | Dátum splnenia |
|---|---|---|
| Analyzovať dostupné riešenia | Cieľ | 31.10.2012 |
| Integrácia spracovania obrazu | Cieľ | 14.11.2012 |
| Odovzdanie dokumentácie | Míľnik | 14.11.2012 |
| Prihlásenie na TP Cup | Míľnik | 26.11.2012 |
| Rozoznanie emócií | Cieľ | 28.11.2012 |
| Zostavenie prototypu | Cieľ | 12.12.2012 |
| Odovzdanie prototypu | Míľnik | 14.12.2012 |

### 3.3.2 Letný semester

Dôležité míľniky v letnom semestri, nám ešte nie sú známe. Ciele ešte nie sú priamo definované ale bude sa jednať primárne o strojové učenie aplikované na rozoznanie emócií, modelovanie používateľov a rôzne odporúčania súvisiace s identifikovaným emočným stavom používateľa.

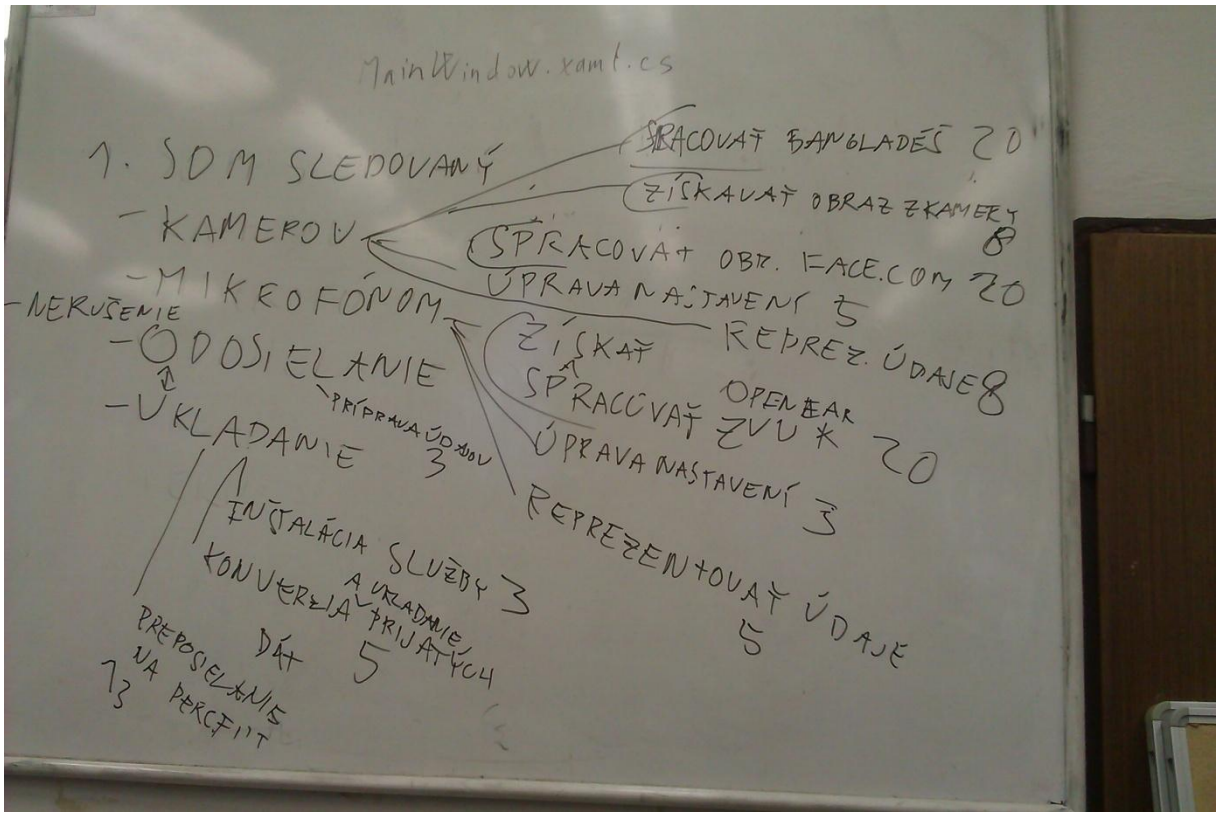
## *3.4 Produktové požiadavky*

Produktové požiadavky(angl. *Produck Backlog*), sú dôležitým aspektom v rámci plánovania, nakoľko obsahujú hlavné náležitosti, ktoré má výsledné riešenie napĺňať. Tieto jednotlivé položky sú rozčleňované na menšie úlohy, ktoré sa riešia počas jednotlivých šprintov. Produktové požiadavky a k nim identifikované úlohy sú zobrazené na nasledovnom obrázku. Podoba požiadaviek sa môže v priebehu vývoja projektu obmieňať. Na obrázkoch uvádzame aktuálnu podobu produktových požiadaviek.




**Obrázok 1** Produktové požiadavky a k nim identifikované úlohy

Na ďalšom obrázku je zobrazené jednotlivé úlohy, ktoré sme podrobnejšie identifikovali k prvej produktovej požiadavke spolu s odhadovanou náročnosťou. Riešenie týchto úloh bolo následne predmetom šprintu.



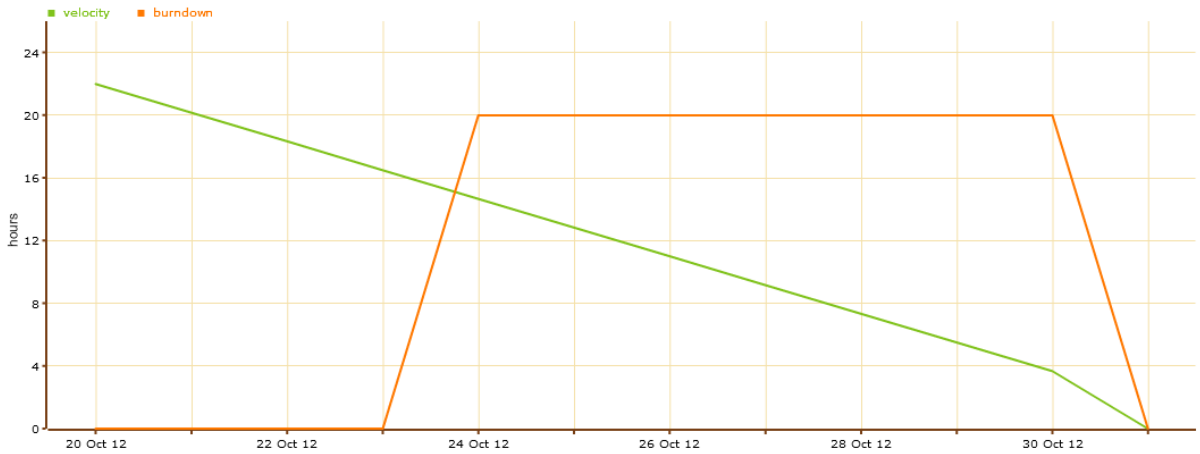

**Obrázok 2** Odhad náročnosti jednotlivých úloh

## *3.5 Priebeh plánovania*

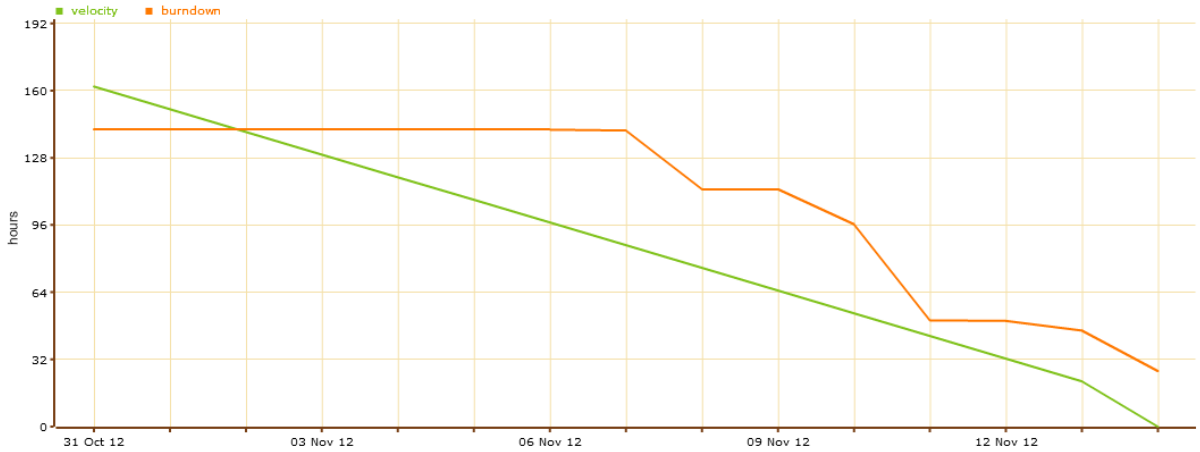
Pomocou metodiky SCRUM sa riadime prvý krát a to sa odzrkadľuje aj na úspešnosti nášho plánovania. Naplánované úlohy sa darí napĺňať v akceptovateľnej miere, avšak problémy sa objavovali pri odhade náročnosti jednotlivých úloh a to najmä v prvom šprinte, čo je pochopiteľné.

V druhom šprinte sa vyskytol problém s úlohou spracovania obrazu pomocou projektu HED, ktorý sa nakoniec ukázal byť ako nevhodný najmä oproti riešeniu, ktoré sa nám podarilo identifikovať až v tomto šprinte (názov riešenia: Luxand). Preto sa odhadnuté náročnosti opäť vychyľovali od reálnych hodnôt.

V nasledujúcich šprintoch sa budeme snažiť týchto chýb vyvarovať a pokúsime sa naše odhady náročností zlepšiť.




**Obrázok 3** Burndown graf pre prvý šprint




**Obrázok 4** Burndown graf pre druhý šprint

# 4 Úlohy členov tímu

## *4.1 Dlhodobé úlohy*

Vedúci tímu – Filip Staňo
Manažér podpory vývoja – Bálint Szilva
Manažér rozvrhu a plánovania – Milan Martinkovič
Manažér rizík - Tomáš Kunka
Manažér monitorovania projektu – Michal Biroš
Manažér kvality – Tomáš Caban
Manažér dokumentovania – Tomáš Lekeň

## *4.2 Krátkodobé úlohy*

**Filip Staňo**

- Analýza riešenia na sledovanie emócií
- Spracovanie obrazu pomocou projektu HED

**Bálint Szilva**

- Príprava servera a webovej stránky
- Spracovanie obrazu pomocou face.com
- Konfigurácia TFS

**Milan Martinkovič**

- Analýza sledovania pohľadu používateľa a natočenia tváre
- Získavanie obrazu z kamery

**Tomáš Kunka**

- Analýza odporúčaní pre emočné stavy
- Spracovanie obrazu z kamery

**Michal Biroš**

- Analýza prebratého softvérového riešenia
- Inštalácia služby do IIS
- Preposielanie dát

**Tomáš Caban**

- Analýza sledovania zvuku cez mikrofón
- Získanie a spracovanie zvuku cez mikrofón

**Tomáš Lekeň**

- Analýza strojového učenia
- Získanie a spracovanie zvuku cez mikrofón
- Tvorba dokumentácie

# 5 Zápisnice

## *5.1 Zápisnica zo stretnutia číslo 1*

**Téma stretnutia:** Úvodné stretnutie
**Dátum:** 3.10.2012
**Čas:** 14:00
**Miesto:** Softvérové laboratórium

**Vedúci stretnutia:** Ing. Martin Labaj
**Zapisovateľ:** Bc. Tomáš Lekeň

**Zúčastnení:** Bc. Michal Biroš
Bc. Tomáš Caban
Bc. Tomáš Kunka
Bc. Filip Staňo
Bc. Tomáš Lekeň
Bc. Milan Martinkovič
Bc. Bálint Szilva

**Priebeh stretnutia:**

- Zahájenie spoločného stretnutia tímov 10 a 14 pod vedením vedúcich tímov doc. Mgr. Daniely Chudej, PhD. a Ing. Martina Labaja
- Predstavenie tímov
- Informovanie tímov o projekte
- Dohodli sme sa na práci metodikou SCRUM
- Diskusia o podporných nástrojoch pri tvorbe softvérového produktu
- Rozdelenie úloh členom tímu

**Úlohy:**

| Úloha | Riešiteľ | Dátum vzniku | Obtiažnosť | Stav |
|---|---|---|---|---|
| Pripraviť web stránku tímu | Szilva | 3.10.2012 | | Vytvorená |
| Pripraviť šablónu zápisnice | Lekeň | 3.10.2012 | | Vytvorená |

## *5.2 Zápisnica zo stretnutia číslo 2*

**Téma stretnutia:**
**Dátum:** 10.10.2012
**Čas:** 14:00 – 16:20
**Miesto:** Softvérové laboratórium

**Vedúci stretnutia:** Bc. Filip Staňo
**Zapisovateľ:** Bc. Tomáš Lekeň

**Zúčastnení:** Bc. Michal Biroš
Bc. Tomáš Caban
Bc. Tomáš Kunka
Bc. Filip Staňo
Bc. Tomáš Lekeň
Bc. Milan Martinkovič
Bc. Bálint Szilva

**Zhodnotenie úloh z predchádzajúceho stretnutia:**

| Úloha | Riešiteľ | Dátum vzniku | Obtiažnosť | Stav |
|---|---|---|---|---|
| | | | | |

**Priebeh stretnutia:**

- Dohodli sme sa na používaní manažovacieho nástroja Redmine
- Ing. Martin Labaj nám predviedol existujúce riešenie loggera
- Diskutovali sme etapy a možné riešenia projektu
- Diskutovali sme user story, identifikovali sme nasledovné:
  - Používateľ je sledovaný pri práci na počítači
  - Používateľ je modelovaný
  - Modely sú rozpoznávané
  - Používateľ dostane odporúčanie
- Rozdelili sme si úlohy na ďalšie stretnutia

**Úlohy:**

| Úloha | Riešiteľ | Dátum vzniku | Obtiažnosť | Stav |
|---|---|---|---|---|
| Nainštalovať logger | Biroš | 10.10.2012 | | Vytvorená |
| Analýza funkcionality loggera | Biroš | 10.10.2012 | | Vytvorená |
| Vytvoriť projekt v Redmine | Szilva | 10.10.2012 | | Vytvorená |
| Analyzovať riešenia na sledovanie emócií | Staňo | 10.10.2012 | | Vytvorená |
| Nahodiť web | Szilva | 10.10.2012 | | Vytvorená |
| Analyzovať sledovanie pohľadu používateľa a natočenia tváre | Martinkovič | 10.10.2012 | | Vytvorená |
| Úprava plagátu | Biroš | 10.10.2012 | | Vytvorená |

| | | | | |
|---|---|---|---|---|
| Analyzovať sledovanie zvuku cez mikrofón | Caban | 10.10.2012 | | Vytvorená |
| Analyzovať možnosti strojového učenia | Lekeň | 10.10.2012 | | Vytvorená |
| Analyzovať odporúčania pre jednotlivé emočné stavy | Kunka | 10.10.2012 | | Vytvorená |

## *5.3 Zápisnica zo stretnutia číslo 3*

**Téma stretnutia:** Kontrola priebehu prvého šprintu
**Dátum:** 17.10.2012
**Čas:** 14:00
**Miesto:** Softvérové štúdio

**Vedúci stretnutia:** Bc. Tomáš Lekeň
**Zapisovateľ:** Bc. Bálint Szilva

**Zúčastnení:** Bc. Filip Staňo
Bc. Milan Martinkovič
Bc. Michal Biroš
Bc. Tomáš Kunka
Bc. Tomáš Caban
Bc. Tomáš Lekeň
Bc. Bálint Szilva

**Zhodnotenie úloh z predchádzajúceho stretnutia:**

| Úloha | Riešiteľ | Dátum vzniku | Obtiažnosť | Stav |
|---|---|---|---|---|
| Nainštalovať logger | Biroš | 10.10.2012 | | Rozpracované |
| Vytvoriť projekt v Redmine | Szilva | 10.10.2012 | | Rozpracované |
| Analyzovať riešenia na sledovanie emócií | Staňo | 10.10.2012 | | Ukončené |
| Vytvoriť a nahodiť web | Szilva | 10.10.2012 | | Rozpracované |
| Analyzovať sledovanie pohľadu používateľa a natočenia tváre | Martinkovič | 10.10.2012 | | Rozpracované |
| Úprava plagátu | Biroš | 10.10.2012 | | Ukončené |
| Analyzovať sledovanie zvuku cez mikrofón | Caban | 10.10.2012 | | Rozpracované |
| Analyzovať možnosti strojového učenia | Lekeň | 10.10.2012 | | Rozpracované |
| Analyzovať odporúčania pre jednotlivé emočné stavy | Kunka | 10.10.2012 | | Rozpracované |

**Priebeh stretnutia:**

- Dohodli sme sa o tom, že tím bude používať Windows Server 2008, SQL Server 2008 a Team Foundation Server
- Technická špecifikácia
  - vývoj projektu bude prebiehať v jazyku C#
  - definovanie minimálnych hardvérových požiadaviek na server (4 GB RAM, aspoň 50 GB diskového priestoru)
- Prezentácia jednotlivých úloh:
  - Filip Staňo odprezentoval svoju úlohu o riešení na sledovania emócií

  - Milan Martinkovič analyzoval možnosti na sledovanie pohľadu používateľa a natočenia tváre
  - Michal Biroš vytvoril novú upravenú verziu plagátu
  - Tomáš Caban odprezentoval možnosti na sledovanie a rozpoznanie zvuku cez mikrofón
  - Tomáš Lekeň analyzoval možnosti strojového učenia
  - Tomáš Kunka odprezentoval svoju úlohu a analyzoval možné typy odporúčaní
- Vyhodnotenie úloh
- Na základe prezentácií vytvorený Backlog projektu
- Rozdelenie nových úloh

**Úlohy:**

| **Úloha** | **Riešiteľ** | **Dátum vzniku** | **Obtiažnosť** | **Stav** |
|---|---|---|---|---|
| Analyzovať sledovanie pohľadu používateľa a natočenia tváre | Martinkovič | 10.10.2012 | | |
| Analyzovať sledovanie zvuku cez mikrofón | Caban | 10.10.2012 | | |
| Analyzovať možnosti strojového učenia | Lekeň | 10.10.2012 | | |
| Vyriešiť virtuálny stroj a nainštalovať Windows Server 2012 | Szilva | 17.10.2012 | | |
| Vytvoriť projekt v Redmine | Szilva | 10.10.2012 | | |
| Analyzovať existujúce riešenie - Logger | Biroš | 17.10.2012 | | |
| Analyzovať odporúčania pre jednotlivé emočné stavy | Kunka | 10.10.2012 | | |
| Analyzovať možné licencie FaceAPI | Staňo | 17.10.2012 | | |

## *5.4 Zápisnica zo stretnutia číslo 4*

**Téma stretnutia:** Kontrola priebehu prvého šprintu
**Dátum:** 24.10.2012
**Čas:** 14:00 – 16:40
**Miesto:** Softvérové štúdio

**Vedúci stretnutia:** Bc. Bálint Szilva
**Zapisovateľ:** Bc. Filip Staňo

**Zúčastnení:** Bc. Michal Biroš
Bc. Tomáš Caban
Bc. Tomáš Kunka
Bc. Filip Staňo
Bc. Tomáš Lekeň
Bc. Milan Martinkovič
Bc. Bálint Szilva

**Zhodnotenie úloh z predchádzajúceho stretnutia:**

| Úloha | Riešiteľ | Dátum vzniku | Obtiažnosť | Stav |
|---|---|---|---|---|
| Nainštalovať logger | Biroš | 10.10.2012 | | Ukončené |
| Vytvoriť projekt v Redmine | Szilva | 10.10.2012 | | Ukončené |
| Analyzovať riešenia na sledovanie emócií | Staňo | 10.10.2012 | | Ukončené |
| Vytvoriť a nahodiť web | Szilva | 10.10.2012 | | Ukončené |
| Analyzovať sledovanie pohľadu používateľa a natočenia tváre | Martinkovič | 10.10.2012 | | Ukončené |
| Úprava plagátu | Biroš | 10.10.2012 | | Ukončené |
| Analyzovať sledovanie zvuku cez mikrofón | Caban | 10.10.2012 | | Ukončené |
| Analyzovať možnosti strojového učenia | Lekeň | 10.10.2012 | | Ukončené |
| Analyzovať odporúčania pre jednotlivé emočné stavy | Kunka | 10.10.2012 | | Ukončené |

**Priebeh stretnutia:**

- Milan Martinkovič odprezentoval analýzu sledovania pohľadu, nebolo nájdené vhodné riešenie pre ďalšie použitie. Z analýzy vyplynulo, že kamery integrované v notebookoch nie sú príliš vhodné na snímanie pohľadu. Riešenia nie sú užívateľsky príjemné. Zatiaľ sme nenašli vhodné riešenie pre bežné používateľské podmienky, túto funkcionalitu zatiaľ odkladáme.
- Tomáš Caban doplnil analýzu sledovania pomocou mikrofónu a modelovania nálady na základe zvuku. Venoval sa sledovaniu prítomnosti používateľa pomocou Sonaru. Z analýzy vyplynulo, že sledovanie prítomnosti na princípe Sonaru je veľmi závislé od HW notebooku. Program neposkytuje výstup, k dispozícii je ale zdrojový kód.

Z hľadiska rozpoznávania používateľa podľa hlasu nebolo nájdené vhodné riešenie. Riešenia EmoVoice a OpenEar sú použiteľné.

- Tomáš Lekeň odprezentoval analýzu strojového učenia. Priblížil existujúce alternatívy strojového učenia. Prioritne bude používané offline strojové učenie, v prípade potreby sa zavedie aj online. Na základe analýzy sme získali prehľad v prístupoch ku strojovému učeniu.
- Prebrali sme stav servera. Bude potrebné riešiť nesplnenie požiadaviek na Harddisk a RAM.
- Filip Staňo prezentoval analýzu detekovateľných emočných stavov a analýzu existujúcich riešení na sledovanie tváre. Nadobudli sme znalosti o jednotlivých emočných stavoch a o tom, čo je pre ne typické. Rozobrali sme aplikácie FaceAPI, Face.com a Human Emotion Detection. Primárne chceme používať FaceAPI, ale čakáme na licenciu od spoločnosti SeeingMachines. Čiastočne použiteľné sú aj zvyšné dve riešenia, v prípade neudelenia licencie zvážime ďalší postup.
- Michal Biroš prezentoval analýzu funkcionality Loggera PerConik. Nadobudli sme znalosti o funkcionalite loggera. Rozoberali sme alternatívy doplnenia našich dát do systému.
- Následne sme sa venovali identifikácií nových úloh a ohodnoteniu ich náročnosti.

**Úlohy:**

| **Úloha** | **Riešiteľ** | **Dátum vzniku** | **Obtiažnosť** | **Stav** |
|---|---|---|---|---|
| Získavanie obrazu z kamery | Martinkovič | 24.10.2012 | 8 | Pridelené |
| Spracovanie obrazu pomocou študentského projektu HED (Bang) | Staňo | 24.10.2012 | 20 | Pridelené |
| Spracovanie obrazu pomocou študentského projektu HED (Bang) | Kunka | 24.10.2012 | 20 | Pridelené |
| Spracovanie obrazu pomocou face.com | Szilva | 24.10.2012 | 20 | Pridelené |
| Spracovanie obrazu pomocou face.com | Biroš | 24.10.2012 | 20 | Pridelené |
| Získavanie a spracovanie zvuku | Lekeň | 24.10.2012 | 20 | Pridelené |
| Získavanie a spracovanie zvuku | Caban | 24.10.2012 | 20 | Pridelené |
| Analyzovať možnosti licencie FaceAPI | Staňo | 17.10.2012 | | Rozpracované |
| Nainštalovať server + SQL + TFS | Szilva | 24.10.2012 | 5 | Rozpracované |

## *5.5 Zápisnica zo stretnutia číslo 5*

**Téma stretnutia:** Zhodnotenie prvého šprintu, plánovanie druhého.
**Dátum:** 31.10.2012
**Čas:** 14:00 – 16:40
**Miesto:** Softvérové štúdio

**Vedúci stretnutia:** Bc. Filip Staňo
**Zapisovateľ:** Bc. Tomáš Kunka

**Zúčastnení:** Bc. Michal Biroš
Bc. Tomáš Caban
Bc. Tomáš Kunka
Bc. Filip Staňo
Bc. Tomáš Lekeň
Bc. Milan Martinkovič
Bc. Bálint Szilva

**Zhodnotenie úloh z predchádzajúceho stretnutia:**

| Úloha | Riešiteľ | Dátum vzniku | Obtiažnosť | Stav |
|---|---|---|---|---|
| Získavanie obrazu z kamery | Martinkovič | 24.10.2012 | 8 | Splnené na 4 |
| Spracovanie obrazu pomocou študentského projektu HED (Bang) | Staňo | 24.10.2012 | 20 | Splnené na 10 |
| Spracovanie obrazu pomocou študentského projektu HED (Bang) | Kunka | 24.10.2012 | 20 | Splnené na 10 |
| Spracovanie obrazu pomocou face.com | Szilva | 24.10.2012 | 20 | Splnené – nedá sa použiť |
| Spracovanie obrazu pomocou face.com | Biroš | 24.10.2012 | 20 | Splnené – nedá sa použiť |
| Získavanie a spracovanie zvuku | Lekeň | 24.10.2012 | 20 | Pridelené |
| Získavanie a spracovanie zvuku | Caban | 24.10.2012 | 20 | Pridelené |
| Analyzovať možnosti licencie FaceAPI | Staňo | 17.10.2012 | | Splnené |
| Nainštalovať server + SQL + TFS | Szilva | 24.10.2012 | 5 | Splnené na 4 |

**Priebeh stretnutia:**

- Na začiatku stretnutia sme ohodnotili splnenie jednotlivých úloh z predchádzajúcich šprintov. Spätne sme ohodnotili úlohy z prvého šprintu.
- Zhodnotili sme, že face API, nebude možné použiť, kvôli licencii. Preberali sme ďalšie riešenia na zachytávanie ľudskej tváre. Rozhodli sme sa pre projekt HED.

- Milan Martinkovič predviedol získavanie obrazu z kamery. Jeho ďalšou úlohou bude implementovať jeho riešenie do projektu PerConIK.
- Tomáš Caban zhodnotil aký pokrok bol urobený v zachytávaní zvuku. Vyskytol sa problém s knižnicou. Rozhodli sme sa spracovanie zvuku vrátiť do Product Backlogu.
- Vytvorili sme nové úlohy a zároveň sme ich ohodnotili.

**Úlohy:**

| **Úloha** | **Riešiteľ** | **Dátum vzniku** | **Obtiažnosť** | **Stav** |
|---|---|---|---|---|
| 1. Konfigurácia TFS a Webu | Szilva | 31.10.2012 | 5 | Vytvorená |
| 2. Tvorba dokumentácie | Leken | 31.10.2012 | 40 | Vytvorená |
| 3. Získať obraz z kamery | Martinkovič | 31.10.2012 | 20(4) | Prenesená |
| 4. Spracovať obraz pomocou študentského projektu HED | Staňo, Kunka, Szilva | 31.10.2012 | 60(10) | Prenesená |
| 5. Skompilovanie programu na spracovanie zvuku | Caban | 31.10.2012 | 8 | Vytvorená |
| 6. Inštalácia služby do IIS | Biroš | 31.10.2012 | 5 | Vytvorená |
| 7. Reprezentácia údajov z kamery | Lekeň | 31.10.2012 | 8 | Vytvorená |
| 8. Príprava údajov na odosielanie | Caban | 31.10.2012 | 3 | Vytvorená |
| 9. Preposielanie dát | Biroš | 31.10.2012 | 13 | Vytvorená |

## *5.6 Zápisnica zo stretnutia číslo 6*

**Téma stretnutia:** Kontrola priebehu druhého šprintu
**Dátum:** 7.11.2012
**Čas:** 15:00 – 17:40
**Miesto:** Softvérové štúdio

**Vedúci stretnutia:** Bc. Tomáš Kunka
**Zapisovateľ:** Bc. Milan Martinkovič

**Zúčastnení:** Bc. Michal Biroš
Bc. Tomáš Caban
Bc. Tomáš Kunka
Bc. Filip Staňo
Bc. Tomáš Lekeň
Bc. Milan Martinkovič
Bc. Bálint Szilva

**Zhodnotenie úloh identifikovaných z predchádzajúceho stretnutia:**

| Úloha | Riešiteľ | Dátum vzniku | Obtiažnosť | Stav |
|---|---|---|---|---|
| 1. Konfigurácia TFS a Webu | Szilva | 31.10.2012 | 5 | Splnená |
| 2. Tvorba dokumentácie | Leken | 31.10.2012 | 40 | Rozpracovaná |
| 3. Získať obraz z kamery | Martinkovič | 31.10.2012 | 20 | Rozpracovaná |
| 4. Spracovať obraz pomocou študentského projektu HED | Staňo, Kunka, Szilva | 31.10.2012 | 60 | Rozpracovaná |
| 5. Skompilovanie programu na spracovanie zvuku | Caban | 31.10.2012 | 8 | Rozpracovaná |
| 6. Inštalácia služby do IIS | Biroš | 31.10.2012 | 5 | Rozpracovaná |
| 7. Reprezentácia údajov z kamery | Lekeň | 31.10.2012 | 8 | Rozpracovaná |
| 8. Príprava údajov na odosielanie | Caban | 31.10.2012 | 3 | Rozpracovaná |
| 9. Preposielanie dát | Biroš | 31.10.2012 | 13 | Rozpracovaná |

**Priebeh stretnutia:**

- Tomáš Lekeň nám v súvislosti so svojou úlohou, ktorou je tvorba dokumentácie, priblížil náležitosti, ktoré bude treba obsiahnuť v projektovej dokumentácií. Odovzdať ju je potrebné do 14.11.2012. Dohodli sme sa, že každý spracuje svoju časť analýzy, na ktorej pracoval v prvom šprinte. Manažér plánovania (Milan Martinkovič) pripraví formálnu podobu plánu pre použitie v dokumentácií. Michal Biroš zostaví dokumentáciu k product backlog-u. Ďalej každý člen tímu pár vetami popíše podstatu svojej úlohy, ktorú v tíme napĺňa.
- Ďalej sme zdôraznili fakt, že je potrebné dôslednejšie uvádzanie podrobností k jednotlivým úlohám v nástroji Redmine.
- Nasledovala prezentácia o pokroku svojej úlohy, zo strany Milana Martinkoviča. Zatiaľ sa mu podarilo odstrániť chybové hlásenia, pri pokuse o použitie zostaveného riešenia pre zachytávanie obrazu mimo vzorového projektu knižnice, ktorú na túto úlohu využíva. Momentálne je jeho riešenie pripravené na nasadenie do PerConik-a.
- Tomáš Caban zosumarizoval svoju úlohu, ktorej cieľom je kompilácia programu na spracovaniu zvuku. Problémy sa vyskytli pri samotnej kompilácií a kvôli nekonzistentnej dokumentácií sa mu tento problém nepodarilo odstrániť. Podarilo sa mu nájsť projekt OpenSmile, ktorý sa na základe zbežnej analýzy zdá byť použiteľný.
- Michal Biroš sa stretol s problémom pri plnení svojej úlohy, ktorou je nasadenie PerConik-a na IIS server. Príčinou problému je nízka miera skúseností s IIS.
- Bálint Szilva mal za úlohu nainštalovať Team Foundation Server, čo sa mu aj podarilo. V čase stretnutia sa vyskytol problém s prístupom cez URL, ktorá bola vygenerovaná pri inštalácií.
- Ďalej sme sa snažili dosiahnuť úspešnú kompiláciu zdrojových kódov PerConik. Po nalinkovaní, niektorých knižníc, ktoré neboli priamo prítomné v riešení, ktoré nám bolo zaslané sa nám podarilo program spojazdniť.
- Následne na stretnutie prišiel Mate Fejes, ktorý rieši diplomovú prácu s podobnou témou akú máme my. Ako odrazový bod nám odporučil riešenie s názvom Luxand, ktoré dokáže analýzou obrazu vrátiť súradnice kľúčových bodov na tvári. Zdôraznil dôležitosť rozhodnutia sa, že v akej forme budeme uvádzať emócie a taktiež pripomenul, že dôležitým faktorom bude trénovanie na rôznych datasetoch. Spomenul, že sa môžeme použiť aj FACS (Facial Coding System). Táto cesta si ale podľa neho vyžaduje pomerne sofistikovaný nástroj na získavanie súradníc. Ďalej nám ukázal, čo má už rozpracované. Ukázal napríklad SVM lib, čo je knižnica na strojové učenie sa, ktorú nám aj odporúčil. Stretnutie sme uzavreli diskusiou o nástroji Luxand. Dozvedeli sme sa, že sa dá vybaviť skúšobná licencia, ktorá je zadarmo.

# 6 Metodiky použité pri vývoji

## *6.1 Zber požiadaviek*

### 6.1.1 Roly

| Rola | Zodpovednosti | Procesy |
|---|---|---|
| **Zákazník** | • Určuje prvotné požiadavky<br>• Stará sa o konzultácie s vývojovým tímom<br>• Stanovuje prioritu jednotlivých úloh | • Stanovanie požiadaviek<br>• Prioritizácia úloh<br>• Konzultácia o zmene |
| **Manažér tímu** | • Priraďuje členom tímu úlohy<br>• Vedie komunikáciu so zákazníkom<br>• Vyberá tím na jednotlivé úlohy<br>• Predáva vypracované úlohy v tíme | • Stanovanie požiadaviek<br>• Spracovanie požiadaviek<br>• Konzultácia o zmene<br>• Dekompozícia požiadaviek<br>• Schválenie product backlogu |
| **Analytik** | • Vykonáva analýzu požiadaviek<br>• Pripravuje alternatívy v prípade nerealizovateľnosti požiadavky | • Spracovanie požiadaviek<br>• Dekompozícia požiadaviek<br>• Konzultácia o zmene<br>• Schválenie product backlogu |
| **Návrhár** | • Pracuje pri dekompozícii používateľských príbehov na features a následne na úlohy | • Dekompozícia požiadaviek<br>• Schválenie product backlogu |

### 6.1.2 Procesy zberu požiadaviek

#### 6.1.2.1 Proces stanovenia požiadaviek

**Vstup:** požiadavka na vytvorenie softvéru
**Výstup:** stanovené a zaznamenané požiadavky zákazníka
Vykonanie prvotnej komunikácie so zákazníkom. Jedná sa o získanie prvotnej predstavy o požiadavkách od zákazníka.

| # | Názov kroku |
|---|---|
| **1.** | Stanovenie podmienok stretnutia |
| **2.** | Zostavenie tímu na stretnutie |
| **3.** | Realizácia stretnutia |

#### 6.1.2.2 Proces spracovania požiadaviek

**Vstup:** stanovené a zaznamenané požiadavky zákazníka
**Výstup:** rozhodnutie o realizovateľnosti požiadaviek
Po získaní požiadaviek od zákazníka je nutné zistiť realizovateľnosť daných požiadaviek. Tomuto sa bude venovať analytický tím, pričom riadenie procesu má na starosti manažér projektu.

| # | Názov kroku |
|---|---|
| **1.** | Odovzdanie požiadaviek na analýzu |
| **2.** | Analýza požiadaviek |
| **3.** | Oboznámenie manažéra projektu o výsledku analýzy |

### 6.1.2.3 Proces konzultácie o zmene požiadavky

**Vstup:** nutnosť úpravy požiadavky
**Výstup:** výber alternatívy vysporiadania sa s problémom
Nastáva v prípade, keď sa po analýze zistí, že daný proces nie je realizovateľný. Na základe toho je nutné skonzultovať požiadavku so zákazníkom.

| # | Názov kroku |
|---|---|
| **1.** | Kontaktovanie zákazníka a žiadosť o konzultáciu |
| **2.** | Príprava možných alternatív riešenia problému |
| **3.** | Konzultácia so zákazníkom |

### 6.1.2.4 Proces dekompozície požiadaviek

**Vstup:** schválené požiadavky
**Výstup:** namodelovaný systém
Dochádza k modelovaniu systému z hľadiska používateľa a postupnej dekompozícii potrebnej k dosiahnutiu výsledku. Na záver je potrebné úlohy ohodnotiť a predať na ďalšie spracovanie.

| # | Názov kroku |
|---|---|
| **1.** | Tvorba používateľských príbehov |
| **2.** | Dekompozície používateľských príbehov na features |
| **3.** | Dekompozície features na úlohy |
| **4.** | Ohodnocovanie úloh |

### 6.1.2.5 Proces prioritizácie úloh

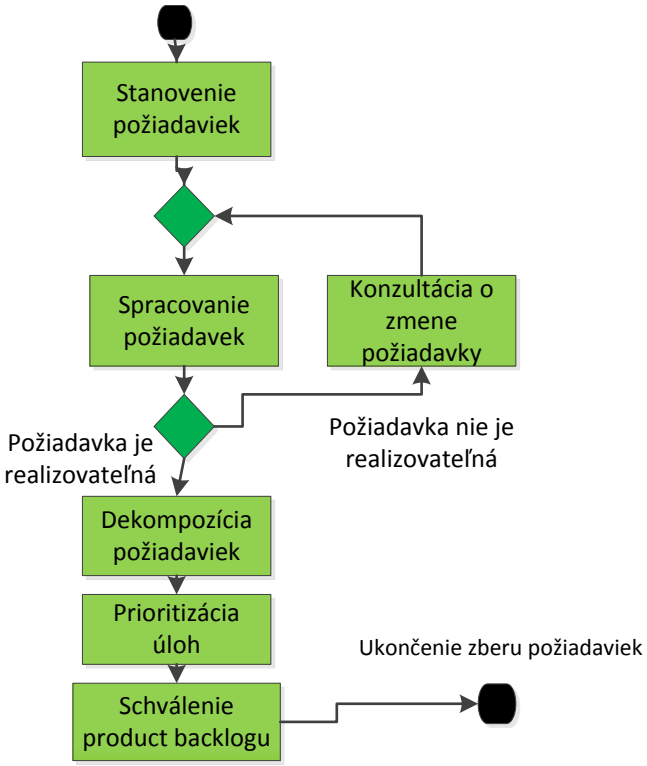
**Vstup:** úlohy v man. systéme
**Výstup:** určená priorita úloh
V tomto procese zákazník v manažérskom systéme priraďuje úlohám prioritu vykonania.

| # | Názov kroku |
|---|---|
| **1.** | Zobrazenie úloh zákazníkom v systéme |
| **2.** | Nastavenie priority úloh |

### 6.1.2.6 Proces schválenia product backlogu

**Vstup:** úlohy po určení priority
**Výstup:** predanie namodelovaného systému na implementáciu
V tomto procese sa vykoná definitív ne schválenie product backlogu celým vývojovým tímom.

| # | Názov kroku |
|---|---|
| **1.** | Stretnutie tímu |
| **2.** | Revízia úloh |
| **3.** | Predanie úloh na rozdelenie |




**Obrázok 5** Diagram činností procesu zberu požiadaviek

## 6.1.3 Vytvorenie novej zákazníckej požiadavky v systéme Redmine

Táto metodika slúži ako popis presného postupu na vytvorenie požiadavky zadanej zákazníkom do systému Redmine. Tento spôsob sa využíva v prípade, keď zákazník potrebuje doplniť požiadavku k už existujúcemu zoznamu požiadaviek, tz. už bolo vykonané prvotné spracovanie požiadaviek.
Kroky:

1. Predpokladom je byť na stránke projektu v systéme Redmine. Následne zvoliť položku „**New Issue**“.
2. Nasleduje **nastavenie detailov** pridávanej požiadavky.

3. Položku **Tracker** nastaviť na hodnotu „Feature“.
4. Do položky **Subject** vložiť stručný názov novej požiadavky.
5. **Status** nastaviť na „New“ – jedná sa o novú požiadavku, ktorú následne spracuje a pridelí vývojový tím.
6. V prípade priority sú k dispozícií možnosti:
    a. **Low (Nízka)** – vybrať v prípade, ak zadávaná požiadavka nehrá významnú rolu v systéme a je možné ju doplniť až ku koncu projektu.
    b. **Normal (Bežná)** – vybrať v prípade, ak ide o požiadavku, ktorá nijako významom neprevyšuje dovtedy vytvorené požiadavky a teda nie je potrebné sa jej v blízkom období venovať.
    c. **High (Vysoká)** – vybrať v prípade, ak je požiadavku nutné vykonať v blízkej dobe, ale nie okamžite, teda existujú úlohy s vyššou prioritou
    d. **Urgent (Urgentná)** – vybrať v prípade, ak si požiadavka vyžaduje okamžitú pozornosť, ale zároveň od nej nie je závislý už priebehajúci vývoj softvéru.
    e. **Immediate (Okamžitá)** – vybrať v prípade, ak si požiadavka vyžaduje okamžitú pozornosť a jej vytvorenie výrazne ovplyvňuje už prebiehajúci vývoj softvéru – používa sa aby sme predišli vývoju zbytočných častí softvéru, ktoré by zadaná požiadavka mohla ovplyvniť a vyradiť.
7. **Konkretizáciu požiadavky** je nutné priložiť ako prílohu prostredníctvom tlačidla „Vybrať súbor“, kde sa požaduje nahratie súboru z PC zákazníka. Obmedzenie veľkosti súboru je 24,5 MB.
8. Nasleduje **potvrdenie požiadavky** tlačidlom „Create“, ktoré sa nachádza v ľavom dolnom rohu aplikácie.

## *6.2 Manažment iterácií*

Podstatnú časť iterácie tvorby softvérového produktu tvorí plánovanie šprintu, ktoré prebieha na špeciálnom stretnutí na to určenom. Spravidla sa toto stretnutie uskutočňuje na začiatku šprintu. Stretnutie sa skladá z dvoch hlavných častí. Na prvej časti plánovania sa zúčastňuje vlastník produktu, *scrum master* a celý vývojársky tím. V druhej časti sú prítomní *scrum master* a príslušný vývojársky tím, zatiaľ čo vlastník produktu nie je prítomný v miestnosti ale stále je k dispozícii aby mohol zodpovedať prípadné otázky. Potom, ako sú naplánované príslušné náležitosti šprintu, pokračuje iterácia napĺňaním a riešením týchto náležitostí a v závere sa uskutočnená práca analyzuje za účelom zdokonalenia nadchádzajúcej iterácie.

### 6.2.1 Proces aktualizácie a prezentácie produktových požiadaviek

| | |
|---|---|
| **Popis:** | Doplnenie súčastí produktových požiadaviek na základe požiadaviek zákazníkov a následná prezentácia týchto súčastí. |
| **Vykonáva:** | Vlastník produktu, *Scrum master* |
| **Vstup:** | Požiadavky zákazníkov |
| **Výstup:** | Prioritne usporiadané a odprezentované produktové požiadavky |
| **Postupnosť krokov:** | |
| **1.** | Vlastník produktu získa požiadavky od zákazníkov. |
| **2.** | Aktualizovanie produktových požiadaviek. |
| **3.** | Prezentácia produktových požiadaviek. |

#### 6.2.1.1 Vlastník produktu získa požiadavky od zákazníkov.

- Popis: Zákazníci adresujú svoje požiadavky na vlastníka produktu, ten tieto požiadavky zosumarizuje a následne sformuluje v podobe používateľských príbehov.
- Vstup: Požiadavky zákazníkov
- Výstup: Identifikované používateľské príbehy

#### 6.2.1.2 Aktualizovanie produktových požiadaviek.

- Popis: Vlastník produktu určí priority jednotlivých používateľských príbehov. Následne sa tieto používateľské príbehy usporiadajú v momentálnych produktových požiadavkách na základe ich prislúchajúcej priority.
- Vstup: Používateľské príbehy
- Výstup : Usporiadané produktové požiadavky

#### 6.2.1.3 Prezentácia produktových požiadaviek.

- Popis: Vlastník produktu (alebo v prípade neprítomnosti *Scrum master*) prezentuje produktové požiadavky vývojárskemu tímu v prioritnom poradí. Nasleduje

ozrejmenie akceptačných kritérií za účelom objasnenia významu jednotlivých používateľských príbehov a doladia sa priority na základe vzniknutých pripomienok.

- Vstup: Prezentácia produktových požiadaviek
- Výstup: Finálna podoba produktových požiadaviek v rámci aktuálneho šprintu

### 6.2.2 Proces určenia cieľu šprintu a výberu používateľských príbehov

| | |
|---|---|
| **Popis:** | Vytýčenie hlavného cieľu šprintu a selekcia používateľských príbehov, ktoré sa budú preberať na plánovacom stretnutí. |
| **Vykonáva:** | Vlastník produktu, Vývojársky tím |
| **Vstup:** | Očakávania vlastníka produktu |
| **Výstup:** | Vybraná podmnožina produktových požiadaviek |
| **Postupnosť krokov:** | |
| **1.** | Vlastník produktu uvedie očakávania. |
| **2.** | Výber používateľských príbehov. |
| **3.** | Odchod vlastníka produktu. |

#### 6.2.2.1 Vlastník produktu uvedie očakávania.

- Popis: Vlastník produktu uvedie svoje očakávania od nadchádzajúceho šprintu a formuluje ich v podobe cieľov šprintu.
- Vstup: Očakávania vlastníka produktu
- Výstup: Ciele/téma šprintu

#### 6.2.2.2 Výber používateľských príbehov.

- Popis: Formou dialógu medzi vlastníkom produktu a tímom vývojárov sa určia používateľské príbehy, ktoré budú predmetom aktuálneho stretnutia. Vyberie sa väčšie množstvo používateľských príbehov, aké by bolo možné zvládnuť vyriešiť v nadchádzajúcom šprinte, nakoľko sa bude tento zoznam ešte selektovať.
- Vstup: Ciele/téma šprintu
- Výstup : Vybraná podmnožina produktových požiadaviek

#### 6.2.2.3 Odchod produktového vlastníka.

- Popis: Produktový vlastník odchádza zo stretnutia aby sa mohla otvoriť diskusia v rámci vývojárskeho tímu ale stále zostáva k dispozícii aby si ho mohli ostatní členovia tímu zavolať na zodpovedanie novo vzniknutých otázok. V takomto prípade sa vlastník produktu dostaví do miesta konania stretnutia, zodpovie otázky a následne opäť opúšťa miesto stretnutia.
- Vstup: Odchod vlastníka produktu
- Výstup: Otvorenie diskusie medzi vývojármi

### 6.2.3 Proces spresnenia dodania výsledku šprintu

| | |
|---|---|
| **Popis:** | Tím sa dohodne na súčastiach, ktoré spracujú v plánovanom šprinte, charakterizuje sa náročnosť a určia sa riešitelia jednotlivých úloh. |
| **Vykonáva:** | Vývojársky tím |
| **Vstup:** | Vybraná podmnožina produktových požiadaviek |
| **Výstup:** | Úlohy s identifikovanou náročnosťou a riešiteľmi |
| **Postupnosť krokov:** | |
| **1.** | Zhodnotenie prezentácie a požiadaviek vlastníka produktu. |
| **2.** | Výber používateľských príbehov pre šprint. |
| **3.** | Identifikácia podúloh. |
| **4.** | Pridelenie riešiteľov. |

#### 6.2.3.1 Zhodnotenie prezentácie a požiadaviek vlastníka produktu.

- Popis: Ako náhle odíde vlastník produktu, otvorí sa diskusia v rámci vývojárskeho tímu za účelom zhodnotenia doterajšej prezentácie. Hlavnými témami sú používateľské príbehy s vysokou prioritou so zvážením stanovených cieľov. Pred uzavretím diskusie je dôležité zosumarizovať sprievodné udalosti šprintu. Jedná sa o plánované prerušenia ako napríklad dovolenka poprípade dôležité míľniky, ktoré by mohli narušiť plynulý chod šprintu.
- Vstup: Vybraná podmnožina produktových požiadaviek
- Výstup: Diskusia o podstatných častiach požiadaviek

#### 6.2.3.2 Výber používateľských príbehov pre šprint.

- Popis: Na základe sumarizácie doterajšieho stretnutia dochádza k výberu používateľských príbehov, ktoré budú spracované v nadchádzajúcom šprinte.
- Vstup: Diskusia podstatných častí produktových požiadaviek
- Výstup : Šprintové požiadavky

#### 6.2.3.3 Identifikácia podúloh.

- Popis: Vývojársky tím rozbije jednotlivé používateľské príbehy na prislúchajúce podúlohy (*tasks*). Následne sa určí časová náročnosť jednotlivých identifikovaných podúloh. Časová náročnosť jednotlivých častí sa určí za pomoci SCRUM kariet. Pred samotným odhadovaním je vhodné priblížiť jednotlivé kroky danej činnosti aby sa zvýšilo povedomie o danej úlohe.
- Vstup: Šprintové požiadavky
- Výstup: Podúlohy s odhadovanou náročnosťou

#### 6.2.3.4 Pridelenie riešiteľov.

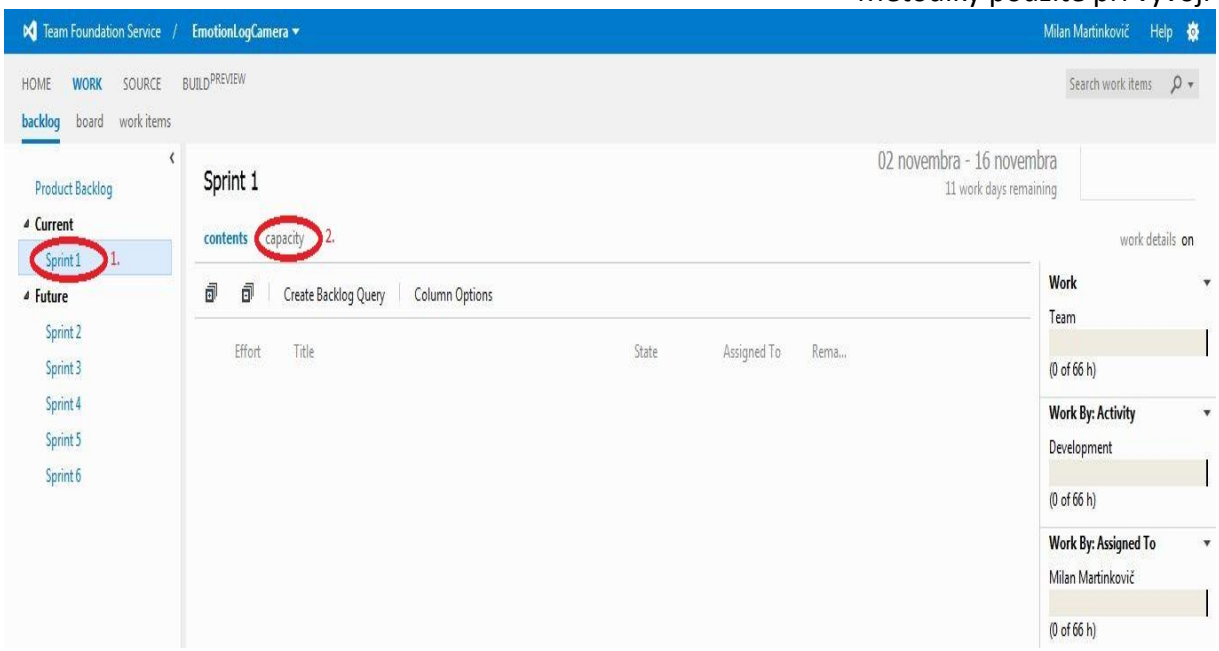
- Popis: Po určení náročnosti jednotlivých úloh sa k riešeniu týchto úloh postupne prihlasujú členovia vývojárskeho tímu. Pokiaľ je záujemcov viacej tak títo členovia tímu uvedú nápady a myšlienky v súvislosti s danou úlohou a v ideálnom prípade sa títo záujemcovia medzi sebou dohodnú na tom, komu bude úloha priradená. Pokiaľ sa nedosiahne konsenzus tak zasahuje vedúci tímu( *SCRUM Master*), ktorý rozhodne o pridelení úlohy.
- Vstup: Podúlohy s odhadovanou náročnosťou
- Výstup: Úlohy s priradenými riešiteľmi

### 6.2.4 Vytvorenie šprintových požiadaviek v MS Team Foundation Server

Kapitola popisuje spôsob pridania požiadaviek pre aktuálny šprint v systéme MS Team Foundation Server.
Identifikácií požiadaviek pre aktuálny šprint predchádza prioritné usporiadanie produktových požiadaviek. Na stretnutí, ktoré je zamerané na plánovanie šprintu sa rozhodne, ktorá položka bude predmetom aktuálneho šprintu. K tejto produktovej požiadavke sa následne jednotlivý účastníci stretnutia snažia identifikovať menšie úlohy, vďaka ktorým dokážeme danú produktovú požiadavku naplniť.

#### 6.2.4.1 Určenie pracovného nasadenia pre aktuálny šprint

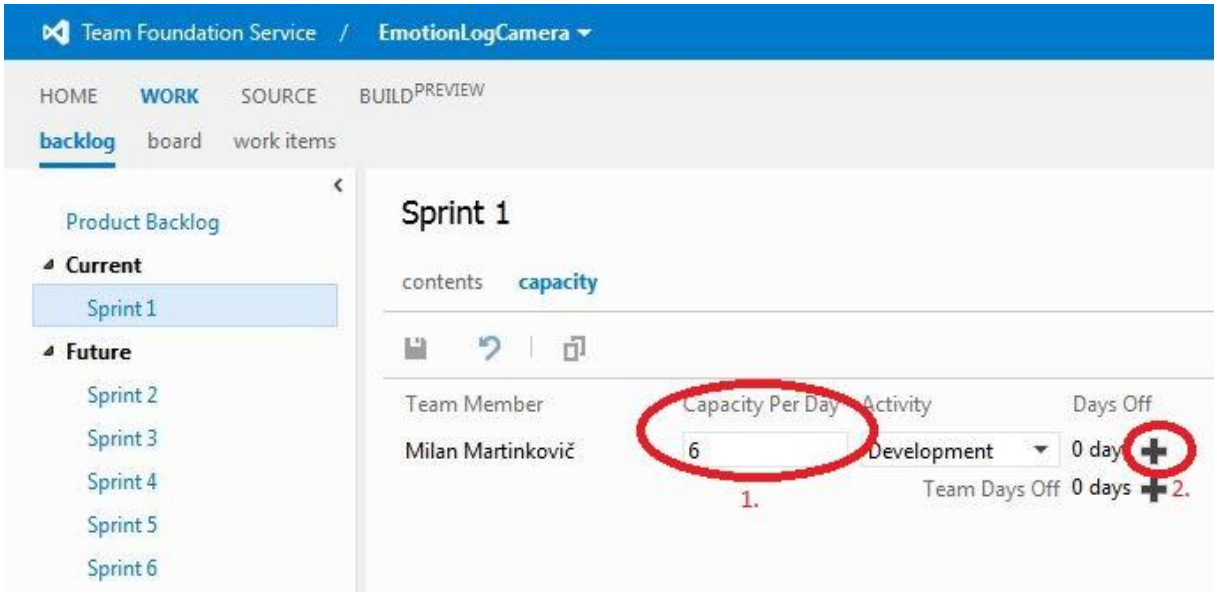
Prvou úlohou je určenie pracovného nasadenia. Jednotlivý členovia vývojárskeho tímu oznámia svoje plánované prekážky v práci ako napríklad dovolenka, rodinná oslava a podobne. Nie je potrebné uvádzať dôvod, dôležité je identifikovanie dĺžky trvania prekážok v rámci aktuálneho šprintu. Určí sa aj počet hodín, počas ktorých budú jednotliví členovia tímu schopní denne na svojej úlohe pracovať.
Nasledovný obrázok popisuje jednotlivé kroky, ktoré je potrebné vykonať pre prístup k položke zaznamenania pracovného nasadenia.




**Obrázok 6** Výber položky pre určenie pracovného nasadenia

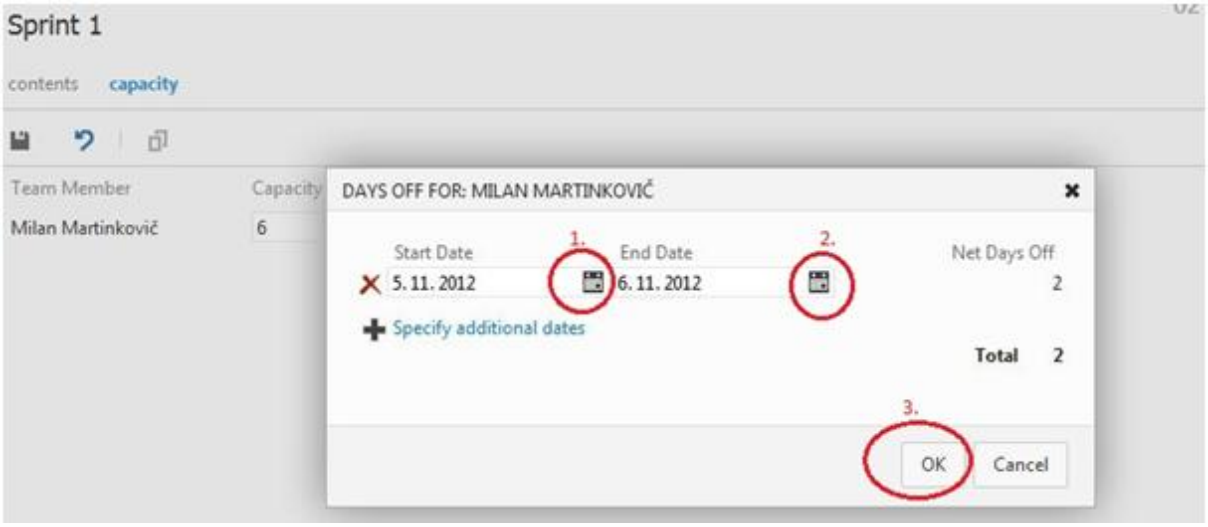
V základnom zobrazení zvolíme aktuálny šprint(1.) a následne prejdeme na položku *capacity*(2.), v ktorej určíme jednotlivé náležitosti popisujúce pracovné nasadenie člena tímu.

Na nasledovnom obrázku môžeme vidieť priradenie denného počtu hodín, počas ktorých bude člen pracovať na danej úlohe(1.) a pokiaľ člen tímu oznámil aj plánované prekážky v práci tak sa tak uvedie cez znamienko "+" v stĺpci *Days Off*(2.). Tento proces zopakujeme pre všetkých členov tímu.




**Obrázok 7** Určenie denného nasadenia a prípadne uvedenie prekážok v práci

Ak sa uvádza prekážka v práci je potrebné uviesť aj časový interval, v ktorom bude prekážka prebiehať. Pomocou ikony kalendára zvolíme začiatočný(1.) a koncový(2.) dátum trvania udalosti a následne potvrdíme pridanie prekážky v práci(3.). Tieto kroky sú zobrazené na obrázku 4.

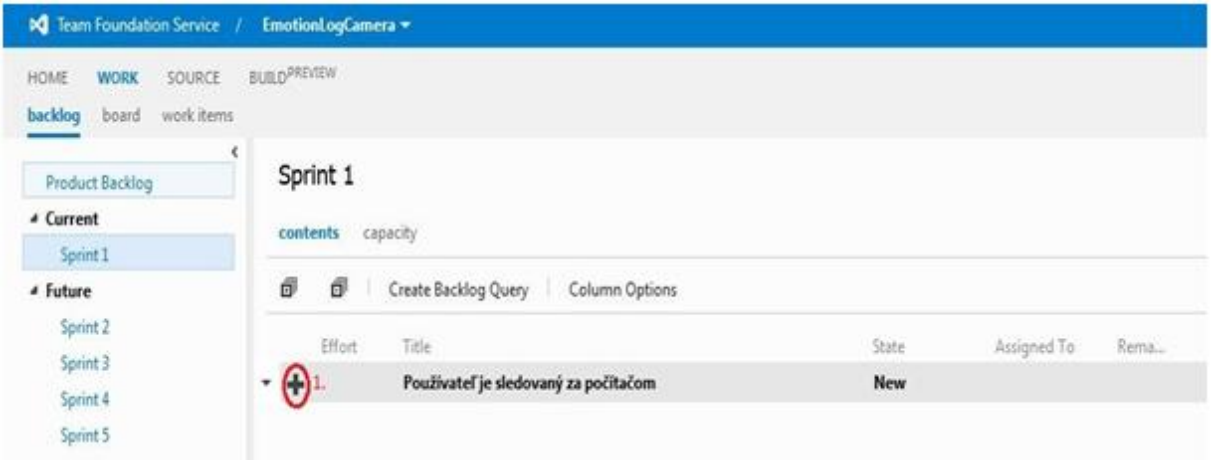



**Obrázok 8** Zaznamenanie prekážky v práci

### 6.2.4.2 Pridanie šprintových požiadaviek

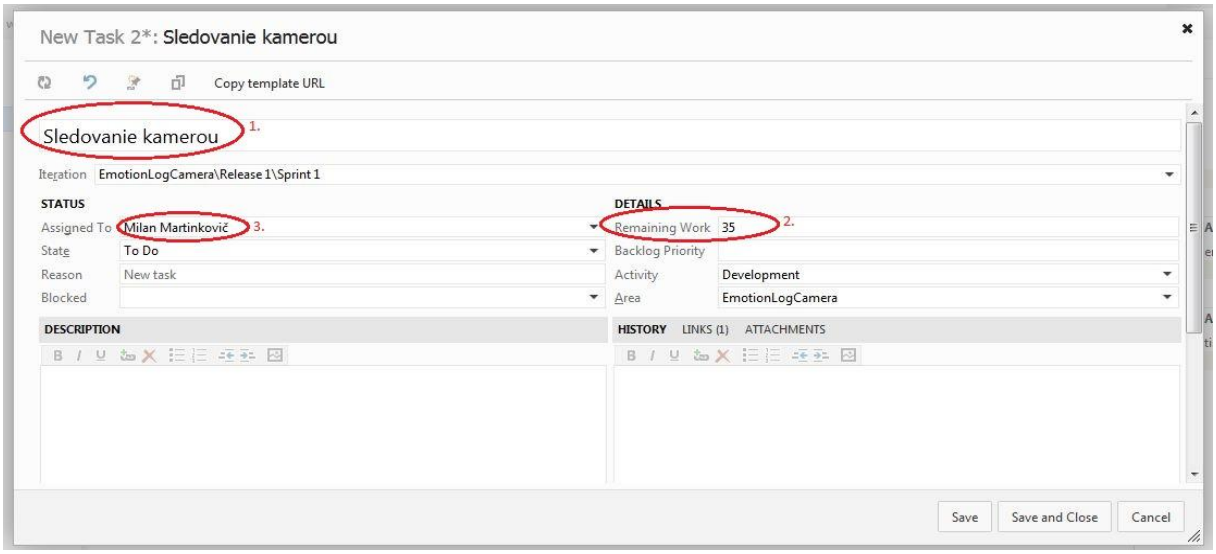
Na plánovacom stretnutí sa okrem iného preberá aj priorita jednotlivých produktových požiadaviek. Na základe priorít, ktoré si účastníci stretnutia stanovili sa do najbližšieho šprintu budú rozpracovávať položky s najväčšími prioritami.

Pre vytvorenie požiadaviek šprintu si teda najprv v nástroji vyberieme produktovú požiadavku s najvyššou prioritou a potiahneme ju na ikonu aktuálneho šprintu. Následne pridáme(1.) k tejto produktovej požiadavke, ktorú sme priradili šprintu, príslušné úlohy, ktorých vypracovaním dosiahneme naplnenie danej požiadavky.




**Obrázok 9** Zvolenie možnosti priradenia podúloh pre produktovú požiadavku, ktorú sme vložili do šprintu

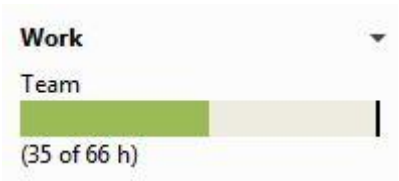
Pri pridávaní úlohy pre požiadavku treba dbať na viacero vlastností identifikovanej úlohy. Okrem názvu(1.) musí mať úloha priradenú aj náročnosť(2.). Za týmto účelom sa zhrnú všetky náležitosti prislúchajúce k danej úlohe aby sa dosiahol presnejší výsledok pri určovaní náročnosti. Následne sa pristúpi k odhadu náročnosti úlohy. Ten sa vykoná na základe SCRUM kariet, kedy jednotliví členovia tímu ukážu v jednom momente svoj odhad na kartičke aby sa predišlo ovplyvňovaniu mienky ďalších zúčastnených. Ďalej sa diskutuje o uvedených odhadoch až kým sa nedostaneme k hodnote, s ktorou budú súhlasiť všetci.
Takisto je potrebné k úlohe prideliť aj riešiteľa/riešiteľov(3.). Riešiteľ sa v rámci plánovacieho stretnutia sám hlási na riešenie danej úlohy. Samozrejme, že môže nastať aj situácia kedy sa na úlohu hlási väčšie množstvo riešiteľov aké je potrebné na vyriešenie danej úlohy. V takom prípade sa otvára diskusia, v ktorej adepti na riešiteľa úlohy prezentujú svoje nápady súvisiace s danou úlohou. Na základe prezentácie nápadov by sa mali jednotliví adepti dohodnúť na tom, kto bude danú úlohu riešiť. Pokiaľ k dohode nedôjde tak o riešiteľovi úlohy rozhodne vedúci tímu.




**Obrázok 10** Pridanie úlohy medzi šprintové požiadavky

Potom ako priradíme úlohu k šprintu sa nám odhadovaný čas prejaví v grafe, ktorý sa nachádza na pravej strane. Môžeme v ňom sledovať časovú náročnosť vo vzťahu k celkovému času, ktorý máme na šprint vyhradený(ten sa vypočítal podľa hodín, ktoré sme priraďovali k členom tímu v predošlej kapitole).

Celý tento proces pridávania podúloh opakujeme až kým nenaplníme časovú kapacitu, ktorú máme k dispozícií pre aktuálny šprint.




**Obrázok 11** Vzťah priradeného času a celkového času na šprint

## *6.3 Manažment kvality*

### 6.3.1 Testovanie

Testovanie je veľmi dôležitou súčasťou vývoja softvéru. Prístup agilných metodík a teda aj Scrumu je odlišný od klasického prístupu. Testovanie nie je fáza vývoja, ale procesy testovania sú integrované do procesu vývoja softvéru.

Proces vytvorenia testov je proces, pri ktorom sa vytvárajú testy, ktoré pokrývajú funkcionalitu produktu podľa definovaných pravidiel. Tento proces zahŕňa prípravu testovacieho prostredia, návrh manuálnych, automatických a jednotkových testov.

#### 6.3.1.1 Konfigurácia vývojového prostredia

Vstup: vývojové prostredie MS Visual Studio 2010 alebo 2012
Výstup: nakonfigurované prostredie MS Visual Studio s podporou NUnit frameworku a s pluginom TestDriven.NET pre vytváranie a spúšťanie unit testov
Zodpovedný: programátor, vedúci testovania

V tomto kroku programátor s prípadnou technickou podporou vedúceho testovania nastaví svoje vývojové prostedie MS Visual Studio s frameworkom NUnit a pluginom TestDriven.NET pre vytváranie a spúšťanie unit testov.

#### 6.3.1.2 Konfigurácia testovacieho prostredia

Vstup: aktuálna verzia produktu, testovacie prostredie
Výstup: aktuálna verzia produktu nasadená v testovacom prostredí
Zodpovedný: vedúci testovania

Tento krok predstavuje nasadenie aktuálnej verzie produktu do testovacieho prostredia, ktoré má parametre ako reálne produkčné prostredie. Toto prostredie je následne pripravené na vykonávanie manuálnych či automatických testov.

#### 6.3.1.3 Návrh funkčných testov pre novú funkcionalitu (feature)

Vstup: popis novej funkcionality
Výstup: popísané testy v systéme Testopia
Zodpovedný: programátor, manažér testovania, vedúci testovania

V tomto kroku sa pre každú novú funkcionalitu, ktorá bude implementovaná počas šprintu, navrhnú testovacie prípady. Za návrh testov zodpovedajú programátori s podporou vedúceho testovania. Navrhnuté testy môžu byť manuálne aj automatické. Manažér

testovania je zodpovedný za výber vhodných testov a ohodnotenie priority jednotlivých testov.

#### 6.3.1.4 Vytvorenie unit testu

Vstup: testovaná trieda
Výstup: unit testy
Zodpovedný: programátor

Tento krok predstavuje vytvorenie unit testov pre jednotlivé testované triedy, pritom sa programátor musí držať stanovených pravidiel a konvencií. Unit testami by mal byť pokrytý celý kód.

### 6.3.2 Vytvorenie unit testu použitím frameworku NUnit vo vývojovom prostredí Visual Studio

V tejto časti je podrobne popísaný postup pri vytváraní unit testov využitím frameworku NUnit, nástroja pre vytváranie unit testov pre .NET programovacie jazyky. Uvedený postup je navrhnutý pre vývojové prostredie Visual Studio 2010 spolu s pluginom TestDriven.NET, ktorý umožňuje spúšťanie NUnit testov priamo z IDE. Tento postup by mal byť aplikovateľný aj pre Visual Studio 2012.

#### 6.3.2.1 Vytvorenie nového testovacieho projektu

Tento krok sa vykonáva len ak pre testovaný projekt ešte nie je vytvorený prislúchajúci testovací projekt.

1. Vo Visual Studio je potrebné otvoriť „Solution“, v ktorom sa nachádza projekt, ktorý sa bude testovať
2. V okne „Solution Explorer“ sa pravým tlačidlom klikne na „Solution“ a z kontextovej ponuky sa zvolí možnosť Add -> New Project
3. Z ponúkaných šablón v ľavom paneli sa vyberie Visual C# -> Windows a v strednom paneli sa následne vyberie projekt „Class Library“.
4. Vyplní sa meno testovacieho projektu. Meno testovacieho projektu musí mať rovnaké meno ako testovaný projekt s príponou Tests. Teda pre projekt s názvom „*NUnitTutorial*“ bude mať testovací projekt mať názov „*NUnitTutorialTests*“. V prípade ak je potrebné rozdeliť testy do niekoľkých testovacích projektov, projekt musí mať rovnaký názov ako časť projektu, ktorý testuje (napr. názov modulu, balíka tried)
5. Ak je potrebné vytvoriť viac testovacích tried pre jednu testovanú triedu, za názov testovacej triedy sa pridá poradové číslo počínajúc 1, napr.:
   *NUnitTutorialTests*

*NUnitTutorialTests1*
*NUnitTutorialTests2*

6. Je potrebné pridať pre projekt referencie na framework NUnit. Pravým tlačidlom sa klikne na testovací projekt, zvolí sa možnosť „Add Reference“, zvolí sa prvá záložka „.NET“. Tu je potrebné vyhľadať a pridať „*nunit.framework*“ a „*nunit.mocks*“.
7. Pridá sa referencia na testovaný projekt. Pravé tlačidlo myši na testovací projekt -> „Add Reference“ -> záložka „Projects“ a vyberie sa testovaný projekt
8. Testovací projekt je vytvorený a nakonfigurovaný

### 6.3.2.2 Vytvorenie unit testu

Tento krok predstavuje vytvorenie jednej NUnit testovacej triedy („TestFixture“), ktorá predstavuje množinu unit testov pre jednu triedu z testovaného projektu. Každá NUnit trieda obsahuje niekoľko metód (Test), pričom každá metóda predstavuje test pre jednu metódu testovanej triedy.

### 6.3.2.3 Vytvorenie NUnit triedy

NUnit trieda je testovacia trieda, ktorá obsahuje testy pre jednu triedu z testovaného projektu.

1. V okne „Solution Explorer“ sa pravým tlačidlom myši sa klikne na testovací projekt, zvolí sa možnosť Add -> New Item. (klávesová skratka Ctrl + Shift + A)
2. Z ponúkaných šablón sa zvolí „Visual C# Items -> Code -> Class“
3. Meno triedy sa vyplní rovnaké ako je meno triedy, ktorú bude táto trieda testovať doplnené o príponu „Test“. To znamená, že ak sa vytvára test napr. pre triedu s názvom „*CameraWatcher*“, testovacia trieda bude mať názov „*CameraWatcherTest*“
4. Po vytvorení triedy je potrebné pridať referencie na NUnit framework

```
using NUnit.Framework;
using NUnit.Mocks;
```

5. Nad názov metódy sa pridá atribút [TestFixture]. Ak je to potrebné, pridajú sa konfiguračné metódy pre testovaciu triedu. Na ukážke kódu č. 1 je vytvorená testovacia NUnit trieda spolu so konfiguračnými metódami a ich krátkym popisom.

### 6.3.2.4 Vytváranie unit testov

Jeden test predstavuje jednu metódu označenú atribútom [Test] v testovacej triede.

- Testovacia metóda musí byť typu „public void“, t.j. musí byť verejne prístupná a nesmie vracať žiadnu návratovú hodnotu a tiež nesmie brať žiadne argumenty, napr.:

```
[Test]
```

```
public void calculateCoordinatesTest() {}
```

- Testovacia metóda musí mať rovnaké meno ako metóda, ktorú testuje, doplnené o príponu Test. Ak testovaná metóda má názov „*calculateCoordinates*()“, potom testovacia metóda bude mať názov „*calculateCoordinatesTest*()“.
- Ak je potrebné vytvoriť viac testovacích metód (testov) pre jednu metódu, pridá sa za názov testovacej metódy poradové číslo počínajúc číslom 1, napr.:
  *calculateCoordinatesTest()*
  *calculateCoordinatesTest1()*
  *calculateCoordinatesTest2()*
- unit testami by mal byť pokrytý celý kód testovaného projektu, čo znamená, že pre každú public metódu testovanej triedy by mal existovať vhodný unit test.
- Každá NUnit trieda s Unit testami musí mať v hlavičke v komentári ďalšie informácie o testovacej triede :
  - meno autora alebo autorov testov
  - stručný popis testovanej funkcionality, čo a ako sa testuje, prípadné problémy či čo ešte treba doplniť
  - priemerný čas trvania testu

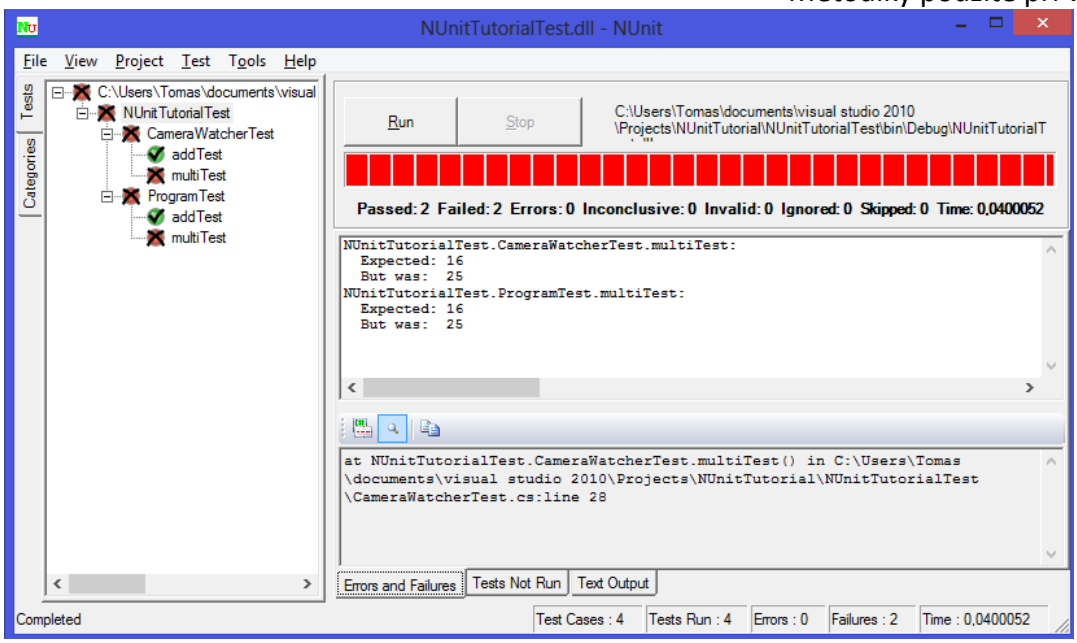
### 6.3.2.5 Spustenie unit testu

Vytvorený test je potrebné spustiť, aby bolo možné overiť správnosť a funkčnosť unit testu. Unit testy je možné spúšťať pomocou externých nástrojov NUnit frameworku alebo vďaka pluginu TestDriven.NET do Visual Studio aj priamo z vývojového prostredia. Tento krok popisuje spúšťanie testov z IDE.

Plugin TestDriven.NET umožňuje spúšťať unit testy priamo v konzole Visual Studio, alebo spustením grafického rozhrania NUnit Frameworku.

Pre spustenie testov v konzole IDE je potrebné kliknúť pravým tlačidlom myši na testovací projekt ak sa majú spustiť všetky testy. Pre spustenie testov konkrétnej testovacej triedy v konzole IDE sa klikne pravým tlačidlom na konkrétnu testovaciu triedu a z kontextovej ponuky sa vyberie možnosť „Run Test(s)“

Pre spustenie testov pomocou grafického rozhrania, ktoré poskytuje NUnit sa klikne na testovací projekt pravým tlačidlom a zvolí sa možnosť „Test With -> NUnit“. Následne sa spustí grafické rozhranie (obr. č. 1). V grafickom prostredí je možné spúšťať jednotlivé testy, testovacie triedy alebo celý projekt.




**Obrázok 12** NUnit grafické používateľské rozhranie pre spúšťanie testov

## *6.4 Manažment dokumentácie*

### 6.4.1 Roly

| Rola | Zodpovednosť | Procesy |
|---|---|---|
| Programátor | • Písanie komentárov k zdrojovému kódu<br>• Udržiavanie aktuálnosti komentárov k zdrojovému kódu | • Tvorba komentárov k zdrojovému kódu |
| Manažér dokumentácie | • Vytvorenie pravidiel pre komentovanie zdrojového kódu<br>• Finalizácia dokumentácie | • Vytvorenie pravidiel pre komentovanie<br>• Generovanie a sprístupňovanie dokumentácie |
| Manažér podpory vývoja | • Dokumentácia technologických požiadaviek produktu<br>• Dokumentácia postupu nasadenia | • Vytvorenie dokumentu s technologickými požiadavkami<br>• Vytvorenie dokumentu s postupom nasadenia produktu |
| Manažér kvality | • Dohľad nad kvalitou a úplnosťou komentárov | • Kontrolovanie komentárov |

### 6.4.2 Procesy

1. Vytvorenie pravidiel pre komentovanie
2. Tvorba komentárov k zdrojovému kódu
3. Kontrolovanie komentárov
4. Vytvorenie dokumentu s technologickými požiadavkami
5. Vytvorenie dokumentu s postupom nasadenia produktu
6. Generovanie a sprístupňovanie dokumentácie

#### 6.4.2.1 Vytvorenie pravidiel pre komentovanie

**Vstup:** Neexistujúce pravidlá pre komentovanie zdrojového kódu
**Výstup:** Dokument obsahujúci pravidlá pre komentovanie zdrojového kódu
**Zodpovedný:** Manažér dokumentácie

Manažér dokumentácie vytvorí pravidlá pre komentovanie zdrojových kódov, ktoré musia dodržiavať všetci členovia tímu. Výstupom tohto procesu je dokument, obsahujúci pravidlá pre komentovanie jednotlivých tried, metód a vlastností. Pravidlá obsahujú štruktúru a obsah komentárov, ktorými musí byť zdrojový kód okomentovaný, aby z neho mohla byť vygenerovaná dokumentácia.

#### 6.4.2.2 Tvorba komentárov k zdrojovému kódu

**Vstup:** Zdrojový kód bez komentárov
**Výstup:** Okomentovaný zdrojový kód
**Zodpovedný:** Programátor

Programátor doplní komentár do zdrojového kódu podľa pravidiel pre komentovanie, vytvorených manažérom dokumentácie. Proces spočíva vo vytvorení komentárov jednotlivých tried, metód a vlastností. Ak už boli komentáre vytvorené, ale neprešli kontrolou manažérom kvality, programátor musí upraviť, prípadne doplniť komentáre tak, aby zodpovedali pravidlám. Takto upravené komentáre znova prechádzajú kontrolou manažérom kvality.

#### 6.4.2.3 Kontrolovanie komentárov

**Vstup:** Okomentovaný zdrojový kód
**Výstup:** Skontrolované komentáre zdrojového kódu, zodpovedajúce pravidlám pre komentovanie
**Zodpovedný:** Manažér kvality

Manažér kvality skontroluje jednotlivé zdrojové kódy, či obsahujú všetky potrebné komentáre a či tie zodpovedajú pravidlám definovaným manažérom dokumentácie. V prípade, že nájde neokomentované, alebo slabo okomentované časti kódu, neschváli kontrolovaný súbor a vráti ho programátorovi na okomentovanie. Po opravení alebo doplnení komentárov programátorom musia byť opäť skontrolované manažérom kvality. Po schválení sa z komentárov môže vygenerovať dokumentácia.

#### 6.4.2.4 Vytvorenie dokumentu s technologickými požiadavkami

**Vstup:** Neexistujúci dokument s technologickými požiadavkami systému
**Výstup:** Dokument obsahujúci technologické požiadavky
**Zodpovedný:** Manažér podpory vývoja

Manažér podpory vývoja vytvorí dokument obsahujúci technologické požiadavky pre správne fungovanie výsledného softvérového produktu. V dokumente musí byť zoznam požiadaviek, ktoré musia byť splnené pre úspešné nasadenie produktu zákazníkom. Jedná sa o hardvérové (použitá architektúra, množstvo pamäte,...), ale aj softvérové (knižnica, *framework*,...) požiadavky.

#### 6.4.2.5 Vytvorenie dokumentu s postupom nasadenia produktu

**Vstup:** Neexistujúci postup nasadenia produktu
**Výstup:** Dokument obsahujúci postup nasadenia produktu
**Zodpovedný:** Manažér podpory vývoja

Manažér podpory vývoja vytvorí dokument opisujúci postup nasadenia produktu. Dokument nadväzuje na dokument opisujúci technologické požiadavky systému a podrobne opisuje všetky kroky potrebné pre nasadenie produktu. Produkt môže byť nasadený, až keď sú splnené všetky požiadavky uvedené v dokumente obsahujúcom technologické požiadavky.

#### 6.4.2.6 Generovanie a sprístupňovanie dokumentácie

**Vstup:** Vytvorené a schválené komentáre k zdrojovým kódom
**Výstup:** Sprístupnená dokumentácia
**Zodpovedný:** Manažér dokumentácie

Manažér dokumentácie vygeneruje dokumentáciu z komentárov zdrojového kódu a sprístupní ju všetkým členom tímu. Dokumentáciu vygeneruje nástrojom Doxygen, ktorý z komentárov zdrojového kódu vytvorí dokumentáciu vo forme webových stránok. Vytvorená dokumentácia obsahuje popis jednotlivých tried, metód a vlastností.

### 6.4.3 Dokumentovanie zdrojových kódov

#### 6.4.3.1 Dokumentácia tried

Komentár každej triedy musí obsahovať opis, ktorý vysvetľuje, na čo trieda slúži. Musí byť umiestnený bezprostredne pred triedou, ktorú opisuje a musí byť v rovnakom súbore ako je zdrojový kód.
Komentár pozostáva z nasledujúcich častí:

- <summary> - obsahuje detailný opis triedy, na čo trieda slúži

Ak je opis triedy dlhý, treba použiť tag <para> na rozčlenenie textu do odsekov.

V ukážke zdrojového kódu je uvedený príklad komentára triedy.

```
/// <summary>
/// Detailny opis triedy
/// </summary>
/// <remarks>Zoznam metod
```

### 6.4.3.2 Dokumentácia metód

Komentár každej metódy musí obsahovať podrobný opis jej funkčnosti. Musí byť umiestnený bezprostredne pred metódou, ktorú opisuje a musí byť v rovnakom súbore ako je zdrojový kód.
Komentár pozostáva z nasledujúcich častí:

- <summary> - podrobný opis metódy a jej funkčnosti. Aké vstupy má metóda, čo vykonáva a aké má výstupy.
- <param> - ak má metóda vstupné parameter, musí byť každý uvedený samostatným tagom, s menom aj opisom parametra. Inak sa neuvádza. Každý parameter musí mať uvedené:
    - názov, v parametri *name* tagu <param>
    - dátový typ
    - jeho opis
- <returns> - ak vracia metóda návratovú hodnotu, musí byť uvedený jej opis. Inak sa neuvádza. Návratová hodnota musí mať uvedené:
    - dátový typ
    - opis návratovej hodnoty

Ak je opis metódy dlhý, treba použiť tag <para> na rozčlenenie textu do odsekov.

V ukážke zdrojového kódu je uvedený príklad komentára metódy.

```
/// <summary>
/// Metoda ma na vstupe dve cele cisla, ktore vydeli a vrati
/// vysledok operacie, hodnotu typu float
/// </summary>
/// <param name="first">Delenec - parameter typu int</param>
/// <param name="second">Delitel - parameter typu int</param>
/// <returns>Vysledok operacie delenia - vysledok je typu float</returns>
```

### 6.4.3.3 Dokumentácia vlastností triedy

Komentár vlastností triedy (anglicky *property*) sa musí nachádzať bezprostredne pred vlastnosťou a musí sa nachádzať v rovnakom súbore ako je zdrojový kód. Komentár pozostáva z nasledujúcich častí:

- <summary> - obsahuje opis reprezentácie vlastnosti
- <value> - obsahuje opis hodnoty (premennej), ktorú vlastnosť reprezentuje

V ukážke zdrojového kódu je uvedený príklad komentára vlastnosti

```
/// <summary>
/// Vlastnost Name reprezentuje meno zamestnanca
/// </summary>
/// <value>
/// Vlastnost Name nastavuje/vracia _name premennu
/// </value>
```

# Preberací protokol

| | |
|---|---|
| Projekt: | Emotion Log |
| Odovzdávajúci tím: | Bc. Michal Biroš<br>Bc. Tomáš Caban<br>Bc. Tomáš Kunka<br>Bc. Filip Staňo<br>Bc. Tomáš Lekeň<br>Bc. Milan Martinkovič<br>Bc. Bálint Szilva |
| Preberajúci: | Ing. Martin Labaj |
| Dátum odovzdania: | 14.11.2012 |

| | |
|---|---|
| ..... | ..... |
| Ing. Martin Labaj | Bc. Filip Staňo<br>(za tím č.14) |