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

Zábavný systém pre spolucestujúcich v automobile

Tímový projekt

Dokumentácia riadenia projektu

Vedúci tímového projektu: Ing. Vanda Benešová, PhD.
Členovia tímu:
Bc. Patrik Polatsek
Bc. Martin Petluš
Bc. Peter Hamar
Bc. Róbert Sabol
Bc. Jakub Mercz
Bc. Lukáš Sekerák
Bc. Marianna Mušinská
Bc. Juraj Jarábek

Názov tímu: Tím č. 3 (Carlos)
Web: http://labss2.fiit.stuba.sk/TeamProject/2013/team03is-si/
Kontakt: team03.1314@gmail.com
Akademický rok: 2013/2014

Obsah

1. Úvod

Tento dokument poskytuje informácie o riadení projektu s názvom Zábavný systém pre spolucestujúcich v automobile, ktorý je vypracovaný v rámci predmetu Tímový projekt. Tento projekt je riešený tímom č. 3 - Carlos.

Tím Carlos je pod vedením Ing. Vandy Benešovej, PhD. v nasledovnom zložení:

- Bc. Patrik Polatsek - vedúci tímu
- Bc. Martin Petluš - manažér kvality
- Bc. Peter Hamar - manažér podpory vývoja
- Bc. Róbert Sabol - manažér monitorovania projektu
- Bc. Jakub Mercz - manažér rozvrhu
- Bc. Lukáš Sekerák - manažér rizík
- Bc. Marianna Mušinská - manažér ľudských zdrojov
- Bc. Juraj Jarábek - manažér dokumentovania

Úvodná časť popisuje náš tím, jej členov a ich kvalifikácie. Obsahuje Zoznam kompetencií tímu, ktorý bol vypracovaný pri uchádzaní sa o tímový projekt.

Jadro dokumentu tvoria nasledovné metodiky na dolnej úrovni vytvorené v rámci predmetu Manažment v informačných systémoch/Manažment v softvérovom inžinierstve:

- Metodika evidencie nových úloh v Redmine
- Metodika písania jednotkových (Unit) testov
- Metodika zdielania zdrojov v Google Drive
- Metodika - Dokumentácia zdrojových kódov za použitia Doxygen (Manažment zdrojových kódov)
- Metodika plánovania úloh
- Metodika - Manažment verzií zdrojového kódu pomocou Git
- Metodika prehliadky kódov vo dvojiciach
- Metodika - Manažment požiadaviek na zmenu

Na záver dokumentu sú pridané jednotlivé zápisy zo stretnutí, ktoré dokumentujú jednotlivé stretnutia tímu. V každom zápise je uvedené čo bolo cieľom tohto stretnutia a aké boli úlohy jednotlivých členov tímu.

2. Zoznam kompetencií tímu

V tejto časti sa nachádza zoznam kompetencií tímu, ktorý sa odovzdával za účelom získania projektu na začiatku zimného semestra. Obsahuje predstavenie tímu Carlos spolu s ponukami k 3 projektom. V závere tejto kapitoly sa nachádza poradie preferencií k projektom.

2.1 Predstavenie tímu

Patrik Polatsek (IS)

- ovláda: C/C++, Java, SQL, OpenCV
- bakalárska práca: Detekcia žmurknutia používateľa počítača (prostredníctvom snímkov z webkamery sa detegovalo žmurknutie v C++ programe s využitím knižnice OpenCV)
- pracovná pozícia: SAP ABAP developer

Martin Petluš (IS)

- ovláda: C/C++, Java, SQL, JavaScript
- bakalárska práca: Strojové učenie ohodnocovacej funkcie pre vyhľadávač (vo webovom vyhľadávači sme zaznamenávali dopyty používateľov a na základe zaznamenaných dát, sme sa naučili modely pre používateľov, podľa ktorých sme používateľom zoraďovali výsledky vo vyhľadávači, Java a Ruby on Rails)
- pracovná pozícia: vyvíjanie webovej aplikácie, chat klient, video konferencie (HTML5)

Jakub Mercz (IS)

- ovláda: C/C++, Java, SQL, XML
- bakalárska práca: Vytváranie databázových dopytov v prirodzenom jazyku (aplikácia na základe vstupu v prirodzenom jazyku vytvorila SQL dopyt pre databázu s využitím parsovania a prehľadávania štruktúry databázy)

Lukáš Sekerák (IS)

- ovláda: C/C++, Java, MysqlSQL, OracleSql, PHP, Ruby, XML, JavaScript, HTML5, UML, CSS, Android, jQuery, phoneGap. 1 rok Android skúsenosti. 7 rokov web developer.
- bakalárska práca: Interaktívna vizualizácia informačnej siete (Aplikácia ktorá zobrazovala vzťahy vyše 40 miliónov entít. Entity boli zobrazované vrcholmi v grafe a predstavovali rôzne osoby, tel. čísla, mená, informácie,...)
- pracovná pozícia: CIIT.at Java web junior developer + Android developer

Peter Hamar (IS)

- ovláda: C/C++, PHP, Java, SQL, Ruby on Rails, XML

- bakalárska práca: Správa citácií (automatizovane tvorby citácií a bibliografických odkazov v existujúcom nástroji Annota)

Róbert Sabol (SI)

- ovláda: C/C++, Java, MySQL, C#, Objective-C, XML, HTML, JavaScript, Android development, iOS development, UML
- bakalárska práca: Prepojenie TV vysielania a sociálnych sietí (Android aplikácia)
- pracovná pozícia: iOS developer

Juraj Jarábek (SI)

- ovláda: C, Java, SQL, Android development, Perl, Python
- bakalárska práca: Aplikácia pre záznam nálady s prvkami gemifikácie pre operačný systém Android
- pracovná pozícia: Java Developer

Marianna Mušinská (IS)

- ovláda: C/C++, Java, SQL
- bakalárska práca: Databázový systém webshopu (webová aplikácia slúžiaca ako webshop)

2.2 Téma č. 1 - 2 Zábavný systém pre spolucestujúcich v automobile

Náš tím by chcel vytvoriť program, ktorý bude obohacovať realitu o virtuálne prvky. Aplikácia by mohla pomocou GPS a rozoznávaním objektov z internej databázy upozorňovať a zvýrazňovať zaujímavé objekty v okolí vodiča. Aplikácia by zobrazovala základné informácie o okolitých objektoch, príp. ich vzdialenosť od auta, pričom údaje získa z internetu a zo svojej databázy (napr. pri pamiatkach zobrazí otváracie hodiny, výšku vstupného, fotografie vo forme pútača). Používateľ si na začiatku zvolí kategórie, na ktoré chce byť upozorňovaný (pamätihodnosti, reštaurácie, kaviarne, …). Aplikácia bude doplnená aj o hru pre deti, ktoré by prostredníctvom android mobilného telefónu ovládali postavičku, ktorá by sa pohybovala v zobrazovanej realite.

V našom tíme máme človeka, ktorý pracoval v rámci svojej bakalárskej práce s knižnicou OpenCV, kde rozoznával žmurkanie používateľa pred webkamerou, pričom na detekciu využil viaceré nezávislé metódy. Väčšina tímu mala v bakalárskom štúdiu predmet Počítačová grafika, kde sme sa naučili základy grafiky a prácu s knižnicou OpenGL. Traja členovia tímu v rámci svojej bakalárskej práce vytvorili aplikáciu pre platformu Android.

2.3 Téma č. 2 - 6 Virtuálna FIIT na mobile

Téma virtuálna FIIT na mobile nás zaujala tým, že presne definuje čo by bolo našou úlohou. Vytvorili by sme interaktívnu aplikáciu s pekným používateľským prostredím a grafikou. Krásnu grafiku by nám spravila Marianna, ktorá má talent pre grafické cítenie.

Hlavným cieľom by bolo rozšíriť aplikáciu medzi najväčší počet študentov. Preto navrhujeme spraviť anketu (najlepšie teda medzi prvákmi / novými študentmi), spýtať sa kolegov aké sú ich požiadavky a podľa týchto požiadaviek vytvoriť aplikáciu. Predpokladáme, že typický používatelia by boli noví študenti, ktorí našu budovu nepoznajú a majú problém s orientáciou.

Ďalšie nápady sú:

- plnohodnotná navigácia spolu s QR kódmi
- spolupráca aplikácie s hladnystudent.sk, mhd, možno okolitým prostredím
- rôzne metadáta učební a kancelárií, napríklad kapacita, názov, aktuálna výučba v triede, v prednáške
- prepojená aplikácia s AIS študenta, navigácia, kde má študent najbližšie cvičenie
- prepojenie so sociálnymi sieťami - (priatelia, skupiny (krúžky), poloha priateľov)
- chat v rámci svojho krúžku

V tíme máme 3 expertov na Android, 2 ďalších na JavaScript, jeden člen tímu má skúsenosti s parsovaním hladnystudent.sk. Zároveň máme experta na QR kódy a vyššie spomenutú grafičku. Sme teda dosť silný team pre túto tému a schopní zrealizovať akýkoľvek nápad.

2.4 Téma č. 3 - 5 Vizualizácia informácií v obohatenej realite

Náš tím by chcel vytvoriť aplikáciu, vďaka ktorej by bol používateľ schopný interagovať napr. s grafmi v obohatenej realite. S grafmi by používateľ pracoval prostredníctvom nasnímaných špeciálnych značiek, príp. pohybom rúk.

V našom tíme máme človeka, ktorý pracoval v rámci svojej bakalárskej práce s knižnicou OpenCV, kde rozoznával žmurkanie používateľa pred webkamerou, pričom na detekciu využil viaceré nezávislé metódy. Väčšina tímu mala v bakalárskom štúdiu predmet Počítačová grafika, kde sme sa naučili základy grafiky a prácu s knižnicou OpenGL. Traja členovia tímu v rámci svojej bakalárskej práce vytvorili aplikáciu pre platformu Android.

2.5 Zoradenie všetkých tém podľa priority

1. Zábavný systém pre spolucestujúcich v automobile
2. Virtuálna FIIT na mobile
3. Vizualizácia informácií v obohatenej realite
4. Analýza výsledkov výskumu
5. Webový komunitný systém otázok a odpovedí
6. Digital SweatShop
7. Trojdimenzionálne UML
8. Distribuované počítanie na FIIT
9. Prehliadka kódov v tímových projektoch
10. Sledovanie pohľadu pri používaní aplikácií
11. Monitor programátora v IDE
12. Interaktívne hry na mobile s multimediálnym obsahom
13. 3D Robotický futbal

3. Úlohy členov tímu

V tejto kapitole sa nachádza prehľad jednotlivých členov tímu spolu s ich primárnymi úlohami a manažérskymi rolami.

Bc. Patrik Polatsek - vedúci tímu
Zameranie:

- práca s knižnicou OpenCV
- práca na úlohách súvisiacimi s počítačovým videním
- práca na detekcii a rozoznávaní objektov zo snímok

Bc. Martin Petluš - manažér kvality
Zameranie:

- technologická podpora tímu
- práca na module počítajúcom polohu textu na okne
- zodpovedný za virtuálny server tímu

Bc. Peter Hamar - manažér podpory vývoja
Zameranie:

- práca s knižnicou OpenGL
- práca s 3D objektmi a scénou
- vytvorenie hry, ktorá bude využívať prvky počítačového videnia

Bc. Róbert Sabol - manažér monitorovania projektu
Zameranie:

- práca na Android module
- práca na rozpoznávaní gest
- vytvorenie TCP modulu v Module Android

Bc. Jakub Mercz - manažér rozvrhu
Zameranie:

- práca s údajmi z Kinectu
- detekcia tváre, hlavy

Bc. Lukáš Sekerák - manažér rizík
Zameranie:

- návrh celej architektúry, integrácia modulov

- práca s hlavným algoritmom, riadiacim modulom
- vytvorenie TCP serverového modulu, databázového modulu

Bc. Marianna Mušinská - manažér ľudských zdrojov

Bc. Juraj Jarábek - manažér dokumentovania
Zameranie:

- návrh a implementácia časti Android modulu
- dokumentácia zdrojových kódov (Doxygen)

4. Metodiky (dolná úroveň)

V tejto kapitole sa nachádzajú jednotlivé metodiky členov tímu Carlos, ktoré boli vypracované k predmetu Manažment v informačných systémoch/Manažment v softvérovom inžinierstve.

V nasledovnej tabuľke sa nachádza prehľad metodík a ich autorov.

Metodika	**Autor**
Metodika evidencie nových úloh v Redmine	Polatsek
Metodika písania Jednotkových (Unit) testov	Hamar
Metodika - Dokumentácia zdrojových kódov za použitia Doxygen (Manažment zdrojových kódov)	Sekerák
Zdielanie zdrojov v Google Drive	Petluš
Metodika - Manažment požiadaviek na zmenu	Sabol
Metodika plánovania úloh	Mercz
Metodika prehliadky kódov vo dvojiciach	Mušinská
Metodika - Manažment verzií zdrojového kódu	Jarábek

4.1 Metodika evidencie nových úloh v Redmine

Autor: Patrik Polatsek

4.1.1 Úvod

Táto metodika definuje pravidlá a postupy pri zakladaní úloh vo webovom nástroji na manažment projektov Redmine. Po schválení a priradení úloh na stretnutí pred ďalším šprintom je potrebné úlohy zaevidovať do systému procesmi spomínaných v tejto metodike.

Pravidlami definovanými v tejto metodike sa riadi manažér plánovania, ktorý je za evidenciu nových pridelených úloh do systému Redmine a s nimi súvisiacich činností zodpovedný.

4.1.1.1 Použité pojmy a skratky

- *Redmine*: open-source webový nástroj na manažment projektov a sledovanie chýb, ktorý zahŕňa okrem iného aj Ganttov graf a kalendár
- *Scrum*: metóda agilného vývoja softvéru
- *Šprint*: základná časová jednotka vo vývoji v Scrume. Každému šprintu predchádza plánovanie, kde sú identifikované úlohy, ktoré sa majú do konca šprintu vykonať.

4.1.1.2 Roly a zodpovednosti

V Tabuľke 1 sú uvedené roly, ktoré vystupujú v tejto metodike a ich zodpovednosti relevantné s touto metodikou.

Rola	**Zodpovednosť**
Manažér plánovania	● Zaevidovanie a pridelenie novej úlohy ● Zaevidovanie nového šprintu

Tabuľka 1. Roly a zodpovednosti

4.1.1.3 Príbuzné metodiky

- Metodika plánovania
- Metodika zdielania zdrojov na Google Drive

4.1.2 Vytvorenie novej úlohy v Redmine

Vstup: identifikovaná úloha na stretnutí tímu
Výstup: vytvorená nová úloha
Zodpovedný: manažér plánovania

Na stretnutí tímu sa pri novom šprinte identifikujú nové úlohy, ktoré sa následne priradia ako je opísané v *Metodike plánovania*. Pri založení novej úlohy v systéme Redmine, ktorá nie je priamou podúlohou už k existujúcej úlohe, sa manažér plánovania riadi nasledovnými krokmi, ktoré sú zobrazené na Obrázku 1. V prípade vytvorenia novej podúlohy manažér plánovania postupuje krokmi opísanými v časti *Vytvorenie novej podúlohy k existujúcej úlohe* v tejto metodike.

1. Prihlásiť sa do systému Redmine.
2. Vybrať si príslušný projekt.
3. V prípade, že nie je v systéme zadefinovaný šprint, do ktorého sa má úloha pridať, je potrebné pred vykonaním nasledovného kroku vytvoriť tento šprint ako je opísané v tejto metodike v časti *Vytvorenie nového šprintu* od kroku číslo 3.
4. Kliknúť na tlačidlo *New issue*. Zobrazí sa okno zobrazené na Obrázku 1.
5. Vybrať typ úlohy v časti *Tracker* ako je uvedené v Tabuľke 2.

Typ úlohy	**Prípady použitia**
Feature	● úlohy týkajúce sa vývoja ● implementačné úlohy ● úlohy týkajúce sa testovania
Support	● úlohy týkajúce sa analýzy ● úlohy týkajúce sa návrhu ● tvorba dokumentácie ● všeobecné úlohy týkajúce sa chodu tímového projektu
Bug	● opravy zistených chýb

Tabuľka 2. Typy úloh

6. Napísať výstižný názov úlohy v časti *Subject* v nasledovnom tvare: *[modul] popis*, kde
   - *modul* charakterizuje zaradenie úlohy jedným slovom – opisuje modul alebo oblasť, ktorej sa úloha týka
   - *popis* opisuje výstižne (max. 5 slovami) pointu úlohy.
7. Opísať štruktúrovane úlohu v časti *Decsription* nasledovne:
   1. Popis prvej časti úlohy

2. Popis druhej časti úlohy
3. …

Jedna úloha je rozdelená číselne do samostatných bodov, na ktoré sa potom člen tímu pri aktualizácii stavu svojej úlohy odkazuje.

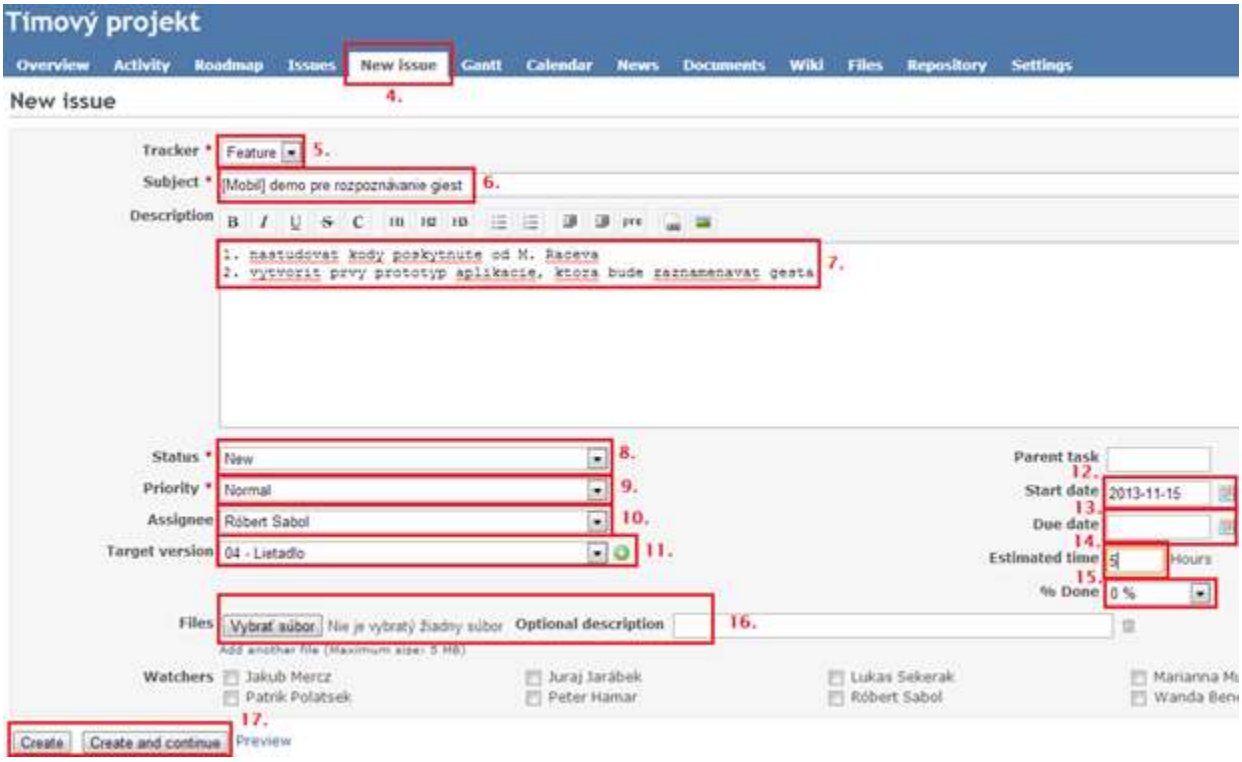
8. Ponechať *Status* úlohy na defaultný – *New*.
9. Zvoliť prioritu úlohy podľa jej dôležitosti v časti *Priority* podľa Tabuľky 3.

Priorita úlohy	**Prípady použitia**
Normal	Štandardná úloha, ktorá by mala byť vyriešená podľa vopred zadaného termínu
High	Dôležitá úloha, ktorá musí byť vyriešená do ďalšieho týždňa
Low	Úloha nie je dôležitá a jej vyriešenie nie je kľúčové pre chod tímu a systému
Urgent	Veľmi dôležitá úloha, ktorú treba riešiť prioritne
Immediate	Závažná úloha, ktorá musí byť vyriešená okamžite po jej vytvorení

Tabuľka 3. Priority úloh

10. Priradiť úlohu zvolenému členovi tímu v časti *Assignee*.
11. Priradiť úlohu do šprintu v časti *Target version*.
12. Zvoliť počiatočný dátum v časti *Start date* na dátum, kedy bola úloha členovi tímu zadaná.
13. V prípade, že má byť úloha vyriešená skôr než je koniec šprintu, vybrať tento dátum v časti *Due date*.
14. Vyplniť časť *Estimated* time v prípade, že sa na úlohu stanoví odhadovaný čas v hodinách.
15. Percentuálne dokončenie úlohy v časti *% done* ponechať na 0.
16. Ak k vypracovaniu danej úlohy existuje súvisiaci dokument, vložiť tento dokument cez tlačidlo *Vybrať súbor* v časti *Files* a vložiť k nemu max. 5 slovný popis v časti *Optional Description.* V prípade, že sa bude prikladať viac súborov, stlačiť tlačidlo *Add another file* a postup v tomto kroku opakovať. Do systému Redmine je možné vkladať súbory len s max. veľkosťou 5 MB. V opačnom prípade je potrebné vložiť tento súbor do zdielaného priečinku na Google Drive podľa postupu opísanej v *Metodike zdielania zdrojov na Google Drive*. K takto vloženému súboru vygenerujeme link, ktorý vložíme na koniec popisu úloh v časti *Description* nasledovne: *Príloha X: link,* kde
    - *X* je poradové písmeno prílohy
    - *link* je link k súboru na Google Drive

17. Kliknúť na tlačidlo *Create* alebo *Create and continue*, ak sa bude zadávať ešte ďalšia úloha do systému.
18. V prípade, že má úloha priamo súvisieť s existujúcou úlohou, ktorá je už zaevidovaná v systéme, je potrebné postupovať krokmi v časti *Vytvorenie závislosti medzi úlohami* v tejto metodike od kroku 3.



Obrázok 1. Postup pri vytváraní novej úlohy v Redmine

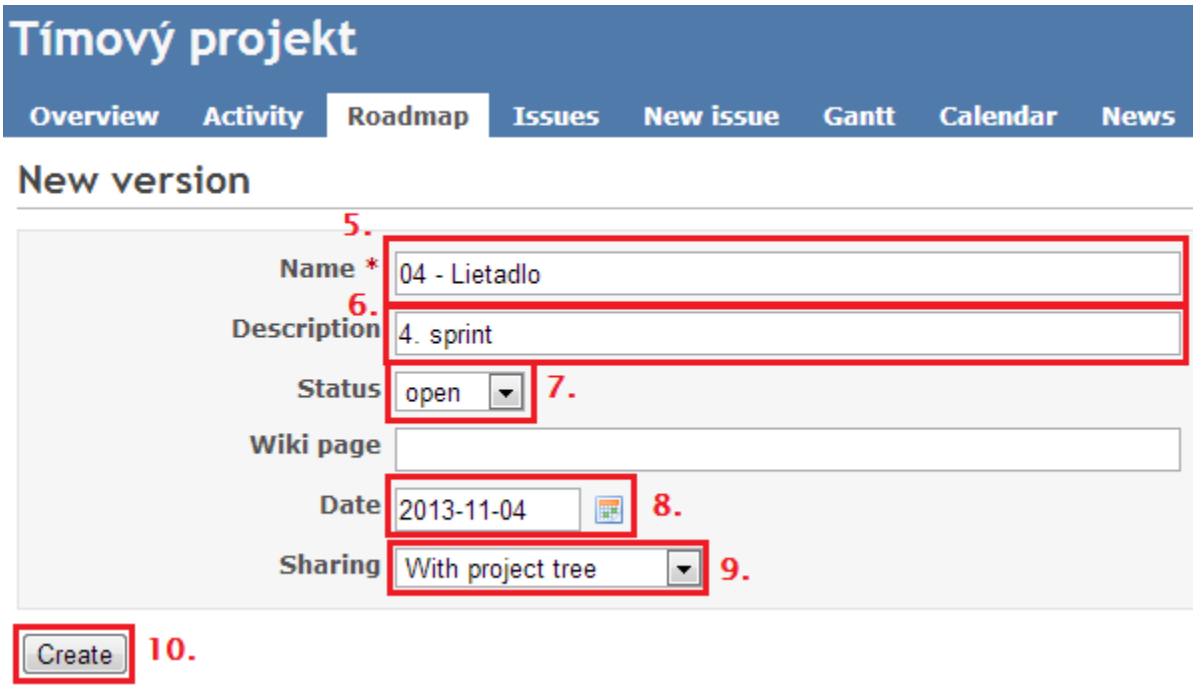
4.1.2.1 Vytvorenie nového šprintu

Vstup: ukončenie predchádzajúceho šprintu, dokument s novými identifikovanými úlohami
Výstup: zadefinovaný šprint v Redmine
Zodpovedný: manažér plánovania

Pri zadefinovaní nového šprintu do systému Redmine musí manažér plánovania postupovať týmito krokmi, ktoré sú zobrazené aj na Obrázku 2:

1. Prihlásiť sa do systému Redmine.
2. Vybrať si príslušný projekt.
3. Kliknúť na tlačidlo *Settings* – *Versions*.
4. Kliknúť na *New version.* Zobrazí sa okno z Obrázku 2.

5. Zadať názov šprintu v časti *Name* v nasledovnom tvare: *por.č. – meno*, kde
   - *por.č.* je 2-ciferné poradové číslo šprintu
   - *meno* je jednoslovný názov šprintu.
6. Do časti *Description* napísať *X. šprint*, kde X je poradové číslo šprintu.
7. Ponechať stav šprintu na otvorený – *open* v časti *Status.*
8. V časti *Date* zadať dátum mítingu, ktorému začiatok šprintu predchádzal.
9. *Sharing* nastaviť na *With project tree*.
10. Kliknúť na tlačidlo *Create*.



Obrázok 2. Postup pri zakladaní šprintu v Redmine

4.1.2.2 Vytvorenie novej podúlohy k existujúcej úlohe

Vstup: identifikovaná nová podúloha k existujúcej úlohe
Výstup: vytvorená nová podúloha v Redmine
Zodpovedný: manažér plánovania

V prípade zaevidovania novej úlohy do systému Redmine, ktorá je podúlohou k existujúcej úlohe, musí manažér plánovania postupovať nasledovne:

1. Prihlásiť sa do systému Redmine.
2. Vybrať si príslušný projekt.
3. Kliknúť na tlačidlo *Issues*.

4. V prípade potreby si vyselektovať úlohy pomocou filtra cez ich stav a potvrdiť cez *Apply*.
5. Kliknúť na úlohu, ktorá má byť nadúlohou tejto úlohy.
6. Kliknúť na *Add* v časti *Subtasks* (bod A. na Obrázku 3).
7. Pokračovať v tejto metodike od kroku 3 v časti *Vytvorenie novej úlohy v Redmine*.

Obrázok 3. Pridanie podúlohy alebo závislosti k úlohe v Redmine

4.1.2.3 Vytvorenie závislosti medzi úlohami

Vstup: identifikovanie závislosti medzi novou zaevidovanou úlohou a inou existujúcou úlohou
Výstup: vytvorená závislosť medzi úlohami
Zodpovedný: manažér plánovania

Ak pri zadaní úloh do systému Redmine je medzi úlohami identifikovaná vzájomná závislosť, manažér plánovania postupuje nasledovnými krokmi:

1. Prihlásiť sa do systému Redmine.
2. Vybrať si príslušný projekt.
3. Kliknúť na tlačidlo *Issues*.
4. V prípade potreby si vyselektovať úlohy pomocou filtra cez ich stav a potvrdiť cez *Apply*.
5. Kliknúť na úlohu, ktorá má s úlohou súvisieť.
6. Kliknúť na *Add* v časti *Related issues* (bod B na Obrázku 3).
7. Zadať typ závislosti úlohy, s ktorou úloha súvisí z možností, ktoré uvádza Tabuľka 4.

Závislosť	**Prípad použitia**	**Následky**
Related to	Vytvorenie prepojenia (linku) na súvisiacu úlohu, ďalšie závislosti medzi úlohami nie sú.	-
Duplicates	Vyriešenie súvisiacej úlohy vyrieši aj túto úlohu.	Zmena stavu súvisiacej úlohy na *closed* automaticky uzavrie aj túto úlohu.
Duplicated by	Vyriešenie tejto úlohy vyrieši aj súvisiacu úlohu (opak *Duplicates*).	Zmena stavu tejto úlohy na *closed* automaticky uzavrie aj súvisiacu úlohu.
Blocks	Úloha, ktorej vyriešenie je potrebné na vyriešenie súvisiacej úlohy.	Pokým táto úloha nemá stav *closed* nie je možné uzavrieť súvisiacu úlohu.
Blocked by	Na vyriešenie tejto úlohy je potrebné vyriešiť súvisiacu úlohu (opak *Blocks*).	Pokým súvisiaca úloha nemá stav *closed* nie je možné uzavrieť túto úlohu.

Tabuľka 4. Závislosti medzi úlohami

8. Zadať ID závislej úlohy.
9. V prípade, že úloha súvisí aj s ďalšou úlohou, opakovať postup od kroku 5.

4.2 Metodika písania jednotkových (Unit) testov

Autor: Peter Hamar

4.2.1 Úvod

Cieľom tejto metodiky je definovať jednotné spôsoby písania testov pre tím Carlos. K jednotlivým štádiám testovania táto metodika definuje zodpovedné osoby a taktiež jednotlivé výstupy pre konkrétne časti. Vývojovým nástrojom je Visual Studio a teda táto metodika bude určovať pravidlá písania unit testov v tomto vývojovom prostredí.

4.2.2 Zoznam pojmov

- Unit test – jednotkový test, slúži na overenie určitej malej časti (jednotky) funkcionality.
- Visual Studio – vývojové prostredie, používané kvôli dobrej podpore jazyka C++.

4.2.3 Zoznam skratiek

- VS – Visual Studio

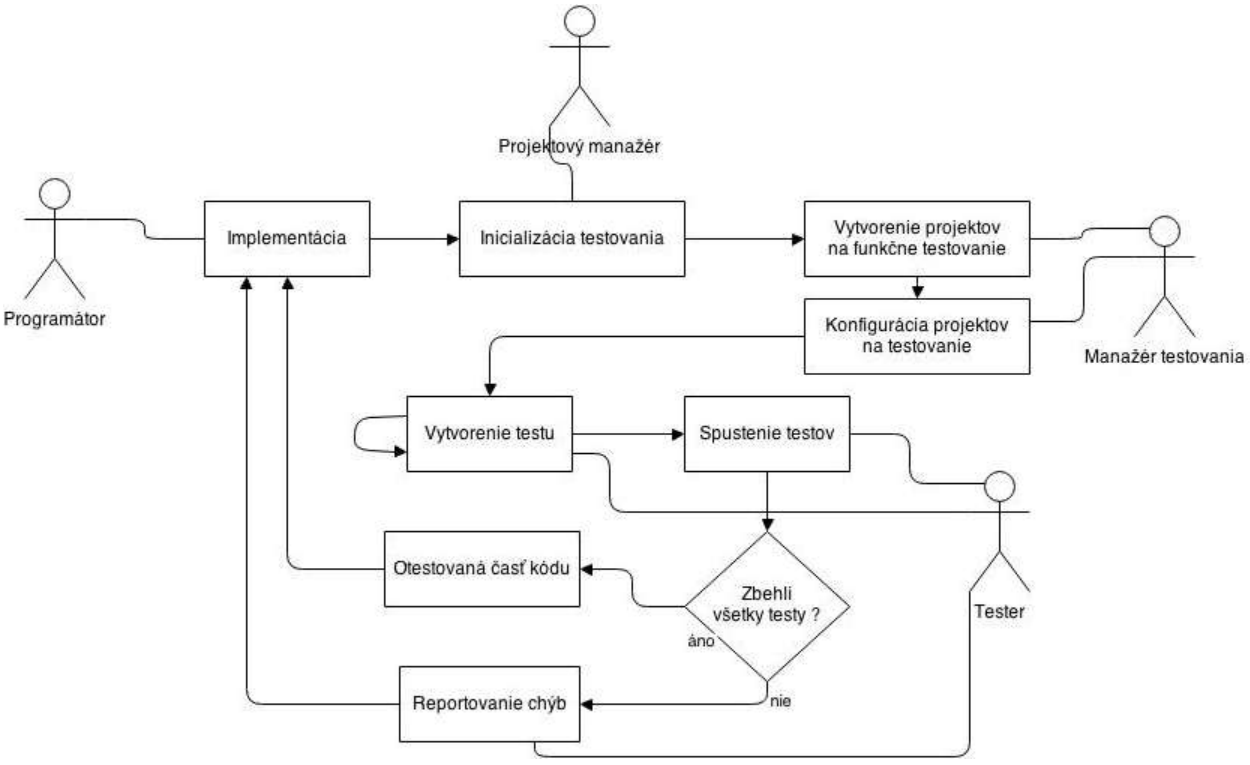
4.2.4 Zoznam nadväzujúcich metodík

- metodika riešenia problémov
- metodika kompilácie kódu

4.2.5 Zodpovednosť za procesy a momenty spustenia procesov

Proces	Zodpovednosť	Kedy
1.Inicializácia Unit testovania	Projektový manažér	Začatie fázy implementácie
2.Vytvorenie projektov na funkčné testovanie	Manažér testovania	Po inicializácii testovania
3.Konfigurácia projektov na testovanie	Manažér testovania	V momente vytvorenia projektov na testovanie
4.Vytvorenie testu	Tester	Po nastaveni projektov
5.Spustenie testu	Tester	V momente ukončenia vytvárania testu
6.Reportovanie chýb	Tester	Po spusteni testu/ov

4.2.6 Diagram



4.2.7 Podrobný opis jednotlivých procesov

1. Inicializácia unit testovania

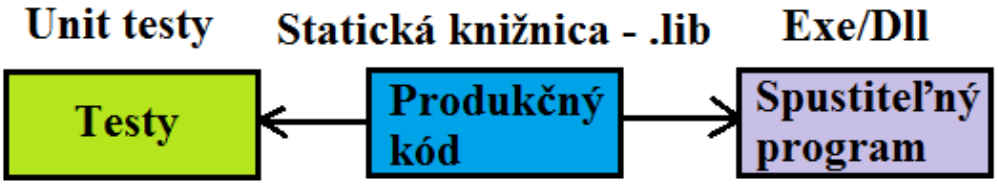
Vstup: Časť funkčného kódu.
Výstup: Overený funkčný kód.
Zodpovedný: Projektový manažér.

Popis: Cieľom tohto procesu je, aby Manažér testovania nastavil všetky potrebné detaily vo vývojovom nástroji – VS 2012. Vďaka čomu bude možné následne vytváranie a spúšťanie samotných testov.
Projektový manažér informuje Manažéra testovania, že je potrebné pripraviť projekt na písanie jednotkových testov. Tomuto procesu predchádza započatie fázy implementácie. Následne nasleduje proces vytvorenia projektov na funkčné testovanie.

2. Vytvorenie projektov na funkčné testovanie

Vstup: Vytvorený projekt vo VS
Výstup: Upravený projekt na písanie jednotkových testov.
Zodpovedný: Manažér Testovania.

Popis:
Manažér testovania spustí projekt, v ktorom sa bude vytvárať kód. Je potrebné vytvoriť 3 projekty v jednom solution. Schéma podľa ktorej je potrebné projekty vytvoriť :



- Produkčný kód – tento projekt by mal byť už vytvorený. Vytvorí sa hneď v momente začatia implementácie. Mal by byť v jazyku C++, vytvorený ako „Empty project“ a ako „Win32 Console Application“.
- Unit Testy –

1. V otvorenom „solution“ vo VS je potrebné kliknúť pravým tlačidlom na „solution“.
2. V roletovom menu vybrať možnosť „Add->New Project“.
3. Následne sa zobrazí okno, kde je potrebné vybrať typ projektu a to konkrétne „Visual C++->Test->Native Unit Test Project“.

4. Výber potvrďte kliknutím na tlačidlo „ok“.

- Spustiteľný program -

1. V otvorenom „solution“ vo VS je potrebné kliknúť pravým tlačidlom na „solution“.
2. V roletovom menu vybrať možnosť „Add->New Project“.
3. Následne sa zobrazí okno, kde je potrebné vybrať typ projektu, a to konkrétne „Visual C++->Win32-> Win32 Console Application“.
4. Potom sa zobrazí ďalšie okno, kde je potrebné kliknúť na tlačidlo „next“. Pokračujte zaškrtnutím políčka „Empty project“, tým sa znefunkční políčko „Precompiled Header“. Výber potvrdíme tlačidlom „finish“.

Tento proces je ukončený v momente keď v jednom „solution“ sú vytvorené všetky 3 projekty. Následne sa pristúpi k procesu konfigurácie projektov.

3. Konfigurácia projektov na testovanie

Vstup: Projekty vo VS.
Výstup: Nakonfigurované projekty, pripravené na vytváranie jednotkových testov.
Zodpovedný: Manažér Testovania.

Popis: Cieľom je konfigurácia jednotlivých projektov, pre správne písanie unit testov.

- **Produkčný kód**
  1. Vo vytvorenom solution kliknite pravým tlačidlom myši na projekt s produkčným kódom.
  2. Následne v roletovom menu zvoľte možnosť properties. Po tejto voľbe sa zobrazí okno s nastaveniami projektu.
  3. V ľavom hornom rohu nastavte voľbu „Configuration“ na „All Configurations“.
  4. V nastavovaní pokračujte vybratím možnosti „Configuration Properties“ a následne vyberte možnosť “General“.
  5. V pravej časti okna, voľbu „Configuration Type“ nastavte na „Static library (.lib)“. Výber potvrďte stlačením tlačidla „Apply“ a ešte raz to celé potvrďte kliknutím na tlačidlo „ok“. Zobrazené okno sa zavrie a uloží vaše nastavenia

- **Unit Testy a Exe/DLL** – Postup pre konfiguráciu oboch projektov je skoro identický, teda je ho potrebné vykonať pre oba projekty.

  1. Pravým tlačidlom myši kliknite na prvý z projektov a z roletového menu vyberte možnosť „Properties“.

2. Možnosť „Configuration“ nastavte na „All Configurations“.
3. Pokračujte zvolením možnosti „Common Properties“, ktoré je dostupné v okne pod predchádzajúcou voľbou.
4. V zobrazenom „submenu“ zvoľte možnosť „Framework and References“.
5. Kliknite na tlačidlo „Add New Reference“ a v zobrazenom okne vyberte statickú knižnicu – Produkčný kód, voľbu potvrdte tlačidlom „ok“.
6. Následne zvoľte možnosť C/C++ v zobrazenom okne.
7. V zobrazenom pravom panely zvoľte možnosť „Additional Include Directories“ a následne vyberte voľbu „edit“.
8. Pridajte nový riadok a to „$(SolutionDir)\ProductLibrary;%(AdditionalIncludeDirectories)“. S tým, že „ProductLibrary“ prepíšte na názov projektu s produkčným kódom. Voľbu potvrďte kliknutím na tlačidlo „ok“.

- **Iba pre Unit Test project:**
  1. Je potrebné kliknúť voľbu „Configuration Properties“.
  2. Zvoliť „General“ a zmeniť hodnotu poľa „Common Language Runtime Support“ na hodnotu „/clr“.
  3. Všetko potvrdiť tlačidlom „apply“a ok.

- **Iba pre Exe/DLL:**
  1. Je potrebné kliknúť na hlavnú voľbu „Configuration Properties“.
  2. V zobrazenom pravom okne zmeňte pole „Configuration Type“ na hodnotu „Application (.exe)“ resp. „Dynamic library (.dll)“.
  3. Všetko podtvrdte tlačidlom „apply“ a „ok“.

Týmto je konfigurácia projektov ukončená a sú pripravené na písanie jednotkových testov. Po tomto procese sa pristúpi k písaniu unit testov.

4. Vytvorenie unit testu

Vstup: Vytvorené solution s potrebnými projektami pripravenými na písanie jednotkového testu.
Výstup: Projekt s napísaným jednotkovým testom.
Zodpovedný: Tester.

Popis:

1. Vo vytvorenom projekte na písanie unit testov je potrebné vytvoriť súbor s príponou .cpp a to tak, že pravým tlačidlom myši kliknite na Source Files v unit test projekte a z roletového menu vyberte „Add->New Item“.
2. Následne sa zobrazí okno a v ňom zvoľte Visual C++/Test/C++ Unit Test Class.
3. Vhodne pomenujte .cpp súbor, konkrétne podľa metodiky programovacích štandardov.
4. Keď máme .cpp súbor vytvorený, je v ňom vložený vygenerovaný kód s 1 metódou. Naplňte telo tejto metódy a vytvorte tak samotný jednotkový test. Test sa vytvára v jazyku C++, teda pre písanie testov je potrebná znalosť tohto jazyka. Tento proces môže prebiehať viackrát za sebou resp. súčasne. Z toho vyplýva, že sa vytvorí viacero testov naraz. Po napísaní samotného testu/ov sa pristúpi k ich spusteniu.

5. Spustenie unit testov

Vstup: Solution s napísaným testom/i.
Výstup: Otestované jednotlivé časti kódu.
Zodpovedný: Tester.

Popis: Cieľom tohto procesu je spustenie jednotkových testov vo Visual Studiu.

1. V zapnutom vývojovom prostredí, veberte z horného menu možnost „Test“. Následne sa zobrazí menu.
2. V ňom zvoľte možnosť „Windows->Test Explorer“. V pravej časti obrazovky sa zobrazí panel, na ktorom je vidieť zoznam testov.
3. Testy spustíte kliknutím na tlačidlo „Run All“, ktoré sa nachádza na ľavej časti tohto panelu. Potom sa testy vykonajú. Všetky testy ktoré prebehli úspešne sú označené zelenou farbou. Testy ktoré nezbehli, naopak červenou. Samozrejme tento proces sa môže viac krát opakovať.

V prípade, že nechcete spustiť všetky testy máte na výber z viacerých možností. V zobrazenom panely sa nachádza tlačidlo „Run...“. Kliknutím na neho sa zobrazí ďalšie menu, v ktorom je možné spustenie iba nezbehnutých testov, nových testov, úspešne zbehnutých testov a naposledy spustených testov.

Ak nastane situácia, že všetky testy prebehli správne, tak pre tento moment fáza testovania na určitý okamih končí. Avšak iba do okamihu, kedy bude potrebné otestovať ďalšie funkčné jednotky kódu. Kde následne môže vzniknúť problém aj s predtým funkčným testom.

V prípade, že existujú nejaké testy, ktoré nezbehli správne je potrebné pristúpiť k procesu reportovania chýb.

6. Reportovanie chýb

Vstup: Nesprávne fungovanie určitej jednoty kódu – nezbehnutý test.
Výstup: Správa pre programátora.
Zodpovedný: Tester.

Popis: Tento proces nastáva v prípade, že existujú testy, ktoré sa nevykonali správne. Cieľom tohto procesu je odstránenie chýb v kóde. Tester o každom nevykonanom teste vytvorí správu. Vytvorenú správu následne zašle programátorovi.

Správa pre programátora obsahuje:

- Meno odosielateľa
- Nefunkčný test
- Report o chybe
- Dátum do kedy je potrebná oprava

4.3 Metodika - Dokumentácia zdrojových kódov za použitia Doxygen (Manažment zdrojových kódov)

Autor: Lukáš Sekerák

4.3.1 Úvod

Cieľom tejto metodiky je definovať postup, ako správne vygenerovať dokumentáciu z anotácie a v krátkosti popísať danú anotáciu. Metodika je určená pre tímový projekt. Tento projekt sa nazýva „Carlos“ a je napísaný v jazyku C++. Z tohto dôvodu sme pre generovanie zvolili štandardný nástroj doxygen, ktorý je značne populárny pre tento jazyk. Vďaka „doxygenu“ je možné dokumentáciu generovať automatizovane. Metodika je určená pre tvorcov zdrojových kódov, najmä programátorov. Taktiež pre manažéra dokumentácie a manažéra kvality. Manažéri majú za úlohu dohliadať na dodržiavanie tejto metodiky.

4.3.2. Zoznam pojmov

- ➔ **Doxygen** – Štandardný nástroj pre generovanie dokumentácie, vo viacerích programovacích jazykoch. Podporuje C++, C#, PHP, Java, Python. Dokumentáciu dokáže vygenerovať vo formáte PDF, HTML ale aj v iných.
- ➔ **C++** - Programovací jazyk.
- ➔ **Carlos** – Názov nášho tímu, pre ktorý sa vyvíja metodika.
- ➔ **Doxygen wizard** - Pomocný nástroj, ktorý je súčasťou doxygen inštalačného balíka. Nastroj umožňuje použiť doxygen cez jednoduché používateľské prostredie bez znalosti špeciálnych príkazov.

4.3.3. Nadväzujúce metodiky

- ➔ - Metodika prehliadka a kontrola zdrojového kódu
- ➔ - Metodika zdrojových kódov (všeobecne)

4.3.4. Role

Zoznam rolí, pre ktoré je vhodná táto metodika.

Rola	Úloha
Programátor	Má za úlohu kvalitne anotovať triedy, funkcie, všeobecne povedané zdrojový kód. Pri čom má využívať normy a značky používané doxygen nástrojom a zároveň má dodržiavať pravidlá tejto metodiky.
Manažér dokumentácie	Jeho hlavnou úlohou je vytvorenie, teda vygenerovanie dokumentácie. Vygenerovanie ma byť realizované za pomoci doxygen nástroja.
Manažér kvality	Jeho úlohou, je sledovať prácu manažéra dokumentácie a programátorov. Sledovať či si svoje úlohy plnia správne.

*4.3.*5. Procesy

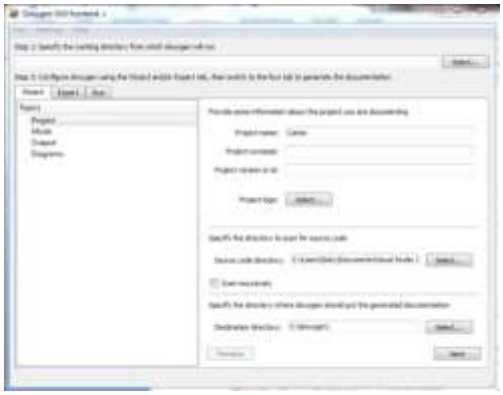
4.3.5.1. Generovanie dokumentácie

Vstup: Kvalitne zdokumentovaný zdrojový kód
Výstup: Vygenerovaná dokumentácia (napríklad v HTML formáte)
Role v procese: Manažér dokumentácie
Opis nástroja: V tomto procese je hlavnou postavou manažér dokumentácie, ktorého úlohou je použiť doxygen wizard, nástroj na generovanie dokumentácie. Program sa skladá z 3 častí:

- **Horná časť** – obsahuje cestu k priečinku, kde je doxygen nainštalovaný
- **Ľavá časť okno so zo zoznamom** – obsahuje zoznam kategórií nastavení
- **Pravá časť okna** – sústava nastavení pre kategóriu

Proces:

1. Spustite doxygen wizard, ktorý nájdete v inštalačnom balíku.
2. Zobrazí sa okno aplikácie, kde je defaultne nastavená kategória „Project“.



Obr. 1 – Kategória nastavenia ,,Project“

2.1. V hornej časti obrazovky nájdite sekciu „Step 1 Specify the working directory from which doxygen will run“. V tejto sekcii kliknite na tlačidlo „Select“. Následne sa zobrazí okno, v ktorom je potrebné nájsť súbor „Doxygen.exe“. Tento súbor sa náchádza v priečinku, do ktorého ste predtým nainštalovali „doxygen“.

2.2. V pravej časti obrazovky nastavte pole s názvom „Project name“ na hodnotu „Carlos“.

2.3. V pravej časti obrazovky nastavte pole s názvom „Project version“ na aktuálnu verziu projektu.

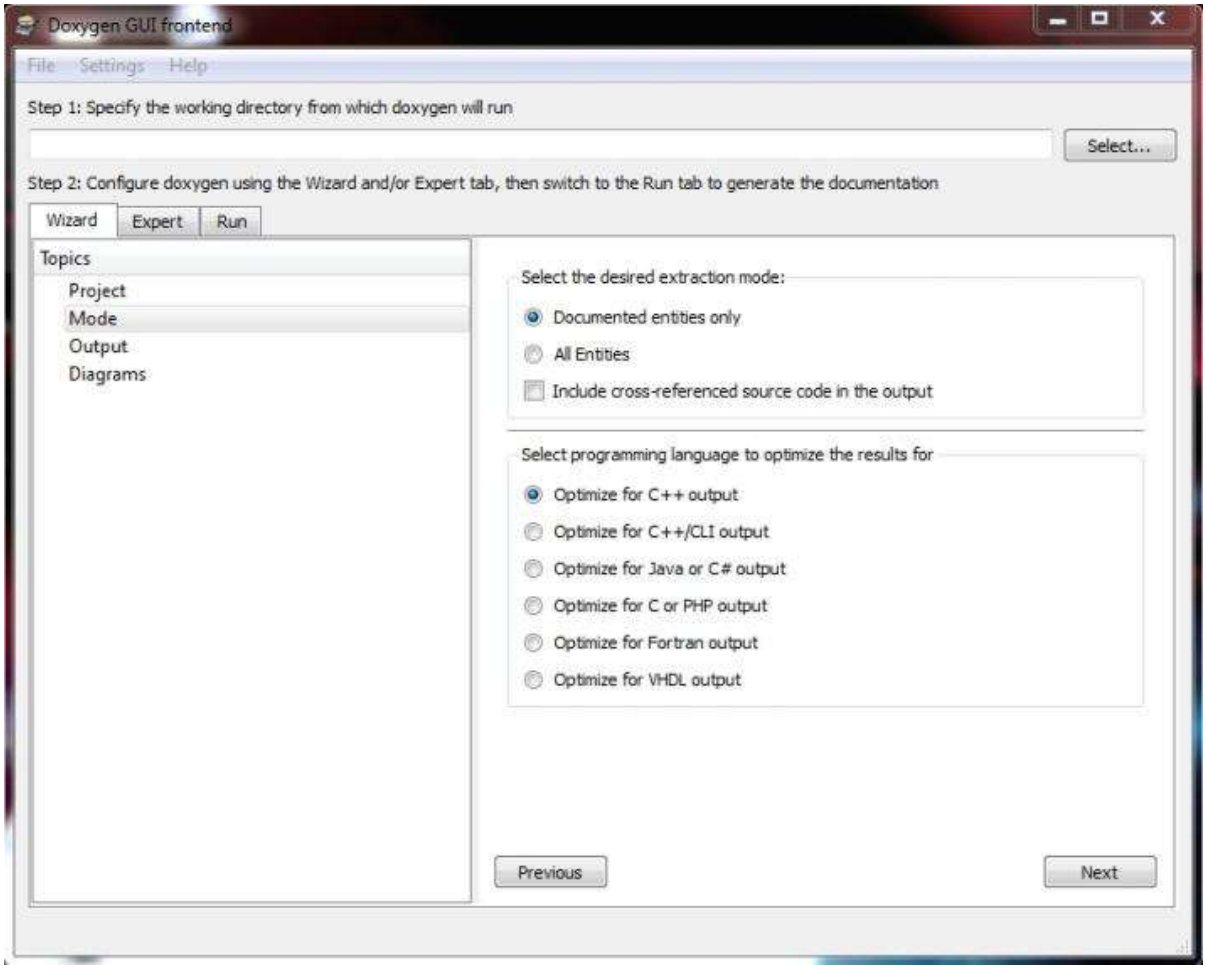
2.4. V pravej časti obrazovky kliknite na „Select...“ tlačidlo, ktoré je v riadku s Project logo. Následne v okne nájdite logo (súbor) pre projekt „Carlos“.
2.5. Pokračujte nájdením položky „Source code directory“, kde kliknite v rovnakom riadku na „Select ...“. Po tejto akcii sa zobrazí okno, v ktorom treba nájsť priečinok ku projektu, pre ktorý, má byť vygenerovaná dokumentácia. Následne potvrďte výber tlačidlom „ok“.

2.6. Pole „Scan recursively“ zaškrtnite, nachádza sa v strede pravej časti okna.

2.7. Následne kliknite na tlačidlo „Select...“, ktoré sa nachádza pri poli „Destination directory“. Opäť sa zobrazí okno, v ktorom je možné prechádzať priečinky v počítači. Zvoľte priečinok, do ktorého sa má výsledná dokumentácia uložiť. Svoj výber opäť potvrďte tlačidlom „ok“.

2.8. Všetky ostatné nastavenia v tomto okne sú voliteľné, pre potreby tímu postačuje ak tieto hodnoty ostanú nezmenené.

3. Kliknite na tlačidlo „Next“ v pravom dolnom rohu okna. Takto sa presuniete na nastavenie pre kategóriu „Mode“.

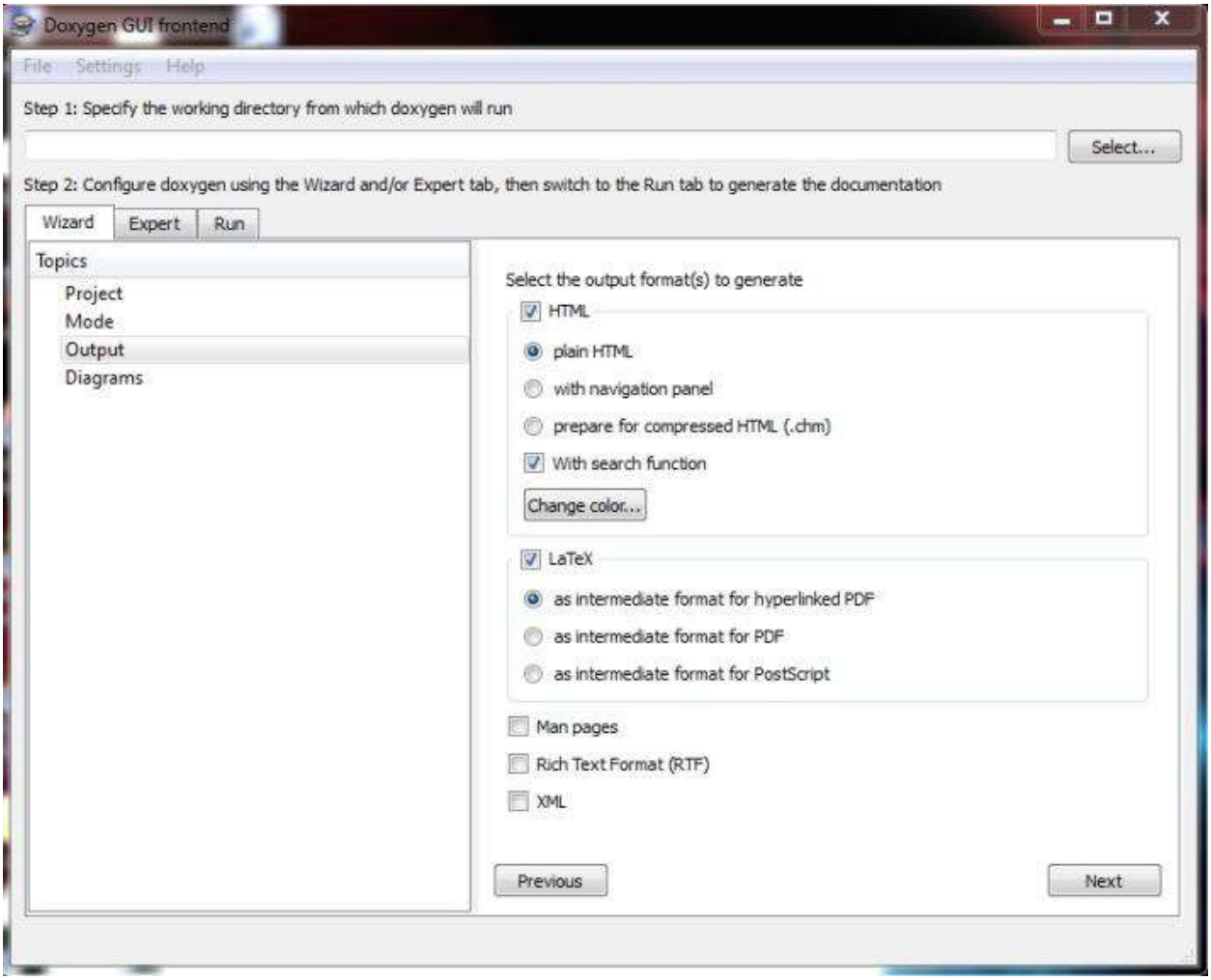


Obr. 2 – Kategória nastavenia „Mode“

3.1. V tomto bode je možné vybrať ku ktorým častiam kódu bude vygenerovaná dokumentácia a teda je tu možnosť výberu. Je možné vygenerovať dokumentáciu ku všetkým entitám v kóde, keď zvolíte „All Entities“ v pravej časti okna. Druhou možnosťou je vygenerovať dokumentáciu len k entitám, ktoré obsahujú anotáciu a to voľbou „Documented entities only“. Odporúčaná možnosť je „Documented entities only“.

3.2. V ďalšej časti vyberte možnosť „Optimus for C++ output“. V prípade iného projektu je potrebné túto možnosť zmeniť na tú, v ktorom programovacom jazyku je projekt napísaný.

4. Kliknite na tlačidlo „Next“ v pravom dolnom rohu aplikácie. Čím sa presuniete na nastavenie kategórie „Output“.

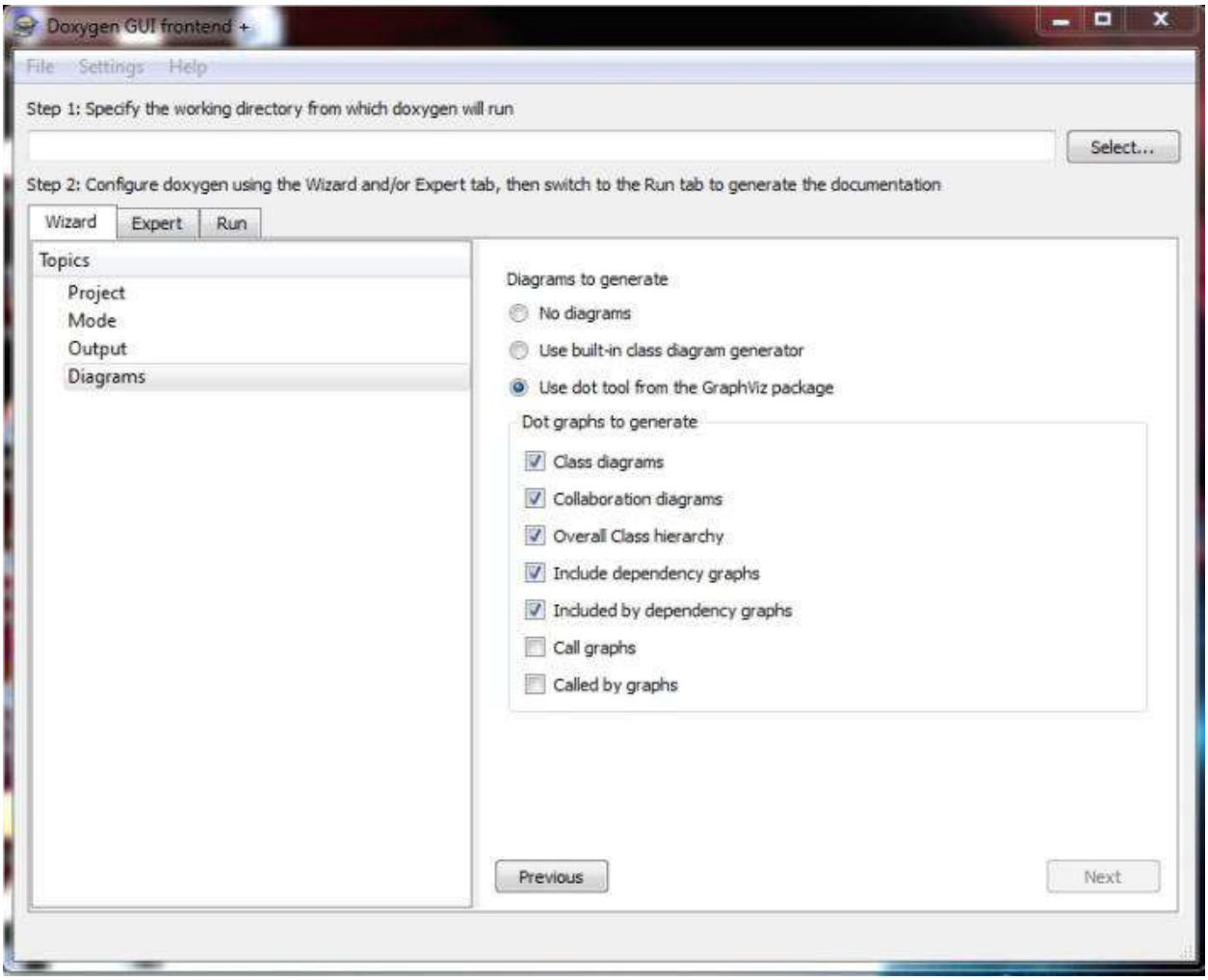


Obr. 3 – Kategória nastavenia Output

4.1. V nastavení kategórie „Output“ je možné nastaviť výstupný formát dokumentácie. Ako najvhodnejší formát pre potreby tímového projektu je formát html. Z toho dôvodu je potrebné zaškrtnúť práve toto políčko.

4.2. Všetky ďalšie nastavenia v tomto okne sú dobrovoľné. Môžete súčasne vygenerovať aj dokumentáciu v iných formátoch napríklad „Latex“. Odporúča sa však nemeniť ostatné hodnoty.

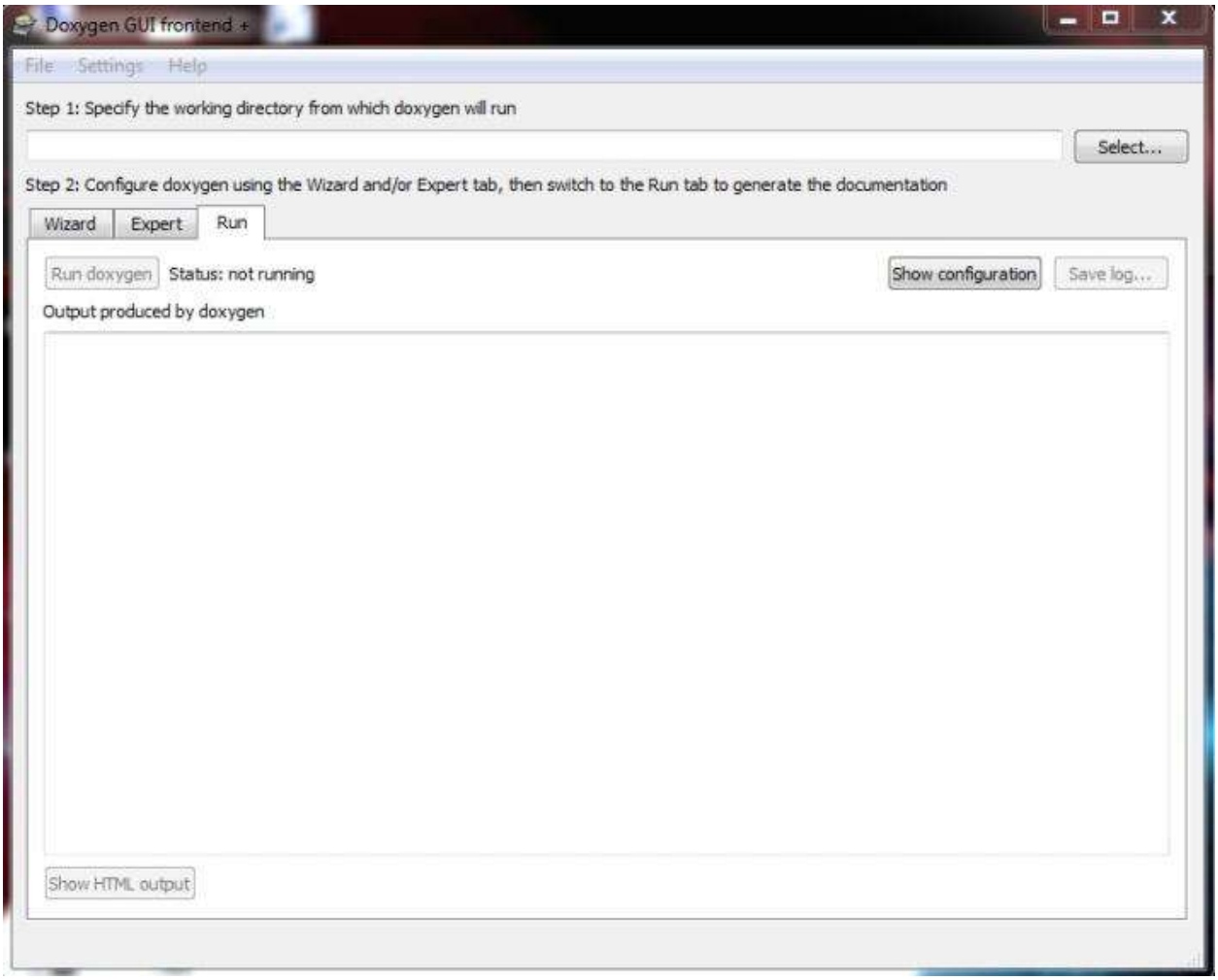
5. Kliknite na tlačidlo „next“ v pravom dolnom rohu. Tým sa presuniete na obrazovku nastavení pre kategóriu „Diagrams“.



Obr. 4 – Kategória nastavenia „Diagrams“

Zaškrtnite voľbu „No diagrams“. V tejto časti sa nachádzajú nastavenia pre generovanie diagramov zo zdrojového kódu. Na generovanie diagramov je potrebný nástroj „ DOT“. Tento nástroj však nie je súčasťou inštalačného balíka. Generovanie diagramov zo zdrojového kódu, je súčasťou metodiky tvorby diagramov zo zdrojového kódu.

6. V hornej časti obrazovky kliknite na tab „Run! Tým sa zobrazí posledná časť. V tejto časti kliknite na „Run doxygen“. Čím sa spustí proces generovania.



Obr. 3 – Tab „Run“.

7. Následne počkajte, kým doxygen vygeneruje dokumentáciu. Skontrolujte v strede okna záznam, či nenastali chyby pri generovaní. V prípade objavenia konfiguračnej chyby, je vašou úlohou zopakovať tento proces. Respektíve opraviť konfiguračnú chybu.

8. Proces končí, dokumentácia je vygenerovaná.

4.4 Metodika zdielania zdrojov v Google Drive

Autor: Martin Petluš

4.4.1 Úvod

Tento dokument popisuje metodiku na dolnej úrovni pri procese zdielania zdrojov (zložiek, súborov) v službe Google Drive. Dokument je vhodný pre projekty a tímy, ktoré vyvíjajú softvérový projekt v malom rozsahu a s počtom ľudí v tíme v rozsahu od 8 do 12.

4.4.1.1 Dedikácia metodiky

Metodikou opisovanom v tomto dokumente sa riadi každý člen tímu.

4.4.1.2 Príbuzné metodiky

- Evidencia úloh v Redmine

4.4.1.3 Role a zodpovednosti

V tabuľke 1 sú vymenované role a ich zodpovednosti, ktoré sa objavujú v tejto popísanej metodike.

Rola	**Zodpovednosť**
Člen tímu	• Študent • Vytvára zdielanú zložku • Vytvára jednotlivé zápisnice v Google Drive • Vytvoriť zápisnicu v stanovenom časovom limite • Je zodpovený za to, aby zápisnica obsahovala všetky potrebné údaje a bola v dohodnutej klavite

Tabuľka 1: Role a zodpovednosti

4.4.1.4 Slovník pojmov

- **Google, Inc.** – je americká spoločnosť zameraná hlavne na internetový trh založená v roku 1998. Spoločnosť založili Larry Page a Sergej Brin. Jej hlavným produktom je internetový vyhľadávač Google.

- **Google Drive** – je služba na ukladanie a synchronizáciu súborov poskytovaná spoločnosťou Google. Táto služba umožňuje ukladanie súborov do cloud-u, zdielanie a kolaboratívne editovanie súborov.
- **Zdielaná zložka** – je taká zložka, ktorá je zdielaná medzi viacerými používateľmi služby Google Drive. Títo používatelia majú rôzne nastavené práva na tejto zložke, ktoré ich obmedzujú v tom čo môžu zo zložkou robiť.
- **Zdielaný súbor** – je taký súbor, ktorý je zdielaný medzi viacerými používateľmi sližby Google Drive a títo používateľa majú rôzne práva pri editovaní tohto súboru.

4.4.2 Zdielanie zdrojov v Google Drive

Táto kapitola sa zaoberá jednotlivými procesmi zdielania zdrojov v službe Google Drive medzi členmi tímu.

4.4.2.1 Vytvorenie zápisnice

Na každom stretnutí tímu k tímovému projektu sa robia poznámky, do ktorých sa zapisuje, ktoré úlohy kto vyriešil, preberajú sa úlohy, ktoré je treba vyriešiť. Tiež sa do poznámok zapisujú úlohy, ktoré sa prideľujú jednotlivým členom tímu. Preberajú sa taktiež rôzne problémy s tímovým projektom, ktoré sa vyskytli, zapisuje sa, kto bol prítomný na stretnutí k tímovému projektu. Tieto poznámky, je treba spracovať do dokumentu, ktorý obsahuje, všetky potrebné náležitosti a je na potrebnej odbornej úrovni. Tento dokument sa volá *Zápisnica* a po jeho vytvorení musia mať k tomuto dokumentu prístup všetci členovia, ak by bolo potrebné spraviť nejaké úpravy samostatne každým členom (spresnenie úlohy, ...).

Proces vytvorenia *Zápisnice* v službe Google Drive (ktorú využívame na zdielanie zápisnice medzi členmi tímu) sa skladá z viacerých podúloh, ktoré nasledujú pri vytvorení prvej zápisnice sekvenčne za sebou. Ak už existuje zdielaná zložka v Google Drive medzi členmi tímu, krok vytvorenia tejto zložky môžeme preskočiť.

4.4.2.1.1 Vytvorenie zdielanej zložky

Vstup: Emailové adresy členov tímu na Google účet
Výstup: Vytvorená zdielaná zložka v Google Drive medzi členmi tímu
Zodpovedný: Člen tímu

Kroky:

1. Prihlásenie sa do služby Google Drive
2. V ľavej časti menu si vyberieme možnosť *My Drive*

3. V ľavej časti obrazovky klikneme na tlačidlo *CREATE*
4. Z ponúknutých možností si vyberieme *Folder*
5. V popupe zadáme zvolené meno zložky (napríklad názov *Zápisnice*)
6. Klikneme na tlačídko *Create*
7. V zozname vecí v našom Google Drive sa nám objaví nová zložka, ktorú sme práve vytvorili
8. Túto novovytvorenú zložku si vyznačíme
9. V hornej časti nad zoznamom vecí v Google Drive vyberieme možnosť *More*
10. V ponúknutom menu vyberieme možnosť *Share* a opäť vyberieme možnosť *Share* na ktorú klikneme
11. V popupe máme možnosť nastaviť viditeľnosť zložky a to tým, že klikneme na tlačidlo *Change*
12. Ponúknuté možnosti zahŕňajú:

Možnosť	**Práva**
Public on the web	Ktokoľvek na Internete bude môcť vidieť túto zložku. Zložka sa môže objaviť vo výsledkoch vyhľadávačov.
Anyone with the link	Ktokoľvek s linkom na túto zložku bude vidieť túto zložku.
Private	Sme jedinou osobou, ktorá má prístup k tomuto dokumentu. Môžeme pridať ďalšie osoby, ktoré budú mať prístup k tejto zložke.

13. Z ponúkaných možností zvolíme prístup *Private*
14. Klikneme na tlačidlo *Save*
15. Nachádzame sa opá v predchádzajúcom popupe
16. Tu v časti *Invite people* máme možnosť pridať používateľov na zdielanie tejto zložky
17. Do textového pola zadáme emailové adresy členov tímu
18. Pridaným členom tímu máme možnosť nastaviť dva typy prístupu k zložke:

Možnosť	**Práva**
Can edit	Používatelia môžu do zložky pridávať veci, mazať veci (dokumenty), pridávať ďalších používateľov (ak je im to povolené), vidieť ostatných používateľov tejto zložky, alebo

	editovať existujúce dokumenty v zložke.
Can view	Používatelia môžu vidieť súbory, zložky, stiahnuť súbory.

19. Vyberieme možnosť *Can edit*, aby ostaní členovia mali možnosť do zložky pridávať svoje vlastné zápisnice, ktorí budú vytvárať
20. Nakoniec klikneme na tlačidlo *Share & save*
21. Posledným krokom, sme vytvorili zdielanú zložku, do ktorej budú môcť ostatní členovia tímu pridávať nové zápisnice

4.4.2.1.2 Vytvorenie zápisnice v zdielanej zložke

Vstup: Poznámky zo stretnutia tímu
Výstup: Vytvorená zápisnica v zdielanej zložke zo stretnutia tímu
Zodpovedný: Člen tímu

Kroky:

1. Prihlásenie sa do služby Google Drive
2. V ľavej časti obrazovky klikneme na tlačidlo *Shared with me*
3. V zozname vecí, ktoré sú so mnou zdielané sa prepneme do zdielanej zložky, v ktorej sa nachádzajú zápisnice
4. Klikneme na tlačidlo *CREATE* v ľavej časti obrazovky
5. Z ponúknutých možností vyberieme možnosť *Document*, na ktorú klikneme
6. V popupe klikneme na tlačidlo *Create & Share*
7. V novom tabe prehliadača sa nám objaví nový dokument
8. V ľavej hornej časti dokumentu klikneme na možnosť *File*
9. Z ponúknutých možností vyberieme možnosť *Rename…*
10. Do textového pola zadáme názov zápisnice v nasledujúcom tvare: *Zapisnica **XX**-**Y***:

Premenná	**Hodnota**
XX	● Naberie hodnotu ZS, ak sa nachádzame v zimnom semestri ● Naberie hodnotu LS, ak sa nachádzame v letnom semestri
Y	● Ak sa jedná o prvú zápisnicu, tak naberá

	hodnotu 1 ● Inak naberá hodnotu, o jedna väčšiu ako je najvyššia hodnota, zo všetkých zápisníc

11. Klikneme na tlačidlo *OK* a týmto novovytvorenú zápisnicu premenujeme
12. Každá zápisnica musí dodržiavať tie isté formátovacie pravidlá (vyzerať tak isto), ako predchádzajúca zápisnica. Pri vytváraní prvej zápisnice, môžeme formát zvoliť ľubovoľne.
13. Každá zápisnica, musí obsahovať hlavičku (prípadne úvodnú stranu), ktorá musí obsahovať tieto údaje:
    - **názov zápisnice** – vo formáte *Zápis Y. stretnutia tímu č. 3*, kde **Y** naberá takú hodnotu, ako je uvedené v tabuľke vyššie
    - **autor zápisu** – kto je autorom tejto zápisnice
    - **dátum** – dátum, kedy sa konalo stretnutie tímového projektu
    - **miestnosť** – miestnosť na škole, v ktorej sa konalo stretnutie tímového projektu
    - **kto je prítomný** – postupne vymenované, ktorý členovia tímu boli na danom stretnutí tímového projektu prítomní na tomto stretnutí
14. Ďalej telo zápisnice musí obsahovať nasledujúce údaje:
    - **priebeh stretnutia** – v krokoch popísané, ktoré sú v časovej postupnosti, popísané čo sa na stretnutí tímového projektu riešilo
    - **tabuľka s pridelenými úlohami –** tabuľka, ktorá obsahuje, id úlohy, meno člena a aká úloha na vypracovanie mu bola pridelená
    - **tabuľku s minulými úlohami –** v akom stave sa nachádzajú úlohy z zadané na predchádzajúcom stretnutí

Po vytvorení zápisnice v zdielanej zložke, a po jej vyplnení obsahom, je teda pripravená na prípadné úpravy druhými členmi tímu. Táto vytvorená zápisnica sa umiestňuje aj na stránku tímu a z tejto zdielanej zložky, ju môže hociktorý člen tímu zobrať a pridať na stránku

4.5 Metodika - Manažment požiadaviek na zmenu

Autor: Róbert Sabol

4.5.1 Úvod

4.5.1.1 Vymedzenie obsahu

Účelom tejto metodiky je definovať a opísať procesy pri zadanej zmene. Tými procesmi je prijatie požiadavky na zmenu, analýza danej požiadavky, implementácia uvedených zmien a následné testovanie.

V metodike sú vymenované a opísané osoby, ktoré sa touto metodikou riadia. V opise sú uvedené ich roly a zodpovednosti v procese zmeny, ktorá vyplýva z požiadavky, ako aj opis súvisiacich procesov, ich vstupov a výstupov.

4.5.1.2 Dedikácia metodiky

Metodika je určená pre osoby, ktoré sa s procesom manažmentu zmien priamo stretávajú. Sú to osoby, ktoré sa zúčastňujú procesu vytvárania požiadavky na zmenu, jej následného spracovania, zanalyzovania, implementácie, ktorá je spojená s testovaním. Záverečnou fázou tohto procesu je odovzdanie zmenenej časti systému. Osoby, ktoré sa tohto procesu zúčastňujú sú uvedené v tabuľke č. 1, v ktorej je aj opis podprocesov, ktorých sa zúčastňujú a tiež činností, z ktorých sa tieto procesy skladajú.

4.5.1.3 Vymedzenie pojmov

- *požiadavka na zmenu* - informácia o žiadosti / požiadavke na funkčnosť, ktorá do momentu uverejnenia požiadavky nebola dohodnutá
- *preberací protokol* - protokol o prebratí zmeny zákazníkom
- *issue* - problém
- *bug* - chyba
- support - označenie požiadavky ako žiadosť o podporu
- *task* - úloha
- *change request* - označenie úlohy ako požiadavky na zmenu
- *unit test* - jednotkový test, slúži na overenie určitej malej časti funkcionality.

- *release* - označenie verzie systému, ktorá musí byť dodaná v stanovený dátum

4.5.1.4 Roly a ich zodpovednosti

Rola	Proces	Činnosť
Zákazník	prijatie požiadavky	- vytvorenie novej požiadavky - kategorizácia požiadavky - pridanie subjektu požiadavky - pridanie opisu požiadavky - pridanie priority požiadavky
	preberacie protokoly o zmene	- podpísanie preberacích protokolov
Projektový manažér	akceptačné testovanie zmeny	- odhad ceny zmeny - vykonanie akceptačných testov - vytvorenie správy o akceptačnom testovaní
	odovzdanie zmeny	- odovzdanie zmeny zákazníkovi
	preberacie protokoly o zmene	- vytvorenie preberacieho protokolu - vyplnenie preberacieho protokolu
Projektový analytik	analýza požiadavky na zmenu	- analýza rozsahu zmeny - odhad času zmeny - analýza zmenených modulov
Programátor	implementácia zmeny	- implementácia zmeny - testovanie počas implementácie
	odovzdanie zmeny	- odovzdanie funkčnej zmeny
Tester	testovanie zmeny	- testovanie zmeny - unit testy

Tabuľka č. 1

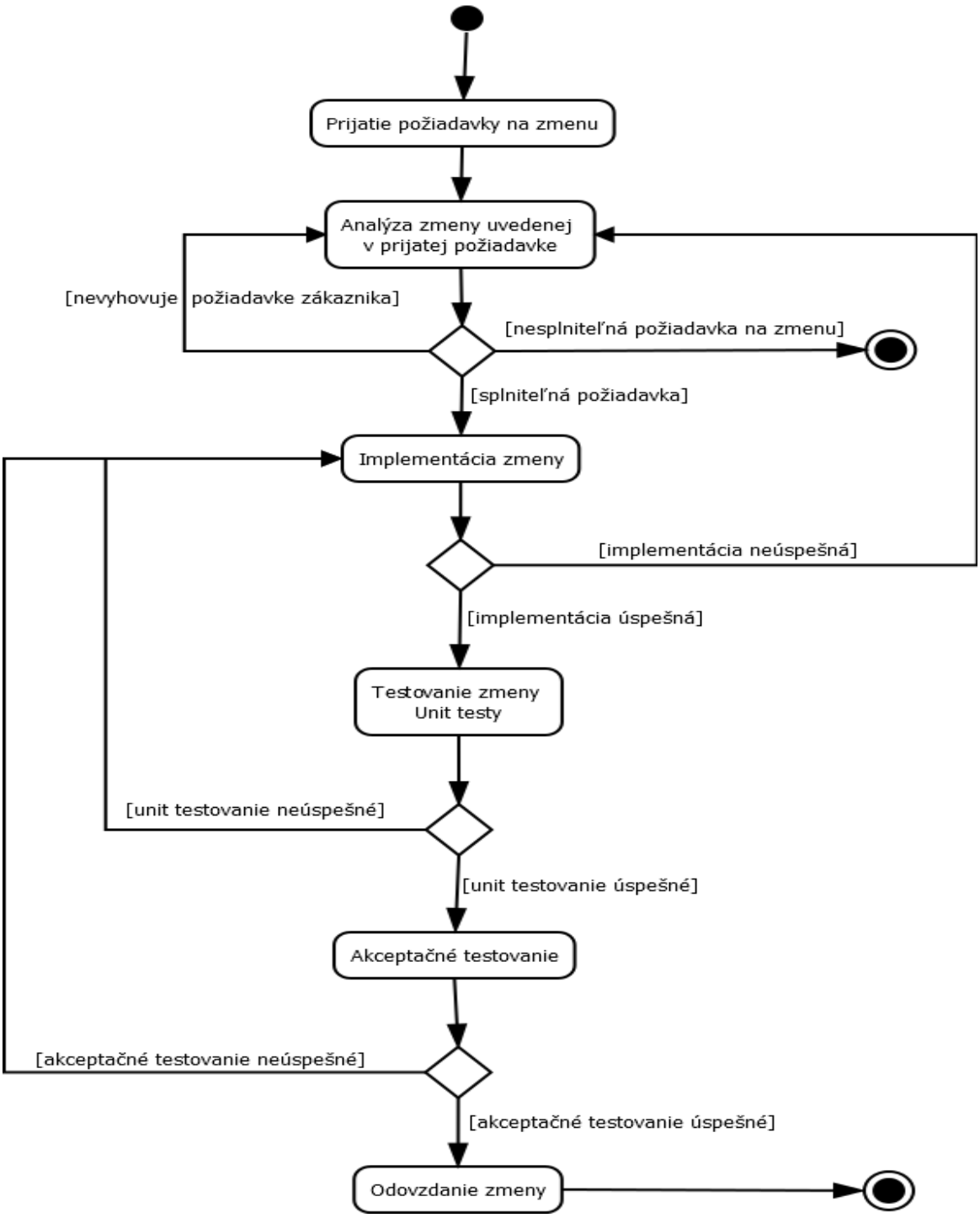
4.5.1.5 Nadväzujúce metodiky

- metodika tvorby analýzy

- metodika unit testovania
- metodika riadenia úloh
- metodika plánovania

4.5.2 Opis procesov

Na nasledujúcom obrázku obr. 1 je zobrazený diagram činností, ktoré sa vykonávajú počas pocesu zmeny v projekte od prijatia požiadavky na zmenu až po odovzdanie zmeny.



Obrázok č. 1

4.5.2.1 Prijatie požiadavky na zmenu

Vstup: Požiadavka na zmenu
Výstup: Záznam o požiadavke v systéme Redmine
Zodpovedná osoba: Zákazník

Popis: Hlavným ťažiskom tejto činnosti je získať požiadavku na zmenu od zákazníka. Vstupom ako je uvedené v štruktúrovanom prehľade je samotná požiadavka na zmenu od zákazníka. Výstupom je zaevidovaný záznam o žiadosti v systéme Redmine. Vytvorený záznam musí mať vyplnené všetky potrebné polia, ktoré sú uvedené v tabuľke č.2, v ktorej je v krokoch opísaný proces prijímania požiadavky. Po tomto procese je požiadavka úspešne zaevidovaná.

Krok	Opis kroku
1	Vytvorenie novej požiadavky
2	Pridanie kategórie pre požiadavku
3	Pridanie subjektu pre požiadavku
4	Pridanie opisu požiadavky
5	Pridanie priority požiadavky

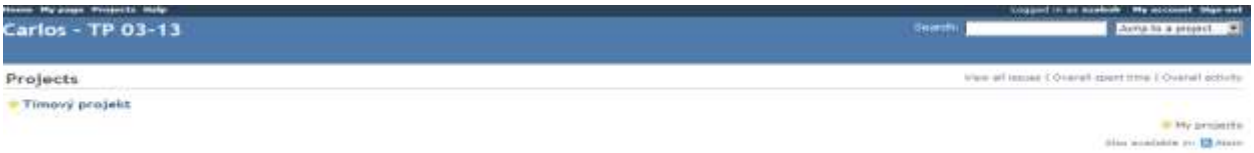
Tabuľka č. 2

4.5.2.1.1 Vytvorenie novej požiadavky

Vytvorenie požiadavky sa vykonáva v systéme Redmine. Dôležitou podmienkou v tomto kroku je to, že zákazník musí byť v systéme registrovaný.

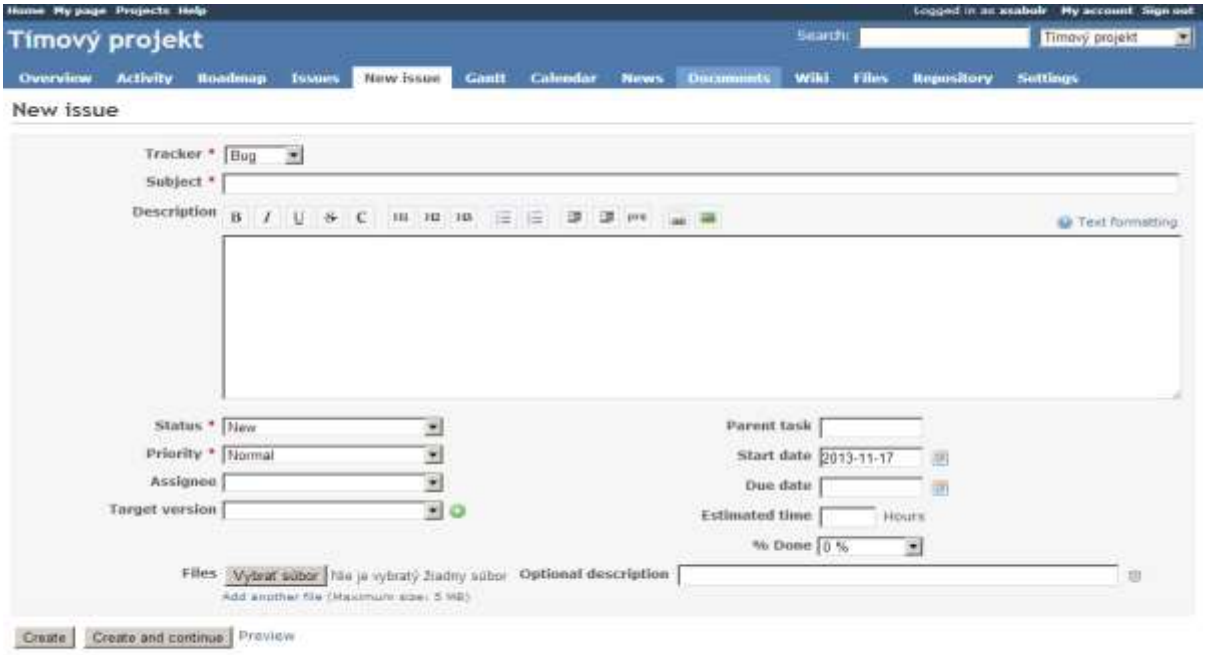
Priebeh činností zákazníka v systéme Redmine:

- vyberie projekt, v ktorom chcem zaevidovať požiadavku, kliknutím na názov projektu (Obr. č. 2.)
- na ďalšej obrazovke (Obr. č. 3), na ktorej je prehľad daného projektu, klikne na položku “New issue” v hornom menu tejto obrazovky
- následne sa zobrazí obrazovka s formulárom (Obr. č. 4), ktorý zákazník výplní



Obrázok č. 2

Obrázok č. 3



Obrázok č. 4

4.5.2.1.2 Pridanie kategórie pre požiadavku

Požiadavke je potrebné priradiť kategóriu z týchto 4 typov:

- chyba (bug)
- podpora (support)
- úloha (task)
- požiadavka na zmenu (change request)

Priebeh činností zákazníka v systéme Redmine:

- v rozbaľovači “Tracker” zvolí možnosť “požiadavka na zmenu” (change request)

4.5.2.1.3 Pridanie subjektu pre požiadavku

Subjekt (predmet) by mal byť čo najvýstižnejší, aby bolo možné požiadavku ľahko identifikovať a preto je vhodné zvoliť takýto formát.

[<Modul v systéme>] <Názov>

Napríklad:

[Architektúra] Spracovanie správ z Androidu

Hodnoty, ktoré môžu jednotlivé polia obsahovať sú uvedené v nasledujúcej tabuľke (Tab. č. 3).

Modul v systéme	Architektúra Kinect Mobil OpenCV Poloha Dokumentácia
Názov	opis požiadavky bez diakritiky v rozsahu max. 10 slov

Tabuľka č. 3

4.5.2.1.4 Pridanie opisu požiadavky

Opis požiadavky na zmenu musí byť presný a výstižný.

Obsahuje:

- modul, ktorého sa zmena týka
- opis zmeny
- dôvod zmeny

4.5.2.1.5 Pridanie priority požiadavky

Dôležitým pre pridanie novej požiadavky je zadať prioritu. Prioritu vyberieme z rozbaľovača, ktorý je vo formulári pod opisom na ľavej strane.

Môžeme vyberať z 5 stupňov priority:

- Low (nízka)
- Normal (normálna)
- High (vysoká)
- Urgent (urgentná)
- Immediate (okamžitá)

Najzávažnejšie sú požiadavky s prioritou “Urgent” a “Immediate”, ktoré je nutné riešiť v aktuálnom release resp. okamžite, aby sa mohlo pokračovať v projekte.

4.5.2.2 Analýza zmeny uvedenej v prijatej požiadavke

Vstup: Záznam o požiadavke v systéme Redmine
Výstup: Dokument s analýzou zmeny s časovým a finančným odhadom
Zodpovedná osoba: Projektový analytik

Popis: Podstatou tejto činnosti je analýza požiadavky na zmenu, ktorá je už riadne evidovaná v systéme Redmine. Po zanalyzovaní sa rozhodne, či sa nejedná o požiadavku, ktorá je v absolútnom rozpore s analýzou. V takomto prípade ide o nepochopenie systému a požiadavka je následne zamietnutá. Ak zákazník aj po zamietnutí požiadavky stále trvá na tejto zmene, požiadavka sa mu vráti na modifikovanie tak, aby nebola v rozpore so súčasnou analýzou.

Požiadavka môže byť tiež označená ako chyba - to sa stane v prípade ak je nejaká časť systému zle implementovaná. Požiadavka sa v tomto prípade označí ako chyba a je odoslaná osobe, ktorá je zodpovedná za túto implementáciu.
Existuje ešte jeden prípad, kedy sa žiada o zmenu už implementovanej časti. V tomto prípade sa na základe analýzy rozhodne o aký zásah do systému ide. Prebehne analýza rozsahu požadovaných zmien, časový a finančný odhad. K tomuto odhadu sa môže vyjadriť zákazník v systéme Redmine.
Požiadavka s analýzou je následne posunutá programátorom. V prípade nesúhlasu s finančným odhadom zo strany zákazníka je požiadavka zamietnutá poprípade zaslaná na prehodnotenie zákazníkovi.

4.5.2.3 Implementácia zmeny

Vstup: Záznam o požiadavke v systéme Redmine, Analýza požiadavky na zmenu
Výstup: Implementovaný modul, ktorý spĺňa požiadavku na zmenu
Zodpovedná osoba: Programátor

Popis: Hlavnou úlohou tejto činnosti je implementovať funkcionalitu, ktorá bola uvedená v požiadavke na zmenu. Modul, v ktorom sa táto implementácia vykonávala je po ukončení implementácie posunutý testerom na vykonanie unit testovania. Vývoj daného modulu má na starosti jeden z programátorov - senior developer. Ak nastane prípad, že sa modul na základe analýzy nedá implementovať, je spolu s opisom problému zaslaný spať projektovému analytikovi.

4.5.2.4 Testovanie zmeny - Unit testy

Vstup: Implementovaný modul
Výstup: Správa o úspešnom unit testovaní
Zodpovedná osoba: Tester

Popis: Podstatou tejto činnosti je otestovanie funkčnosti implementovanej zmeny a to tak, aby sa na základe tejto zmeny nenarušila spolupráca s ostatnými modulmi v systéme. Pomocou unit testov sa zistí, či zmenený modul dokáže správne fungovať v systéme a spolupracovať s ostatnými časťami systému. Ak nastane prípad, že modul neprejde niektorým z unit

testov, tak je vrátený na doimplementovanie. Výstupom testovania je správa o úspešnosti unit testov, ktorá je následne prílohou pre akceptačné testovanie. Unit testovanie sa vykonáva na základe metodiky o unit testovaní.

4.5.2.5 Akceptačné testovanie

Vstup: Správa o úspešnosti unit testovania, akceptačné testy
Výstup: Správa o úspešnom akceptačnom testovaní
Zodpovedná osoba: Projektový manažér

Popis: Hlavným ťažiskom tejto činnosti je vykonanie akceptačných testov projektovým manažérom, ktorý tak učiní na základe vykonanej analýzy. Na základe toho, či zmena prejde akceptačným testovaním je aj rozhodnutie, či sa môže zmena odovzdávať a prejsť k podpísaniu preberacích protokolov. Prílohou k akceptačným testom je aj správa o unit testovaní, ktorá je technickým dôkazom o funkčnosti systému po zahrnutí požadovanej zmeny. Ak nastane zlyhanie akceptačných testov, tak projektový manažér odošle správu o zlyhaní projektovému analytikovi, aby zistil príčinu zlyhania a napravil to.

4.5.2.6 Odovzdanie zmeny

Vstup: Správa o úspešnosti akceptačného testovania
Výstup: Podpísané preberacie protokoly
Zodpovedná osoba: Projektový manažér, zákazník

Popis: V tejto činnosti je na základe úspešného akceptačného testovania podpísaný preberací protokol o implementovanej zmene. V preberacom protokole je uvedený aj dodatok o nasadení zmeny v konkrétnom release, podmienky financovania, prípadne poznámky. Zmenený modul je odoslaný zodpovednej osobe, ktorá sa postará o nasadenie v danom termíne.

4.6 Metodika plánovania úloh

Autor: Jakub Mercz

4.6.1 Úvod

Cieľom tejto metodiky je definovať presné pravidlá a postupy pre plánovanie úloh a ich následné zaznamenania na stretnutiach k tímovému projektu, ktorých sa zúčastňuje aj product owner. Tieto stretnutia majú okrem iného za úlohu zhodnotenie vypracovania predchádzajúcich úloh a naplánovanie úloh ďalších.

Metodika je určená pre všetkých členov tímu, špeciálne postavenie má člen aktuálne poverený písaním zápisnice.

4.6.1.1 Použité pojmy

- *Product owner* – pozícia definovaná v scrum-e, pre účely tohto projektu je ním vedúca projektu

4.6.1.2 Použité skratky

- *PO* – Product owner

4.6.1.3 Nadväzujúce metodiky

- Metodika evidencie nových úloh v Redmine

4.6.1.4 Role

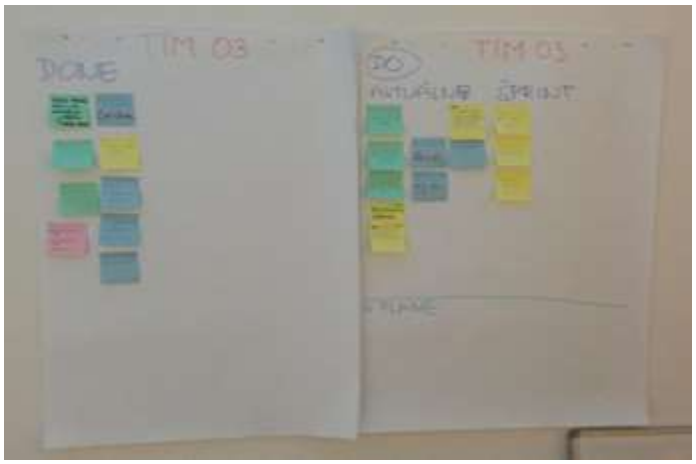
Rola	Popis	Zodpovednosť
Člen tímu	-	prezentácia výsledkov, identifikácia úloh, premiestňovanie úloh na tabuli
Zapisovateľ	Člen tímu aktuálne poverený písaním zápisnice	zapisovanie úloh a ich stavov do zápisnice

4.6.1.5 Nástroje

1) Softvérové

a) Textový editor podľa uváženia zapisovateľa

2) Fyzické
   a) Veľké papiere rozdelené na zóny, zavesené v Laboratóriu počítačového videnia a počítačovej grafiky, ktoré slúžia ako tabuľa
   b) Prilepovacie lístočky a fixy rôznych farieb



Obrázok 1 - *Fyzické nástroje*

4.6.2 Procesy

4.6.2.1 Dokončenie úlohy

Vstup: výsledky práce medzi stretnutiami
Výstup: zaznamenaný aktuálny stav úlohy na tabuli a v tabuľke v zápisnici
Zodpovedný: člen tímu, zapisovateľ

Popis:
Postupne v poradí určenom PO prezentuje každý člen tímu svoje výsledky od posledného stretnutia a na základe diskusie s PO a rozhodnutia PO vyhodnocuje úlohu ako dokončenú.

1. Zapisovateľ si pripraví tabuľku vychádzajúc z tabuľky pridelených úloh, ktorá sa nachádza v zápisnici z predchádzajúceho stretnutia. Do tejto tabuľky pripojí ďalší stĺpec s názvom „Stav“.
2. Člen tímu povie názov pridelenej úlohy z predchádzajúceho stretnutia.
3. Člen tímu opíše čo najpodrobnejšie postup a výstup jeho snaženia od posledného stretnutia.
   a. Ak ho k tomu vyzve PO, člen tímu vizuálne odprezentuje aktuálny stav jeho modulu.
4. Člen tímu povie koľko času strávil danou úlohou.
5. Člen tímu vedie diskusiu s PO, či je úloha splnená alebo je nutná ďalšia práca na danej úlohe.

a. Ak bola úloha splnená

- Člen tímu nájde prilepovací papierik s danou úlohou a premiestni ho do sekcie DONE na tabuli.
- Zapisovateľ zapíše do tabuľky ako stav úlohy „vyriešené“

b. Ak úloha nebola splnená

- Zapisovateľ zapíše do tabuľky ako stav úlohy „v procese riešenia“
- Ak nastali špecifické udalosti týkajúce sa riešenia danej úlohy, zapisovateľ ich tiež popíše do stĺpca Stav

6. V prípade, že člen tímu má ďalšiu úlohu, ktorá ešte nebola spomenutá, pokračuje bodom 2 s nespomenutou úlohou.
7. Po prejdení všetkých úloh aktuálneho člena tímu, PO vyzve ďalšieho člena tímu k reprezentácií svojich výsledkov.

Keď všetci členovia tímu prejdú týmto procesom, zapisovateľ by mal mať hotovú tabuľku, ktorá sa podobá na *Tabuľku 1*.

ID	**Pridelené členovi**	**Opis úlohy**	**Stav**
1	Martin	študovať OpenCV, vytvoriť prvú verziu webovej stránky tímu	vyriešené
2	Lukáš	pozrieť sa na architektúru aplikácie podrobnejšie, nainštalovať databázu	v procese riešenia, nemohol nainštalovať kvôli nefunkčnosti serveru
3	Patrik	porovnávať rôzne metódy detegovania objektov, na nejakej snímke skúšať detegovanie objektov, vyskúšať či je lepšie hľadať objekt nanovo na každom snímku, ale sledovať objekt	vyriešené
4	Robo	skúsiť ukladať gestá do lokálnej	v procese riešenia,

		databázy a skúsiť odoslať informácie o gestách na lokálny server	nemal potrebný HW
5	Marianna a Peťo	preštudovať si OpenGL, pracovať na nejakej jednoduchšej úlohe (napríklad vypisovať texty na rôzne miesta na obrazovke, pracovať s nejakým 3D modelom, dopĺňať text do obrázku)	vyriešené
6	Jakub	nainštaluje knižnicu pre prácu s Kinnectom a spraví experimenty na snímanie hlavy z rôznych pohľadov Kinnectu	vyriešené
7	Juraj	doriešiť aplikáciu na záznam GPS, pozrieť sa spolu s Robom na ukladanie gest v lokálnej databáze	vyriešené
8	všetci členovia	každý si podrobnejšie premyslí nejaký druh hry (napríklad zábavného typu, vzdelávacieho typu, atd.), najmä také, ktoré dokážu zabaviť deti	vyriešené

Tabuľka 1 - *Príklad tabuľky stavu úloh v zápisnici*

4.6.2.2 Zaznamenanie novej úlohy

Vstup: diskusia o príbehu aktuálneho šprintu
Výstup: zaznamenané nové úlohy na tabuli a v tabuľke v zápisnici
Zodpovedný: člen tímu, zapisovateľ

Popis:
Po diskusii členov tímu a PO o príbehoch aktuálneho šprintu (práve prebiehajúci alebo začínajúci šprint) jednotlivý členovia tímu identifikujú úlohy pre svoj modul a po diskusii s PO ich zaznamenajú.

1. Zapisovateľ si pripraví tabuľku pridelených úloh so stĺpcami „ID“, „Pridelené členovi“ a „Opis úlohy“

2. Každý člen tímu si zoberie farebné papieriky a fixy s farbou, ktorá mu bola pridelená na predchádzajúcich stretnutiach.
3. Každí člen tímu presunie svoje papieriky so sekcie V PLÁNE do sekcie AKTUÁLNY ŠPRINT.
4. Člen tímu s identifikovanými úlohami ich postupne odprezentuje PO.
    a. Ak bola úloha schválená PO a bola PO určená ako prioritná:
        - Člen tímu napíše názov úlohy a svoje iniciálky na papierik a prilepí ho do sekcie AKTUÁLNY ŠPRINT.
    b. Ak bola úloha schválená PO a bola PO určená ako sekundárna:
        - V prípade že odhadovaný čas vypracovania úloh pre člena tímu v sekcii AKTUÁLNY ŠPRINT plus prejedávanej úlohy je pod 8 hodín, člen tímu napíše názov úlohy a svoje iniciálky na papierik a prilepí ho do sekcie AKTUÁLNY ŠPRINT.
        - V prípade že odhadovaný čas vypracovania úloh pre člena tímu v sekcii AKTUÁLNY ŠPRINT plus prejedávanej úlohy je nad 8 hodín, člen tímu napíše názov úlohy a svoje iniciálky na papierik a prilepí ho do sekcie V PLÁNE.
5. Zapisovateľ zapíše krstné meno člena tímu prezentujúceho svoje úlohy a do stĺpca opis úlohy zapíše jeho úlohy podľa tabule. V stĺpci „ID“ dá inkrementovanú hodnotu z predchádzajúceho riadku (v prvom údajovom riadku napíše číslo 1).
6. Ak člen tímu odprezentoval všetky svoje identifikované úlohy, pokračuje ďalší člen tímu od bodu 4
7. Ak všetci členovia tímu odprezentovali svoje identifikované úlohy, zapisovateľ zapíše do posledného riadku do stĺpca „Pridelené členovi“ text „všetci členovia“ a úlohu počas diskusie identifikovanú ako všeobecnú a potrebnú pre diskusiu na ďalšom stretnutí alebo inak potrebnú pre daľšiu prácu niektorého člena.

Na konci procesu by mal zapisovateľ mať tabuľku podobnú *Tabuľke 1*, ale bez stĺpca „Stav“.

4.7 Metodika prehliadky kódov vo dvojiciach

Autor: Marianna Mušinská

4.7.1 Úvod

Účelom tejto metodiky je definovanie procesu prehliadky kódov pri vytváraní aplikácií. Za úlohu má taktiež predstaviť postup prehliadky kódov pomocou nástroja Redmine. Metodika je zameraná pre malé a stredné tímy pracujúce v jazyku C++.
Prehliadka kódov prispieva k vytváraniu kvalitných kódov. Keď od začiatku používame prehliadky kódov, tak zabezpečujeme aj nižšie náklady. Pomáha predchádzať, odhaľovať, naprávať a predchádzať chybám, zachováva konzistentnosť, zvyšuje kvalitu kódu.
K prvým krokom k dobrej prehliadky kódov je potrebné zadefinovať dobrý nástroj, ktorý pomáha pri prehliadkach kódov.

Táto kapitola sa zaoberá opisom jednotlivých rolí a opisom procesov prehliadok kódov.

Roly a zodpovednosti

Manažér kvality

- kontroluje kvalitu kódov
- určuje prehliadky v jednotlivých častiach zdrojových kódov
- režíruje časové obmedzenia na prehliadky
- je zodpovedný za to, aby zdrojové kódy boli čo najkvalitnejšie

Vývojár

- vytvára zdrojový kód
- vysvetľuje jednotlivé časti kódu, odôvodňuje prečo boli zvolené jednotlivé štýly programovania
- značkuje jednotlivé časti kódu

v druhej fáze:

- číta jednotlivé značkovania
- vytvára opravy v zdrojovom kóde
- upozorňuje a opravuje chyby v kóde
- pomáha v spísaní zápisnice

Dokumentarista

- Je zodpovedný za vytváranie zápisnice počas stretnutia
- Je zodpovedný za výstupnú zápisnicu prehliadky

Administrátor

- príprava a inštalácia vyžadovaných nástrojov

4.1.2 Procesy

Proces plánovanie prehliadky

Vstup:

- Vývojári
- Zdrojové kódy

Výstup:

- časový postup plánovania kontrol
- plán predstavenia zdrojového kódu

Rola :

- vývojári, dokumentarista, manažér kvality

Plánovanie prehliadky určuje, aké zdrojové kódy má vytvárať vývojár v akom časovom rozsahu a s kým bude spolupracovať v párovej prehliadke.

Cieľom tohto procesu je naplánovať prehliadky kódov, rozdeliť členov tímu do jednotlivých párov, ktorí sa budú navzájom kontrolovať. Taktiež sa musí naplánovať časový postup prehliadok, aby nevznikli prípady kolízie, kde jeden z dvojice vývojárov bude musieť čakať na druhého, ktorý sa nezúčastňuje dostatočne aktívne procesu.

Proces značkovanie kódu

Vstup:

- zdrojový kód

Výstup:

- značkovaná zdrojový kód

Rola:

- vývojári

Vývojár za cieľom dobrej orientácie v zdrojovom kóde označkuje svoje dielo. Značkovanie sa odohráva počas príslušného šprintu jedným z vývojárom. Cieľom značiek je uľahčiť spoluprácu pri párovom programovaní.

Proces vývoja zdrojového kódu

Vstup:

- počiatočná/aktuálna časť zdrojového kódu dokumentáciou a so značkovaním
- požiadavky na zdrojový kód

Výstup:

- zdrojový kód s možnými chybami

Rola :

- vývojári

Vývojár vytvára zdrojový kód, s dokumentáciou a značkovaním počas šprintu. Prípadne keď už má k dispozícií počiatočný zdrojový kód, postupne pokračuje v vývoji.

Proces prehliadky

Vstup:

- počiatočná/aktuálna časť zdrojového kódu s dokumentáciou a so značkovaním
- požiadavky na zdrojový kód
- plán vykonávania prehliadky

Výstup:

- počiatočná/aktuálna časť zdrojového kódu s dokumentáciou a so značkovaním a identifikovanými chybami a anomáliami

Rola:

- vývojári

Podľa vopred určeného harmonogramu jeden z dvojice, ktorý sa nezúčastnil na vývoji zdrojového kódu skontroluje kód podľa dokumentácie a s pomocou značkovania. Pri nájdení chyby označuje a upozorňuje na chyby.

Proces opravy chýb

Vstup:

- zoznam chýb a anomálií identifikovanými počas prehliadky kódu
- zdrojové kódy so súvisiacou dokumentáciou a značkovaním

Výstup:

- opravené a funkčné zdrojové kódy
- odstránené anomálie

Rola:

- vývojári

4.8 Metodika - Manažment verzií zdrojového kódu (nástroj Git)

Autor: Juraj Jarábek

4.8.1 Úvod

Táto metodika má na starosť opísať postupy spojené s procesmi manažmentu verzií zdrojového kódu. Je určená pre menší vývojový team, konkrétne team s počtom členov 8. Metodika sa zameriava na nižšiu úroveň manažmentu verzii zdrojového kódu. Táto úroveň opisuje jednotlivé kroky, resp. konkrétnu činnosť pri používaní nástroju Git, ktorý používa spomenutý tým pri verziovaní.

4.8.2. Zoznam pojmov

- **Git** – Git je distribuovaný systém riadenia revízií, ktorý vytvoril Linus Torvalds. Nemal by sa zamieňať s programom GIT (GNU Interactive Tools), správcom súborov na spôsob programu Norton Commander, ktorý vytvorili Tudor Hulubei a Andrei Pitis.
- **Git Extension** -Git Extension je jediné grafické používateľské prostredie pre Git, ktoré umožňuje kontrolovať Git bez použitia príkazovej riadky. Samotný program obsahuje aj manuál, video tutoriály pre rýchlu orientáciu.
- **Repozitár** - Z angl. "repository", ktoré znamená: schránka, úschovňa, úložisko, skladisko, zdroj. Pre repozitár sa tiež používá označenie "installation source". Jedná sa o miesto (napr. server alebo lokálny počítač to znamená vzdialené a lokálne úložiská), kde sú uložené zdrojové kódy projektu.
- **Vetva** - Predstavuje odklonenie vo vývoji od hlavnej, tzv. master vetvy. Hlavná vetva spravidla obsahuje len otestovaný, funkčný a stabilný kód a každá nová funkcionalita sa vyvíja a testuje v samostatnej vetve.
- **Commit** – Odoslanie zmien z lokálneho úložiska na server
- **Merge** - Po slovensky: zlúčiť, splynúť, spojiť. Konkrétne pri používaní v nástroji Git sa jedná o zlúčenie rôznych vývojových vetiev, zahŕňa riešenie konfliktov medzi rôznymi verziami toho istého súboru.
- **Konflikt**: Vzniká, ak bol rovnaký súbor modifikovaný dvoma rôznymi užívateľmi.
- Rebase: Jedná sa o process získania zmien, ktoré boli vykonané na inej vetve, za účelom aktualizácie aktuálnej vetvy.
- **Revision**: Predstavuje aktuálnej verziu projektu v repozitári, alebo tiež počet komitov do repozitára. Pri vytvorení je nastavené na 0, každý komit ho inkrementuje o 1.

4.8.3. Nadväzujúce metodiky

- ➔ Metodika: Prehliadka a kontrola zdrojového kódu
- ➔ Metodika zdrojových kódov (všeobecne)
- ➔ Metodika: Manažment zmien a požiadaviek na zmenu aplikácie
- ➔ Metodika: Manažment chýb

4.8.4. Role

Zoznam rolí pre ktoré je vhodná táto metodika.

Rola	Úloha
Manažér kvality	Jeho úlohou je vytvorenie / inštalácia repozitára (úložiska). Ďalej konfigurácia úložiska a jeho úložiska
Manažér podpory vývoja	Jeho úlohou je najmä testovanie a hodnotenie kvality,
Manžér rizík	Má na starosti riešenie konfliktov
Programátor	Má na starosti vývoj nových funkcionalít na svojom lokálnom úložisku , opravu chýb a komitovanie zmien na server

*4.8.*5. Procesy

4.8.5.1 Inštalácia nástroja Git Extensions

Vstup: Nakonfigurované vývojové prostredie na serveri
Výstup: Nainštalovaný nástroj na manažment verzií – Git Extensions
Zodpovednosť: Manažér kvality, vývojár

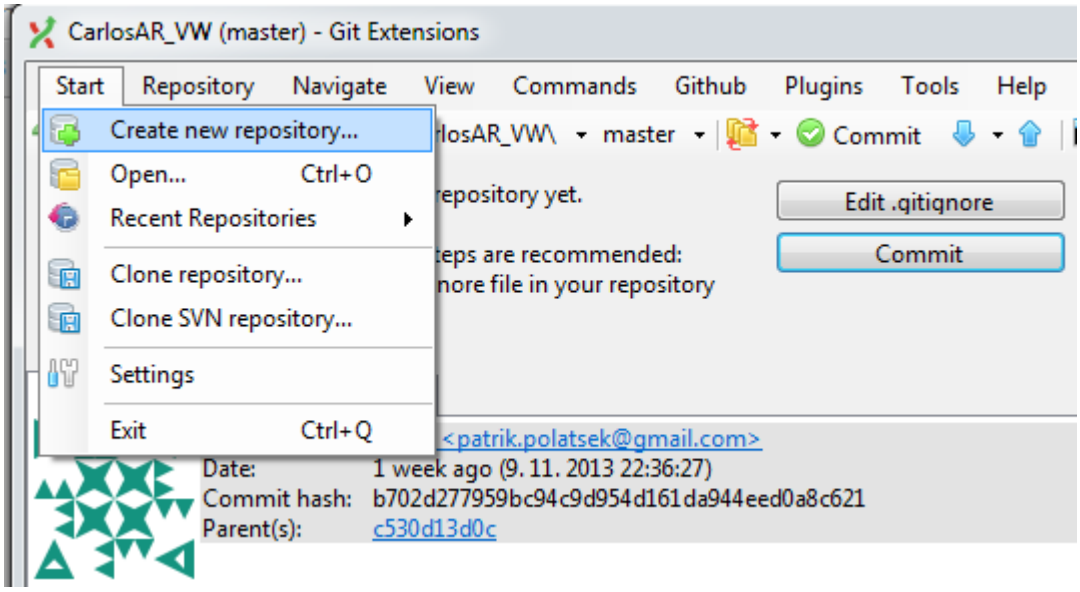
Primárnu zodpovednosť za proces inštalácie má manažér podpory vývoja, ktorý inštaluje nástroj na manažment verzií na serveri. Vývojári zodpovedajú za inštaláciu nástroja na ich lokálnom počítači, splnenie úlohy kontroluje opäť manažér kvality.

4.2 Konfigurácia nástroja Git Extensions

Vstup: Nainštalovaný nástroj na manažment verzií (Git Extensions)
Výstup: Nakonfigurované centrálne úložisko na server
Zodpovednosť: manažér kvality, programátor

Primárnu zodpovednosť za proces konfigurácie má manažér kvality, ktorý konfiguruje centrálne úložisko a prístup k nemu. Vývojár konfiguruje lokálne úložisko.

Preto môžeme tento proces rozdeliť do štyroch hlavných krokov:
1. Vytvorenie centrálneho úložiska
2. Vytvorenie prípravnej (develop) vetvy
3. Nastavenie prístupových práv používateľov k centrálnemu úložisku
4. Konfigurácia lokálneho úložiska



Obr1.Vytvorenie nového úložiska (Repository) v nástroji GitExtensions

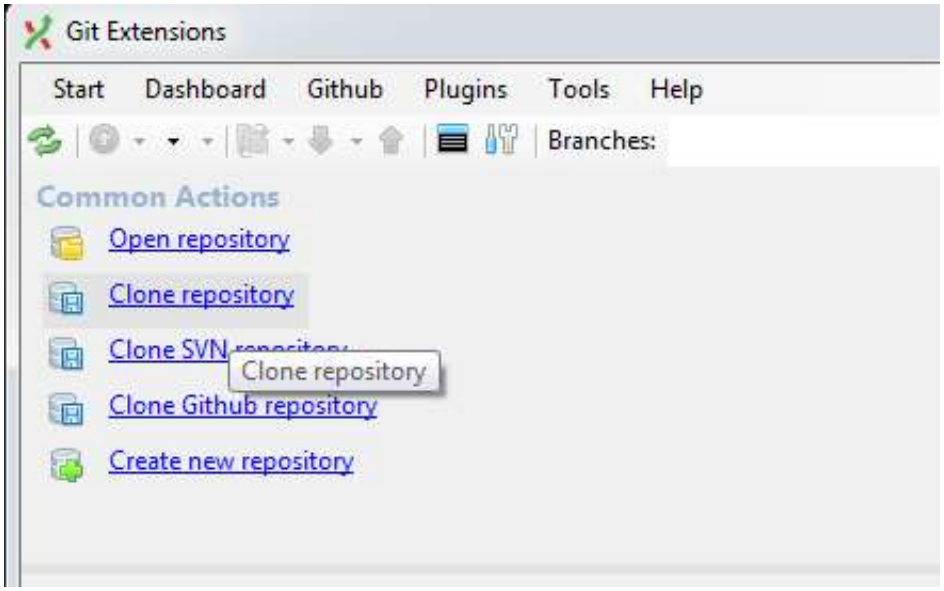
4.3 Prepojenie centrálneho úložiska s nástrojom na manažment verzií

Vstup: Nainštalovaný a nakonfigurovaný nástroj na manažment verzií (Git Extensions)
Výstup: Prístup k centrálnemu úložisku pomocou nástroja na manažment verzíi (Git Extentions)
Zodpovednosť: programátor

Proces prepojenia centrálne s nástrojom na manažment verzií ma na starosti každý člen týmu.

Tento process môžeme rozdeliť do nasledujúcich krokov:

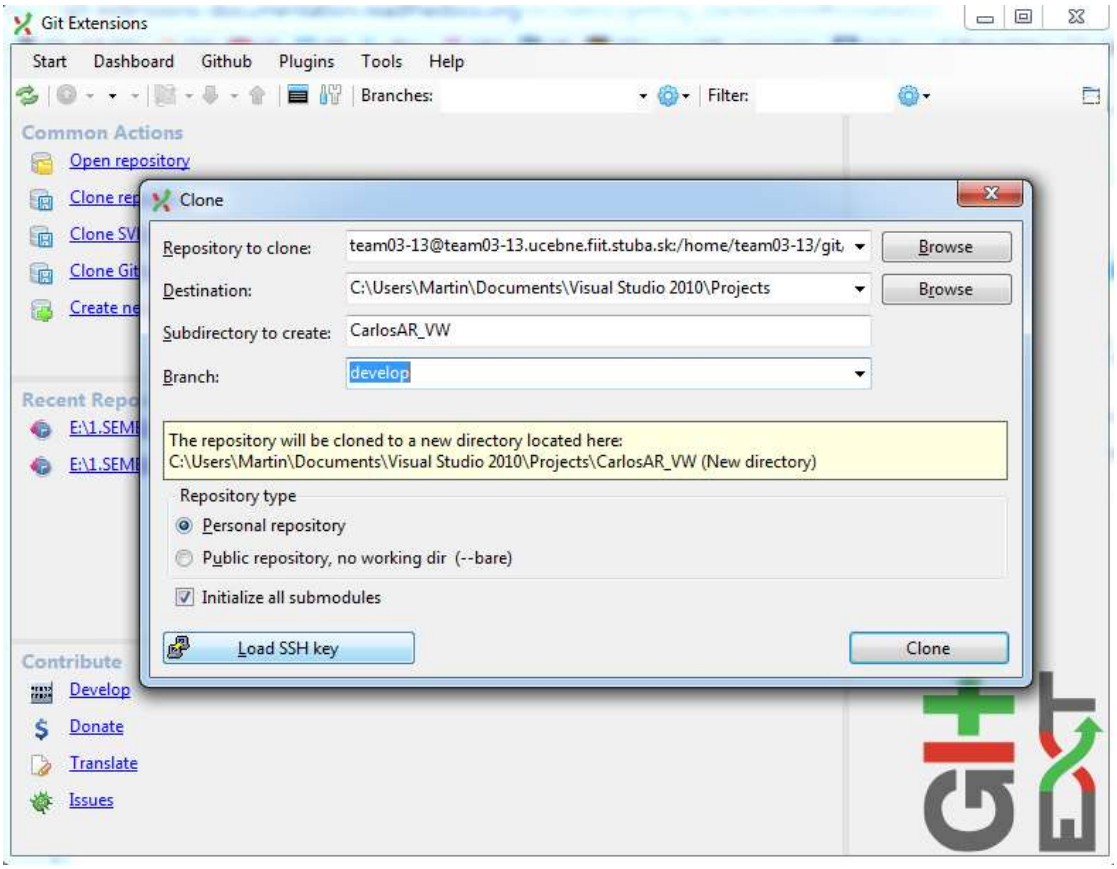
Po prvom spustení aplikácie je potrebné vybrať “Clone repository”



Obr.2 Clone repository

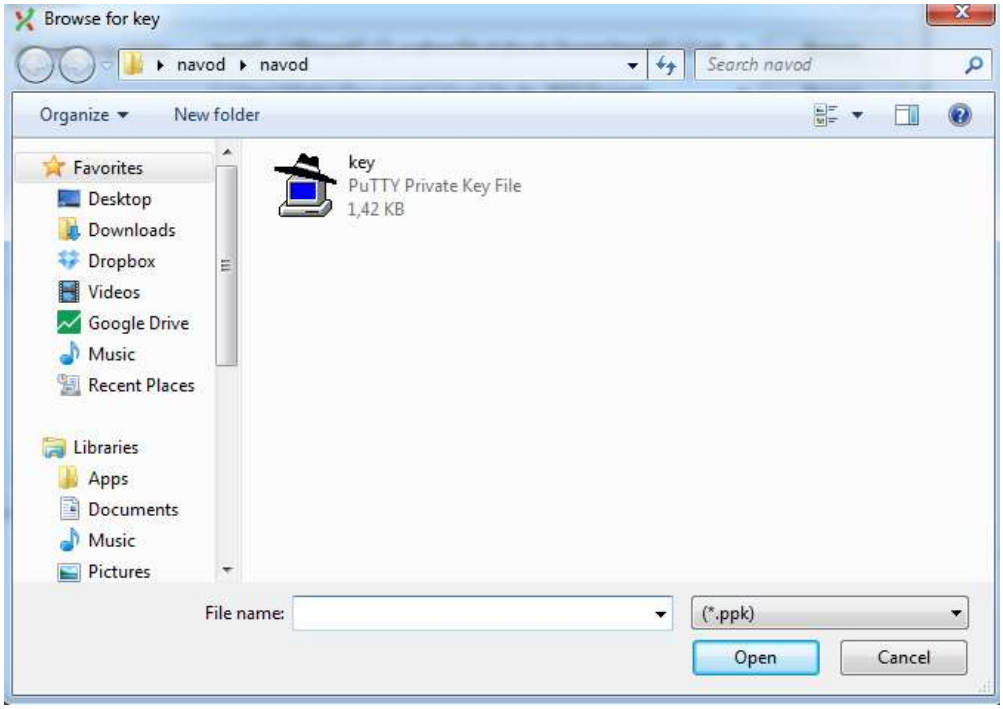
Následne vyplniť údaje podľa našeho tímového repository:

- **Repository to clone**: eam03-13@team03-13.ucebne.fiit.stuba.sk:/home/team03-13/git/CarlosAR_VW.git
- **Destination:** Lokálna zložka napr.: C:\Users\Martin\Documents\Visual Studio 2010\Projects
- **Branch:** develop
- **Repository type:** Personal repository
- **Load SSH key:** vybrať key.ppk (Pri tom treba mať v nastavení nastavené Settings - > SSH.Putty, prípadne tí čo sú na Linuxe tak treba vložit key a mat v Settings -> SSH ... OpenSSH). V remotes treba mať nastavené Origin na team03-13@team03-13.ucebne.fiit.stuba.sk:/home/team03-13/git/CarlosAR_VW.git a správne načítaný ten SSH kľúč



Obr.3 Ukážka vyplnenia nasledujúcich údajov

Po kliknutí na SSH key vybrať key.ppk



Obr.4 Vybratie key.ppk

4.4 Proces nasadzovania

Vstup: Vývojová verzia systém, Akceptačné testy
Výstup: Produkčná verzia systému, Zápis z akceptačného testovania
Zodpovednosť: Manažér vývoja, manažér kvality

Z pohľadu manažmentu verzií je dôležité rozlíšiť medzi rôznymi verziami systému. Preto sa rozlišujú tieto 2 vetvy v našom projekte:

- Hlavná (master)
- Vývojová (develop)

Manažment verzií zdrojového kódu pomocou Git. Prípravná verzia systému slúži na odskúšanie a integráciu otestovaných funkcionalít pred ich nasadením do produkčnej verzie, nasadzuje sa z prípravnej (develop) vetvy. Produkčná verzia systému je prezentovaná zákazníkovi, preto zahŕňa len odladené (otestované) funkcionality, ktoré prešli integračnými a akceptačnými testami. Táto verzia sa aktualizuje po ukončení každého vývojového cyklu, nasadzuje sa z hlavnej (master) vetvy. O nasadení do prípravnej verzie rozhoduje manažér vývoja, do produkčnej verzie manažér kvality. Okrem toho sa podľa potreby vytvárajú ďalšie vetvy systému, zachytávajúce ho v istom stave.

5. Záznamy zo stretnutí

V tejto kapitole sa nachádzajú týždenné zápisnice zo stretnutí nášho tímu. Každá zápisnica obsahuje zápis priebehu stretnutia, stav existujúcich pridelených úloh a zoznam nových úloh.

Zápis 1. stretnutia tímu č. 3

Autor zápisu:	Bc. Patrik Polatsek	
Dátum:	30.9.2013	
Miestnosť:	Laboratórium počítačového videnia a grafiky	
Prítomní:	Vedúci:	Ing. Vanda Benešová, PhD.
	Členovia tímu:	Bc. Peter Hamar
		Bc. Juraj Jarábek
		Bc. Jakub Mercz
		Bc. Marianna Mušinská
		Bc. Martin Petluš
		Bc. Patrik Polatsek
		Bc. Róbert Sabol
		Bc. Lukáš Sekerák

Priebeh stretnutia:

- Vedúca tímu načrtla priebeh práce na projekte
- Na vývoj aplikácie bude k dispozícii LED projektor a transparentná fólia, na ktorú sa bude vysielať obraz
- Aplikáciu, ktorú máme vytvoriť sme rozdelili do nasledovných častí (komponentov) (Príloha A):
    - riadiaca časť: notebook bude celú aplikáciu spravovať a vykonávať
    - mobilná aplikácia: využitie gps a interakcia s mobilom → technológie: Java, Android
    - detekcia objektov: z videosekvencií sa budú detegovať objekty (pomocou obrázkov uložených v databáze) → technológie: C++, OpenCV
    - sledovanie polohy hlavy: pre zobrazenie textu/obrazu v správnej pozícii voči hlave sa bude sledovať poloha hlavy → C++, KINECT
    - zobrazovacia časť: zobrazenie textu/obrazu na transparentnú fóliu → technológie: C++, OpenGL
- Predbežne sme si zadelili, ktorí členovia tímu sa budú venovať jednotlivým častiam projektu
- Na vývoj mobilnej aplikácie by sa dali využiť kódy od Richarda Sámela
- Ďalším cieľom projektu je vytvoriť hru, kt. bude reflektovať reálny svet, diskutovali sme možné námety na hru
    - hádať o akú budovu ide
    - uhádnuť dopravnú značku
- V prvej fáze projektu budeme zobrazovať len text, príp. doplnkový obraz (pre každý POI treba mať pripravené viaceré verzie a vždy zobraziť niečo iné)

Úlohy do ďalšieho stretnutia:

ID	Pridelené členovi	Opis úlohy
1	Lukáš, Jakub, Juraj	vytvoriť Android aplikáciu, ktorá bude ukladať aktuálnu gps polohu → timestamp pre videozáznam
2	Robo	kontaktovať sa s Richardom Samelom, získať kódy na gestá pre mobilnú aplikáciu
3	všetci členovia	pripraviť si nápady ako prepojiť jednotlivé komponenty (programy)

Prílohy:

Príloha A. Prvotný návrh štruktúry aplikácie

Zápis 2. stretnutia tímu č. 3

Autor zápisu: Bc. Martin Petluš
Dátum: 7.10.2013
Miestnosť: Laboratórium počítačového videnia a grafiky

Prítomní:
Vedúci: Ing. Vanda Benešová, PhD.
Členovia tímu: Bc. Peter Hamar
Bc. Juraj Jarábek
Bc. Jakub Mercz
Bc. Marianna Mušinská
Bc. Martin Petluš
Bc. Patrik Polatsek
Bc. Róbert Sabol
Bc. Lukáš Sekerák

Stav zadaných úloh z minulého stretnutia:

ID	Pridelené členovi	Opis úlohy	Stav
1	Lukáš, Jakub, Juraj	vytvoriť Android aplikáciu, ktorá bude ukladať aktuálnu GPS polohu → timestamp pre videozáznam	vyriešené
2	Robo	kontaktovať sa s Richardom Samelom, získať kódy na gestá pre mobilnú aplikáciu	vyriešené
3	všetci členovia	pripraviť si nápady ako prepojiť jednotlivé komponenty (programy)	v procese riešenia

Priebeh stretnutia:

- Zhodli sme sa na tom, že najlepšie bude používať vždy tú istú kameru na snímanie - kamera bude požičaná od Andreja Fogeltona
- Všetky dáta si budeme zbierať sami (GPS a video)
- Treba porozmýšľať nad zaujímavými trasami so zaujímavými objektmi (historické pamiatky, …)
- Ako vstup budeme mať dostupnú aj informáciu o pohybe auta
- Diskutovali sme o Jakubom vytvorenej aplikácie na zaznamenávanie GPS polohy
    - treba vyskúšať častejšie záznamy ako každú sekundu

    - zaznamenaná poloha a čas sa zatiaľ ukladajú do súboru, v budúcnosti bude treba vymyslieť nejaký iný spôsob, ktorý sa bude čo najbližšie približovať realtime času
- Diskutovali sme o Lukášom navrhovanej architektúre aplikácie:
    - modul s Kinnectom bude posielať polohu a uhol hlavy
    - treba si dať pozor, aby to nebolo príliš distribuované
    - systém musí byť skoro realtimeový
    - lokálna databáza na každom zariadení, každý modul bude samostatný
    - aplikácia sa najprv implementuje ako rôzne triedy a potom sa rozhodne čo ďalej (či každý modul bude ako samostatný spustiteľný súbor - program alebo jeden veľký program)
    - tiež je možnosť, že budeme mať jeden hlavný program (najmenší) a potom samostatné knižnice
    - treba sa dohodnúť na moduloch a ich interfaceoch
    - treba si definovať, aké príkazy sa budú posielať z mobilu (dotykové, rečové vstupy)
- možnou zábavnou hrou pre deti by bolo lietadielko, ktoré by sa muselo vyhýbať horizontu, alebo zbierať napríklad mince, lietadielko by sa ovládalo s pomocou mobilného zariadenia
- diskutovali sme o najlepšej polohe Kinnectu pri snímaní polohy hlavy (je treba zisťovať aj polohu očí, alebo stačí len poloha hlavy?, ...)
- vedúca pozná človeka, ktorý vie o nejakej knižnici, ktorá nám uľahčí prácu s Kinnectom
- diskutovali sme o aplikácií, ktorú by sme chceli využiť v našej aplikácií na rozpoznávanie gest:
    - mohli by sme tiež využiť rozpoznávanie hlasu - na slovenčinu to nefunguje dobre a nefunguje to bez internetu, s angličtinou to tiež nefunguje najlepšie (výslovnosť) - mohli by sme avšak použiť jednoduché povely ako one, two, stop, run, …
    - treba si rozmyslieť ako sa dá spojiť aplikácia na rozpoznávanie gest s naším projektom (cez databázu)

Úlohy do ďalšieho stretnutia:

ID	Pridelené členovi	Opis úlohy
1	Martin	študovať OpenCV, vytvoriť prvú verziu webovej stránky tímu
2	Lukáš	pozrieť sa na architektúru aplikácie podrobnejšie, nainštalovať databázu
3	Patrik	porovnávať rôzne metódy detegovania objektov, na nejakej snímke skúšať detegovanie objektov, vyskúšať či je lepšie hľadať objekt nanovo na každom snímku, ale sledovať objekt
4	Robo	skúsiť ukladať gestá do lokálnej databázy a skúsiť odoslať informácie o gestách na lokálny server

5	Marianna a Peťo	preštudovať si OpenGL, pracovať na nejakej jednoduchšej úlohe (napríklad vypisovať texty na rôzne miesta na obrazovke, pracovať s nejakým 3D modelom, dopĺňať text do obrázku)
6	Jakub	nainštaluje knižnicu pre prácu s Kinnectom a spraví experimenty na snímanie hlavy z rôznych pohľadov Kinnectu
7	Juraj	doriešiť aplikáciu na záznam GPS, pozrieť sa spolu s Robom na ukladanie gest v lokálnej databáze
8	všetci členovia	každý si podrobnejšie premyslí nejaký druh hry (napríklad zábavného typu, vzdelávacieho typu, atd.), najmä také, ktoré dokážu zabaviť deti

Prílohy:

Zápis 3. stretnutia tímu č. 3

Autor zápisu: Bc. Jakub Mercz
Dátum: 14.10.2013
Miestnosť: Laboratórium počítačového videnia a grafiky

Prítomní: Vedúci: Ing. Vanda Benešová, PhD.
Členovia tímu: Bc. Peter Hamar
Bc. Juraj Jarábek
Bc. Jakub Mercz
Bc. Marianna Mušinská
Bc. Martin Petluš
Bc. Patrik Polatsek
Bc. Róbert Sabol
Bc. Lukáš Sekerák

Stav zadaných úloh z minulého stretnutia:

ID	Pridelené členovi	Opis úlohy	Stav
1	Martin	študovať OpenCV, vytvoriť prvú verziu webovej stránky tímu	vyriešené
2	Lukáš	pozrieť sa na architektúru aplikácie podrobnejšie, nainštalovať databázu	v procese riešenia, nemohol nainštalovať kvôli nefunkčnosti serveru
3	Patrik	porovnávať rôzne metódy detegovania objektov, na nejakej snímke skúšať detegovanie objektov, vyskúšať či je lepšie hľadať objekt nanovo na každom snímku, ale sledovať objekt	vyriešené
4	Robo	skúsiť ukladať gestá do lokálnej databázy a skúsiť odoslať informácie o gestách na lokálny server	v procese riešenia, nemal potrebný HW
5	Marianna a Peťo	preštudovať si OpenGL, pracovať na nejakej jednoduchšej úlohe (napríklad vypisovať texty na rôzne miesta na obrazovke, pracovať s nejakým 3D	vyriešené

		modelom, dopĺňať text do obrázku)	
6	Jakub	nainštaluje knižnicu pre prácu s Kinnectom a spraví experimenty na snímanie hlavy z rôznych pohľadov Kinnectu	vyriešené
7	Juraj	doriešiť aplikáciu na záznam GPS, pozrieť sa spolu s Robom na ukladanie gest v lokálnej databáze	vyriešené
8	všetci členovia	každý si podrobnejšie premyslí nejaký druh hry (napríklad zábavného typu, vzdelávacieho typu, atd.), najmä také, ktoré dokážu zabaviť deti	vyriešené

Priebeh stretnutia:

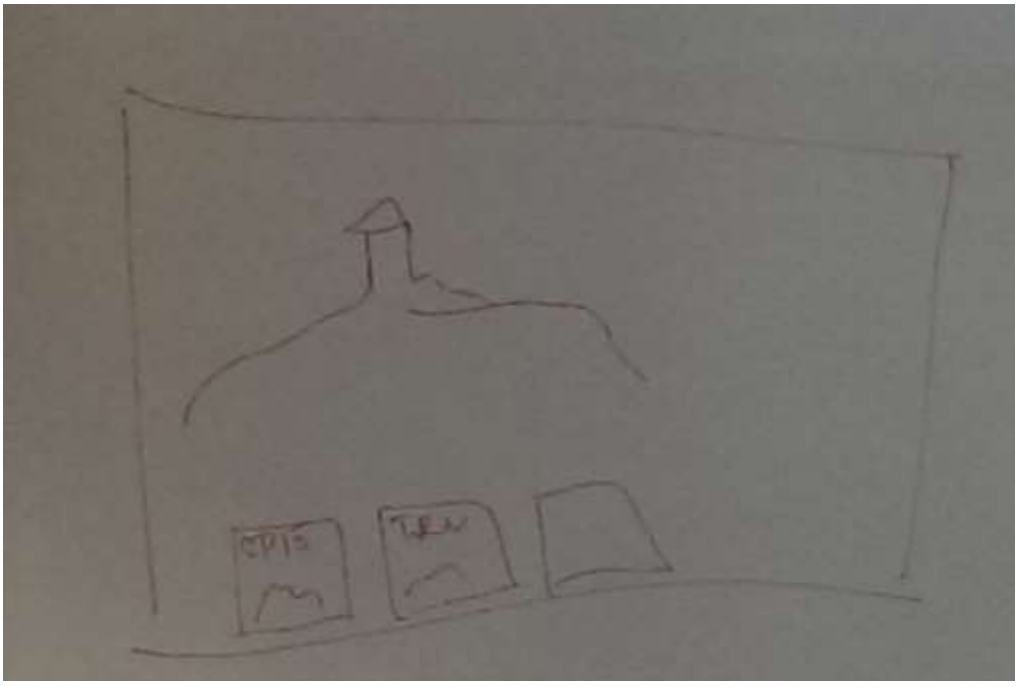
- popri kontrolovaní stavu úloh sme diskutovali o možných hrách
    - vyberanie z 3 obrázkov s názvami a hádanie čo je vidno za oknom (Príloha A)
    - rozstrieľavanie zväčšeného obrazu objektu (Príloha B)
    - lietadlo, ktoré musí zostať nad horizontom a čím bližšie k horizontu letí, tým viac bodov sa pripočítava
    - hra na štýl obesenca, háda sa objekt za oknom alebo informácia k nemu
    - určiť zo zobrazených hodnôt aký vysoký alebo starý je objekt za oknom
    - hry založené na počítaní objektov za oknom, napríklad okien
- zhodli sme sa, že na stránku treba pridať potrebné logá a ochrániť emailové adresy proti spambotom
- treba aby všetci určili vstupy a výstupy svojho modulu
- pozerali sme záznamy od Juraja, ktoré natočil v Trenčíne
    - zistili sme že v noci sú kompletne nevhodné podmienky
    - v meste treba počítať s budovami cez celú zornú plochu
- Dohodli sme sa na dvojiciach pre code review:
    - Peter - Marianna
    - Juraj - Robo
    - Jakub - Lukáš
    - Martin - Patrik

Úlohy do ďalšieho stretnutia:

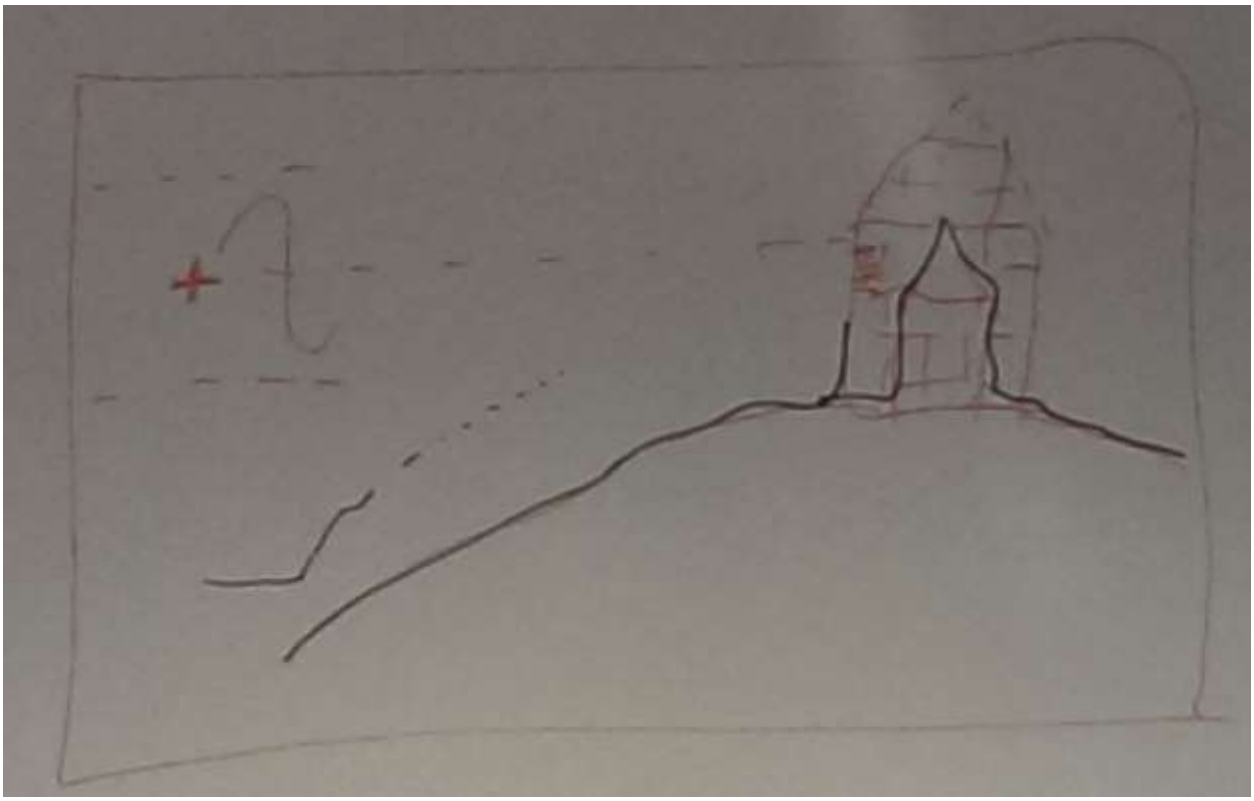
ID	Pridelené členovi	Opis úlohy
1	Juraj	natočiť ďalšie videá, aj mimo mesta, pripojiť k nim GPS informácie, sprístupniť GPS aplikáciu
2	Lukáš	dozerať na vloženie informácií pre architektúru
3	Patrik	zobrať fotky z google streetview a odskúšať detekciu
4	Martin	zaoberať sa novým modulom, ktorý ma za úlohu vypočítať polohu objektov na okne
5	Marianna a Peťo	pokračovať s OpenGL, odskúšať objavujúce sa a miznúce objekty
6	Robo	demo pre rozpoznávanie giest
7	Jakub	zaoberať sa detekciou tváre a spracovávať hĺbkovú informáciu
8	všetci členovia	definovať aké vstupy a výstupy má každý modul

Prílohy:

Príloha A. Návrh hry #1



Príloha B. Návrh hry #2



Zápis 4. stretnutia tímu č. 3

Autor zápisu: Bc. Juraj Jarábek
Dátum: 14.10.2013
Miestnosť: Laboratórium počítačového videnia a grafiky

Prítomní: Vedúci: Ing. Vanda Benešová, PhD.
Členovia tímu: Bc. Peter Hamar
Bc. Juraj Jarábek
Bc. Jakub Mercz
Bc. Marianna Mušinská
Bc. Martin Petluš
Bc. Patrik Polatsek
Bc. Róbert Sabol
Bc. Lukáš Sekerák

Stav zadaných úloh z minulého stretnutia:

ID	Pridelené členovi	Opis úlohy	Stav
1	Martin	zaoberať sa novým modulom, ktorý ma za úlohu vypočítať polohu objektov na okne	Martin sa zaoberal týmto modulom a prišiel s novými nápadmi, vypočítanie vzdialenosti a ďalšie srandy, in progress
2	Lukáš	dozerať na vloženie informácií pre architektúru	Bolo to rozpracované a sú navrhnuté prvé templatey, menšie problémy s databázou. stále in progress.
3	Patrik	zobrať fotky z google streetview a odskúšať detekciu	Porovnanie bolo vykonané, využil existujúce kódy/metódy, stále in progress.
4	Robo	demo pre rozpoznávanie giest	stále sa pracuje, in progress
5	Marianna a Peťo	pokračovať s OpenGL, odskúšať	spravili sa nejaké

		objavujúce sa a miznúce objekty	animácie, pridanie textu, hýbanie textu, jeho zobrazenie
6	Jakub	zaoberať sa detekciou tváre a spracovávať hĺbkovú informáciu	in progress
7	Juraj	natočiť ďalšie videá, aj mimo mesta, pripojiť k nim GPS informácie, sprístupniť GPS aplikáciu	vyriešené
8	všetci členovia	definovať aké vstupy a výstupy má každý modul	

Priebeh stretnutia:

- najskôr sme si skúsili priehľadnú fóliu s projektorom a vyhnotili, že projekcia na fóliu je príliš silná a video v pozadí nie je dobre vidieť.
- optické preloženie projektorov cez seba aby to dávalo zmysel (aby bol obraz správne preložený)
- vytvorili sme si plán / schému architektúry (viď. príloha)
    - každý modul by mohol bežať ako samostatný thread
    - Robova Android aplikácia pre gestá by bola ako samostatná časť (resp. jediná externá časť) a komunikovala s programom cez TCP protokol (zatiaľ budeme)
- diskutovali sme hlbšie o celej architektúre, podrobnejšie rozvrhnutie modulov
- dohodli sme sa sme sa dlhodobejšie:
    - registrácia obrazu/kalibrácia (lokálne deskriptory), najskôr sa budeme hrať staticky akože auto stojí

- Dohodli sme sa na dvojiciach pre code review:
    - Peter - Marianna
    - Juraj - Robo
    - Jakub - Lukáš
    - Martin - Patrik

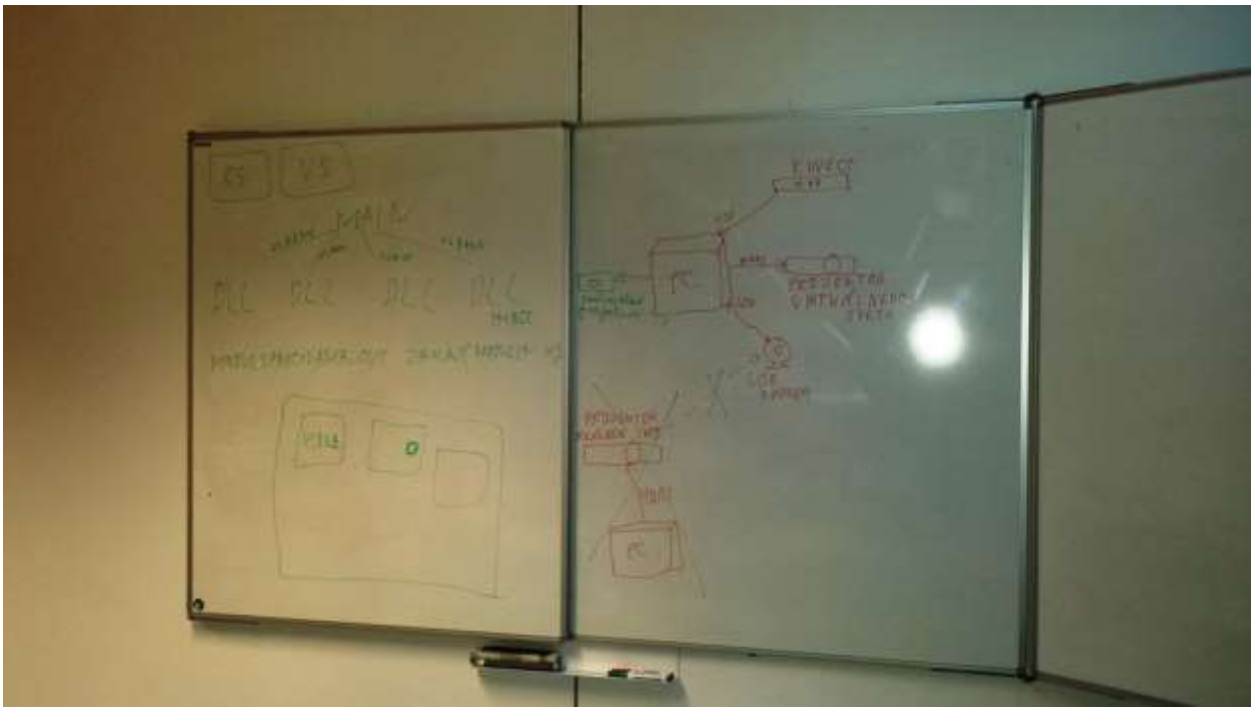
Úlohy do ďalšieho stretnutia:

ID	**Pridelené členovi**	**Opis úlohy**
1	Juraj	pozrieť si projekt s Robom

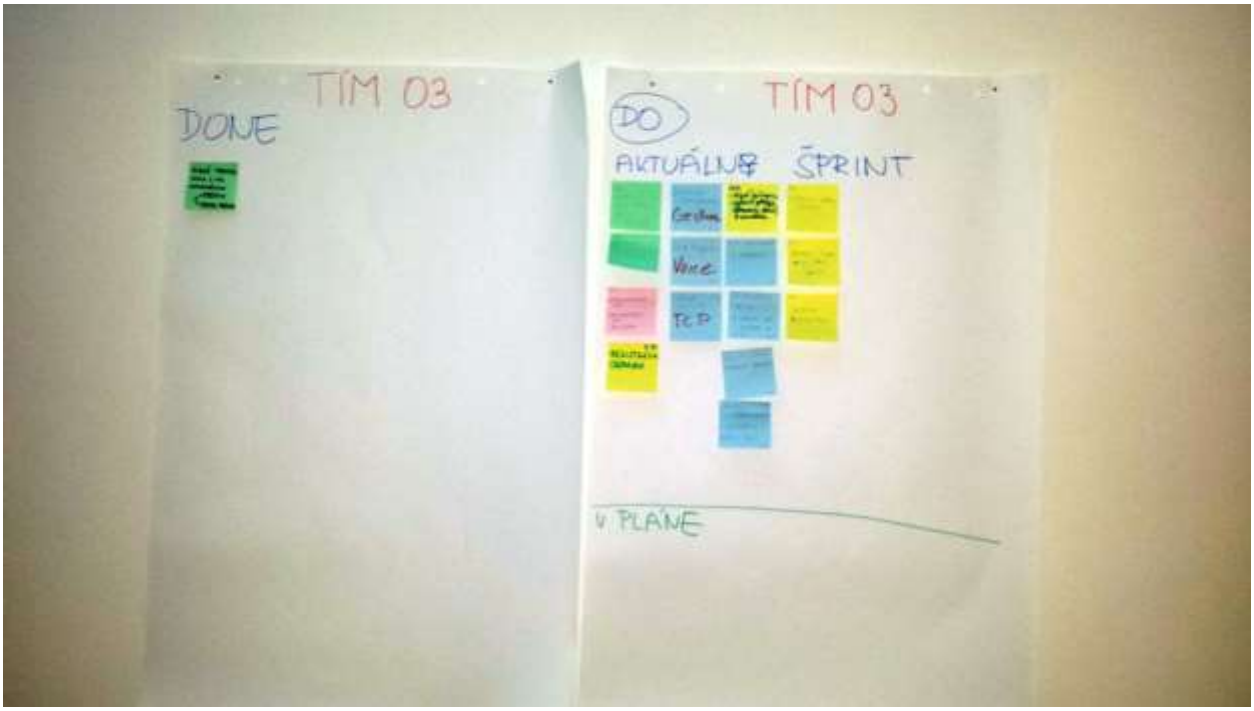
2	Lukáš	pohrať sa s databázou, SQLite
3	Patrik	registrácia obrazu/kalibrácia (lokálne deskriptory) ako priorita, neskôr detekcia
4	Martin	Zastaviť za Kapcom a vytvoriť prototyp na demonštrovanie výpočtov polohy hlavy v Matlabe
5	Marianna a Peťo	využiť OpenCV, začať nejakú najľahšiu hru
6	Robo	TCP Lukášovi, gestá a hlas
7	Jakub	zaoberať sa detekciou tváre a spracovávať hĺbkovú informáciu
8	všetci členovia	

Prílohy:

Príloha A. Návrh architektúry



Príloha B. Nová nástenka


TÍM 03
DONE
TÍM 03
DO
AKTUÁLNÝ ŠPRINT
V PLÁNE

Zápis 5. stretnutia tímu č. 3

Autor zápisu: Bc. Róbert Sabol
Dátum: 28.10.2013
Miestnosť: Laboratórium počítačového videnia a grafiky

Prítomní: Vedúci: Ing. Vanda Benešová, PhD.
Členovia tímu: Bc. Peter Hama
Bc. Jakub Mercz
Bc. Marianna Mušinská
Bc. Martin Petluš
Bc. Patrik Polatsek
Bc. Róbert Sabol
Bc. Lukáš Sekerák

Stav zadaných úloh z minulého stretnutia:

ID	Pridelené členovi	Opis úlohy	Stav
1	Martin	Zastaviť za Kapcom a vytvoriť prototyp na demonštrovanie výpočtov polohy hlavy v Matlabe	Martin za Kapcom nebol, avšak spravil prostredie na testovanie textu + vytvoril v Java Scripte demo ako prvý návrh výpočtov pre pozíciu hlavy
2	Lukáš	pohrať sa s databázou, SQLite	Pripravil SQLite databázu – dátový model, nahodil do nej dáta
3	Patrik	registrácia obrazu/kalibrácia (lokálne deskriptory) ako priorita, neskôr detekcia	Robil detekciu objektov, porovnával objekty . min, max vzdialenosti Spojazdnil knižnicu od Mareka Raceva
4	Robo	TCP Lukášovi, gestá a hlas	Rozpracoval TCP komunikáciu – in progress
5	Peťo	využiť OpenCV, začať nejakú najľahšiu	Spravil jednoduchú hru –

		hru	kvíz, obrázky otázky
6	Jakub	zaoberať sa detekciou tváre a spracovávať hĺbkovú informáciu	in progress
7	Juraj	pozrieť si projekt s Robom	vyriešené
8	Marianna	využiť OpenCV, začať nejakú najľahšiu hru	Neúspešné, nešlo jej to s OpenGL

Priebeh stretnutia:

- najskôr Lukáš predstavil ako spravil hlavný projekt, ktorý bude handlovat všetky ostatné
- ukázal nám databázu – diagram tried, a vysvetlil ako mame využívať moduly z jeho projektu
- zhodli sme sa, že budeme používať Doxygen
- zhodli sme sa, že Marianna bude dokumentaristka
- povedali sme si o kalibrácii hlavy a toho čo vidíme v Patrikovej časti projektu
- zhodli sme sa, že Peťova hra na testovanie stačí – doplní do nej 3D modely
- optické preloženie projektorov cez seba aby to dávalo zmysel (aby bol obraz správne preložený)

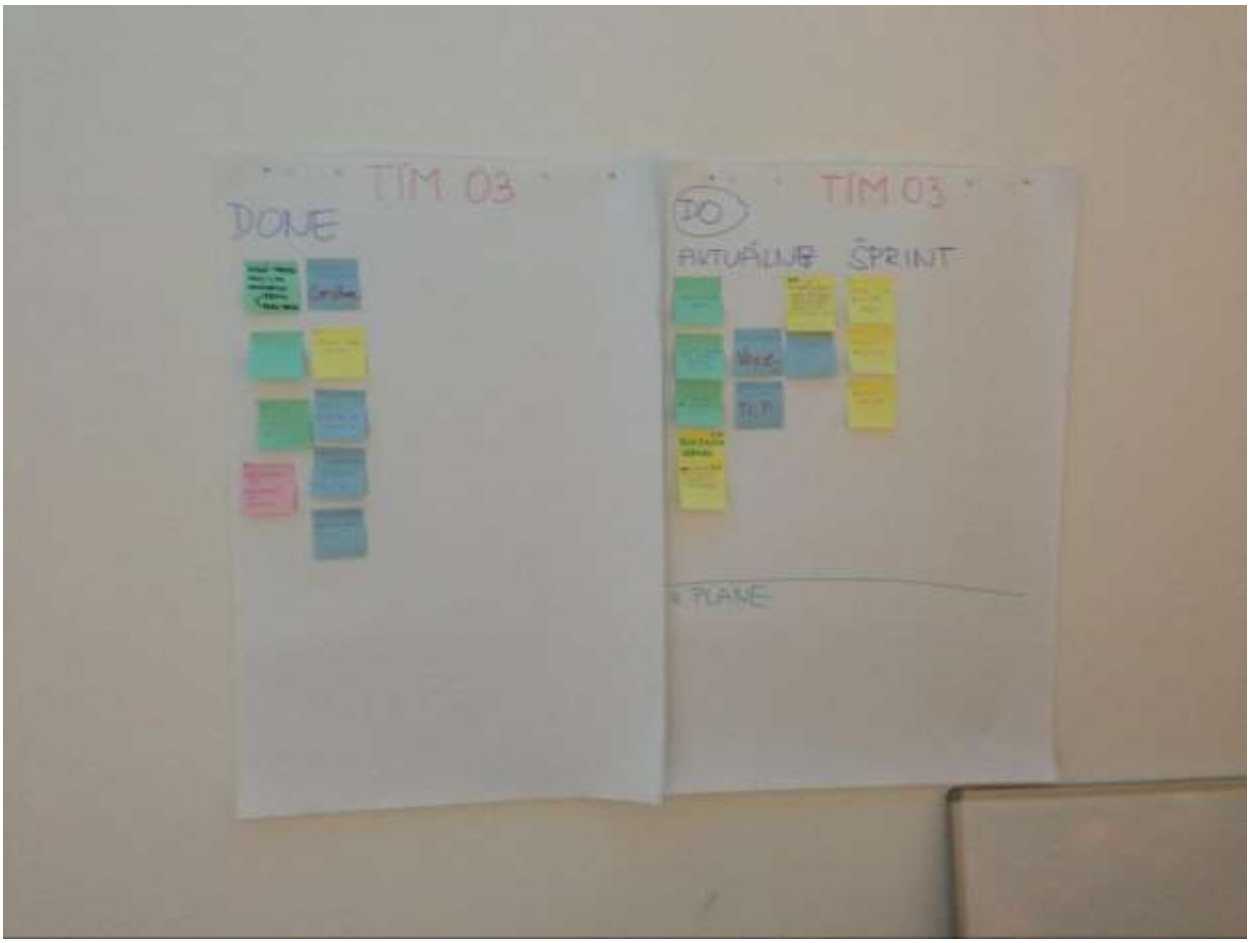
Úlohy do ďalšieho stretnutia:

ID	**Pridelené členovi**	**Opis úlohy**
1	Juraj	Doxygen – pozrieť a pripraviť jednoduchý dokument pokyny čo máme písať a kam aby to potom cele fungovalo
2	Lukáš	Sprístupniť moduly, dávať rady ako používať jeho moduly, Doxygen, video na snímky
3	Patrik	Finalizuje rozoznávanie objektov, registrácia objektov
4	Martin	Vypočítavať pozíciu objektu
5	Marianna	TCP modul do C++ projektu
6	Peťo	3D model, lietadlo
7	Robo	TCP komunikácia z Androidu

8	Jakub	Zameria tvar – nejaký bod na tvarí, vzdialenosť
9	všetci členovia	Nájsť v Lukášovom projekte svoj modul

Prílohy:

Príloha A. Nová nástenka



Zápis 6. stretnutia tímu č. 3

Autor zápisu: Bc. Marianna Mušinská
Dátum: 4.11.2013
Miestnosť: Laboratórium počítačového videnia a grafiky

Prítomní: Vedúci: Ing. Vanda Benešová, PhD.
Členovia tímu: Bc. Peter Hamar
Bc. Juraj Jarábek
Bc. Jakub Mercz
Bc. Marianna Mušinská
Bc. Martin Petluš
Bc. Patrik Polatsek
Bc. Róbert Sabol
Bc. Lukáš Sekerák

Stav zadaných úloh z minulého stretnutia:

ID	Pridelené členovi	Opis úlohy	Stav
1	Juraj	Doxygen – pozrieť a pripraviť jednoduchý dokument pokyny čo mame písať a kam aby to potom cele fungovalo	v procese riešenia
2	Lukáš	Sprístupniť moduly, dávať rady ako používať jeho moduly, Doxygen, video na snímky	vyriešené
3	Patrik	Finalizuje rozoznávanie objektov, registrácia objektov	vyriešené
4	Martin	Vypočítavať pozíciu objektu	vyriešené
5	Robo	TCP komunikácia z Androidu	v procese riešenia
6	Jakub	tvar – nejaký bod na tvári, vzdialenosť	vyriešené
7	Peťo	3D model, lietadlo	v procese riešenia

8	Marianna	TCP modul do C++ projektu	nevyriešené, nevedela sa prepojiť s Androidom
9	všetci členovia	Nájsť v Lukášovom projekte svoj modul	vyriešené

Priebeh stretnutia:

- diskusia o pomenovaní jednotlivých modulov
- diskusia o sfunkčnení git - musí sa sfunkčniť čím skôr
- diskusia o tom, kde budú uložené GPS pozície objektu, ktoré potrebujeme na výpočet pozície
- Lukáš nám ukázal a vysvetlil na konkrétnom prípade ako fungujú moduly
- bolo nám odporučené používať namespace-y
- rozhodli sme sa, že budeme používať debuggery
- Jakubovi nastali menšie komplikácie pri spúšťaní svojho programu a bolo mu odporučené používať hĺbkovú mapu
- diskusia o hernej logike, a ako by mal vyzerať main
    - výsledok diskusie: main musí byť čisto riadiaci
    - a herná logika môže byť jeden modul
- zhodli sme sa, že musíme používať chybové hlášky, a musíme vedieť odchytiť výnimky
- treba vytvoriť class diagram najlepšie na papier, aby sme vedeli na tom rukou editovať

Úlohy do ďalšieho stretnutia:

1	Juraj	Doxygen - pripraviť užívateľskú príručku (dokument)
2	Lukáš	pripojiť Patrika,
3	Patrik	sfinalizovanie modulu spracovania obrazu a registrácie obrazu
4	Martin	sfunkčniť git, testovanie
5	Robo	
6	Jakub	preštudovať používanie hĺbkovej mapy

7	Peťo	lietadlo – pohyb pomocou klávesnice
8	Marianna	dokumentácia
9	všetci členovia	prepojiť sa s git, používať Doxygen, pomoc pri dokumentácii jednotlivých modulov, nakresliť class diagram

Prílohy:

Príloha A. Nová nástenka

Zápis 7. stretnutia tímu č. 3

Autor zápisu: Bc. Peter Hamar
Dátum: 4.11.2013
Miestnosť: Laboratórium počítačového videnia a grafiky

Prítomní: Vedúci: Ing. Vanda Benešová, PhD.
Členovia tímu: Bc. Peter Hamar
Bc. Juraj Jarábek
Bc. Jakub Mercz
Bc. Marianna Mušinská
Bc. Martin Petluš
Bc. Patrik Polatsek
Bc. Róbert Sabol
Bc. Lukáš Sekerák

Stav zadaných úloh z minulého stretnutia:

ID	Pridelené členovi	Opis úlohy	Stav
1	Juraj	Doxygen - pripraviť užívateľskú príručku (dokument)	vyriešené
2	Lukáš	pripojiť Patrika, vygenerovanie class diagramu, nainštalovanie databázy	vyriešené
3	Patrik	sfinalizovanie modulu spracovania obrazu a registrácie obrazu	vyriešené
4	Martin	sfunkčniť git, testovanie	vyriešené
5	Robo	TCP komunikácia z Androidu (od minula)	v procese riešenia
6	Jakub	preštudovať používanie hĺbkovej mapy	vyriešené
7	Peťo	3D model lietadla – pohyb pomocou klávesnice	vyriešené

8	Marianna	dokumentácia	nevyriešené
9	všetci členovia	prepojiť sa s git, používať Doxygen, pomoc pri dokumentácii jednotlivých modulov, nakresliť class diagram	vyriešené

Priebeh stretnutia:

- Definovali sme presný zoznam features
  -presné rozpoznávanie objektov
  -vykonávanie aplikácie v reálnom čase
  -vypočítanie polohy pre zobrazenie doplňujúcich informácií
  -generovanie grafických prvkov AR
  -ovládanie systému pomocou Android aplikácie
  -ovládanie hry pomocou Android aplikácie
  -poskytovanie 2 aplikácií (hier), ktoré využívajú obohatenú
  -získavanie GPS a časových značiek v reálnom čase pomocou Android aplikácie
  -použitie databázy
  -vypočítanie polohy hlavy z Kinectu
  -integrácia komunikácie a celého systému

- Mariana nevedela napísať dokumentáciu. Keďže tento problém je potrebné urýchlene vyriešiť, tak sme sa dohodli, že dokumentáciu spojí a pripraví na odovzdanie Juraj.
- Git – problém s veľkými knižnicami OpenCV a Boost, vyrieši sa to pomocou premenných ciest.
- Diskusia týkajúca sa výpočtu polohy hlavy voči oknu a kalibrácia.
- Diskusia týkajúca sa umiestnenia Kinectu v aute – hĺbka hlavy.
- Diskusia týkajúca sa hry, kto určí horizont. Je potrebná spolupráca s Patrikom, ktorý určí horizont. Poskytne modulu hry 2 obrázky, jeden s horizontom a jeden klasický. Modul hry ich spojí a účelom bude aby lietadlo letelo nad horizontom. Čím bližšie bude letieť tým viac bodov hráč získa.

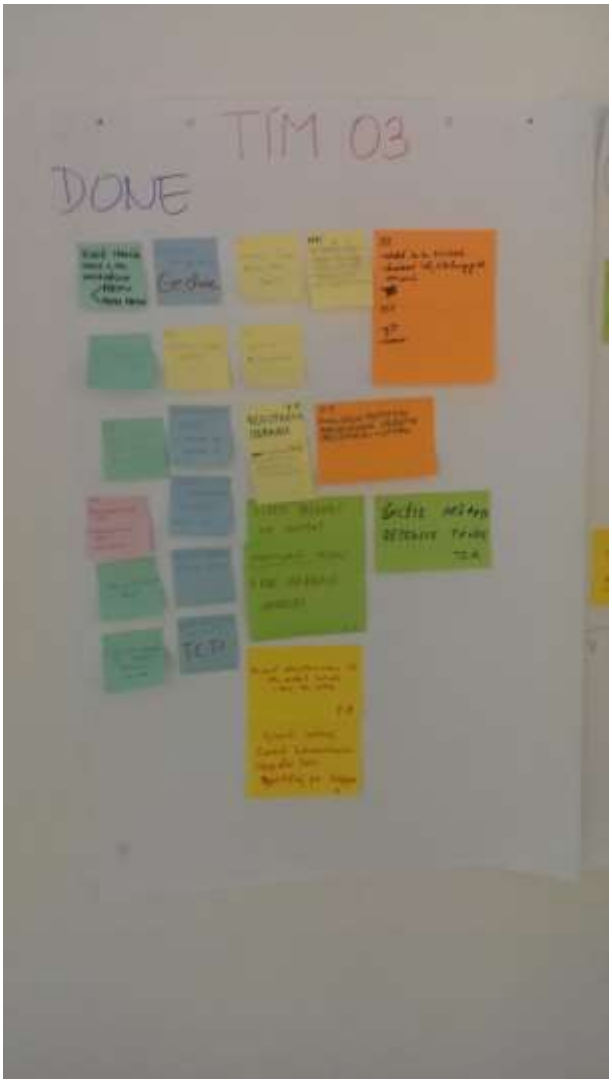
Úlohy do ďalšieho stretnutia:

1	Juraj	Dokumentácia – zintegrovať jednotlivé časti a odovzdanie
2	Lukáš	TCP server, kontrola integrácie, uploadnuť na git projekt
3	Patrik	Detekcia horizontu a vylepšovanie spracovania obrazu
4	Martin	Pripraví knižnice pre Git, nainštaluje team foundation server, testovanie svojho modulu
5	Robo	Prerobiť serverovú časť aby nepadala

6	Jakub	Detekcia hlavy cez hĺbkovú informáciu , kalibrácia na reálny svet
7	Peťo	Pripraviť scénu za objektom (video) a skúsiť lietadlo držať nad horizontom
8	Marianna	
9	všetci členovia	Prečítať inštrukcie od Juraja k Doxygenu

Prílohy:

Príloha A. Aktuálna nástenka




TÍM 03
DO
AKTUÁLNY ŠPRINT
U PLÁNE

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
Fakulta informatiky a informačných technológií
Softvérové inžinierstvo a Informačné systémy

PREBERACÍ PROTOKOL

Názov tímu:	Tím č. 3 (Carlos)
Tímový projekt:	Zábavný systém pre spolucestujúcich v automobile (AUTO)
Vedúci tímu:	Ing. Vanda Benešová, PhD.
Členovia tímu:	Bc. Patrik Polatsek
	Bc. Martin Petluš
	Bc. Peter Hamar
	Bc. Róbert Sabol
	Bc. Jakub Mercz
	Bc. Lukáš Sekerák
	Bc. Marianna Mušinská
	Bc. Juraj Jarábek

Počet strán: .....

Ing. Vanda Benešová, PhD., týmto potvrdzuje prevzatie Projektovej dokumentácie a Dokumentácie riadenia projektu.

Podpis: .....

Bratislava, dňa 18.11.2013