

Tímový projekt

# REKONŠTRUKCIA 3D SCÉNY

## V2

*Riadenie*

**Vedúci projektu:**

Ing. Vanda Benešová, PhD.

**Členovia tímu:**

Bc. Jakub Ginter (SI)  
Bc. Miroslav Haščič (SI)  
Bc. Mário Hunka (SI)  
Bc. Viktor Košťan (IS)  
Bc. Richard Pintér (IS)

**Názov tímu:** 3DRecon (tím č. 11)

**Web:** <http://team11-16.studenti.fiit.stuba.sk/>

**Kontakt:** [teamfiit11@gmail.com](mailto:teamfiit11@gmail.com)

**Akademický rok:** 2016/2017

**Dátum odovzdania:** 21. 11. 2016

# Obsah

1	Úvod.....	1
1.1	Prehľad dokumentu .....	1
2	Role členov tímu a podiel práce .....	2
2.1	Manažérske činnosti .....	2
2.2	Podiel práce.....	2
3	Aplikácie manažmentov .....	3
3.1	Manažment komunikácie .....	3
3.1.1	Tímové stretnutia .....	3
3.1.2	Komunikačné nástroje .....	4
3.2	Manažment vývoja a integrácie.....	4
3.3	Manažment dokumentácie .....	5
3.4	Manažment plánovania .....	5
3.5	Manažment riadenia .....	6
4	Sumarizácia šprintov.....	7
4.1	Šprint 1 .....	7
4.2	Šprint 2 .....	7
4.3	Šprint 3 .....	8
5	Metodiky .....	8
5.1	Metodika Zadávania úloh do TFS .....	8
5.1.1	Tvorba Backlogu.....	9
5.1.2	Plánovanie šprintu.....	10
5.1.3	Práca na taskoch počas šprintu .....	11
5.2	Metodika prehliadok kódu .....	12
5.2.1	Základné ustanovenia .....	12
5.2.2	Priebeh prehliadky .....	12
5.2.3	Testing.....	13
5.2.4	Ako robiť prehliadku.....	14
5.3	Metodika verziovania kódu.....	14
5.3.1	Branches .....	14
5.3.2	Pull-Request.....	15
5.3.3	Commit.....	16
5.3.4	Súhrn pravidiel odosielaného kódu .....	16
5.4	Metodika testovania.....	16
5.4.1	Základné pravidlá.....	17
5.4.2	Priebeh testovania .....	17
5.5	Metodika písania dokumentácie .....	18
5.5.1	Základné požiadavky.....	18
5.5.2	Základné pravidlá.....	18
5.5.3	Správa dokumentácie .....	19
6	Globalna retrospektiva .....	20
7	Export úloh.....	22
8	Preberacie Protokoly .....	30
9	Motivačný dokument .....	31

# 1 ÚVOD

Tento dokument obsahuje postupy a metódy, ktoré boli definované naším tímom za účelom riadenia tímového projektu. Projekt rieši segmentáciu 3D scény z rekonštruovaných 3D dát. Hlavnou myšlienkou je umožniť segmentáciu scény z dát v ľubovoľnom formáte pričom výstup by mal byť použiteľný softvérom *AutoCAD*<sup>1</sup>.

Štafetu sme prebrali po minuloročnom tíme, ktorý viedol Bc. Lukáš Hudec. Vzhľadom nato, sme pokračovali v používaní niektorých metodík (metodika písania zdrojového kódu), avšak definovali sme si aj vlastné, ktoré sú špecifické pre náš tím.

Projekt je výskumného charakteru. Je teda potrebné umožniť testovanie rôznych segmentačných metód, ktoré po evaluácii môžu skončiť ako prototyp na zahodenie. Samotná práca s 3D dátami je výpočtovo, ale aj pamäťovo náročná. V súvislosti s tým, sme sa spočiatku začali uberať smerom, ktorý by urýchlil prototypovanie (ukladanie medzivýsledkov, ktoré neovplyvňujú výsledky testovania novej metódy). Ako ďalší z našich cieľov, je úplná segmentácia tzv. *outlayerov* (kľučka, svetlo atď.), ktoré vytvárajú nepresnosti pri segmentácii objektov, pri ktorých sa nachádzajú, prípadne sú ich súčasťou (stena, stôl atď.). V neposlednom rade, rozširujeme aj možnosti formátov vstupných dát. Ako ďalšie máme v pláne rekonštruovať 3D dáta nasnímané pomocou zariadenia *Kinect*<sup>2</sup> z rôznych uhl'ov pohľadov.

Výsledkom našej práce bude knižnica, ktorá poskytuje rôzne metódy segmentácie 3D dát využiteľná hlavne vo výskumnej sfére. V rámci vývoja tejto knižnice je samozrejme dostupné aj používateľské rozhranie, ktoré umožňuje používateľovi ľahko a efektívne, testovať nové segmentačné metódy.

## 1.1 PREHĽAD DOKUMENTU

V časti 2 sú opísané manažérske činnosti a zodpovednosti členov nášho tímu. V podkapitolách môžeme nájsť aj podiel práce členov tímu na tomto dokumente. V 3 kapitole sme uviedli aplikácie manažmentov, ktoré

---

<sup>1</sup> Nástroj určený na dizajn <http://www.autodesk.com/products/autocad/overview>

<sup>2</sup> Hardware Senzor od Microsoftu <http://www.xbox.com/en-US/xbox-one/accessories/kinect>

používame pri riadení nášho tímu. V kapitole 4 sme sumarizovali výsledky našich 3 šprintov. V rámci vývoja nášho softvéru používame určité metodiky, ktoré sú uvedené v časti 5. V ďalších častiach tohto dokumentu sa nachádzajú exporty z nástroja používaného na manažovanie (TFS), globálna retrospektíva a zápisnice zo stretnutí.

## 2 ROLE ČLENOV TÍMU A PODIEL PRÁCE

### 2.1 MANAŽÉRSKE ČINNOSTI

TABUĽKA 1: ROZDELENIE ÚLOH V TÍME

Meno	Rola	Zodpovednosť
Jakub Ginter	vedúci tímu, biznis manažér	analýza a návrh požiadaviek, komunikácia s produkt ownerom
Miroslav Haščič	manažér kvality	vykonávanie testov, riešenie a správa chýb (bug-ov)
Viktor Košťan	manažér vývoja a architektúry	návrh a implementácia architektúry, správa hlavných segmentálnych metód
Mário Hunka	integrátor, manažér propagácie	prepájanie jednotlivých modulov projektu, tvorba a udržiavanie web stránky
Richard Pintér	manažér dokumentácie	udržiavanie dokumentácie a záznamov zo stretnutí

### 2.2 PODIEL PRÁCE

TABUĽKA 2: PODIEL PRÁCE NA DOKUMENTÁCIÍ RIADENIA

Časť	Člen tímu
Úvod	Mário Hunka
Role členov tímu a podiel práce	Mário Hunka
Manažment komunikácie	Jakub Ginter
Manažment vývoja a integrácie	Jakub Ginter
Manažment plánovania	Mário Hunka

Manažment riadenia	Miroslav Haščič
Manažment dokumentovania	Richard Pintér
Sumarizácia šprintov	Richard Pintér
Metodika zadávania úloh do TFS	Viktor Košťan
Metodika prehliadok kódu	Mário Hunka
Metodika verziovania kódu	Jakub Ginter
Metodika testovania	Miroslav Haščič
Metodika XYZ	Richard Pintér
Globálna retrospektíva	Mário Hunka
Export úloh z TFS	Jakub Ginter
Integrácia a finalizácia	Mário Hunka

**TABUĽKA 3: PODIEL PRÁCE NA DOKUMENTÁCIÍ K INŽINIERSKEMU DIELU**

Časť	Člen tímu
Úvod	Jakub Ginter
Globálne ciele pre ZS	Jakub Ginter
Celkový pohľad na systém	Viktor Košťan
Moduly systému – úvod	Viktor Košťan
Segmentácia Outlayerov	Mário Hunka
Rekonštrukcia dát z Kinectu	Mário Hunka
Zlepšenie vizualizéra	Miroslav Haščič

## 3 APLIKÁCIE MANAŽMENTOV

### 3.1 MANAŽMENT KOMUNIKÁCIE

Naša komunikácia v tíme by sa dala rozdeliť na dva hlavné celky. Prvým sú tímové stretnutia a druhým komunikácia v nástrojoch

#### 3.1.1 TÍMOVÉ STRETNUTIA

Spoločne sme sa stretali podľa rozvrhu každý štvrtok o 13:00 v laboratóriu VGG 4.46. Na týchto stretnutiach sme spolu s vedúcou tímu zhodnocovali čo sa spravilo a čo sa bude robiť. Každý člen vysvetlil na čom pracoval a kde boli problémy, resp. aké by mohli v budúcnosti nastať. Tieto stretnutia

prebiehali podľa agilnej metódy SCRUM, ktorá je opísaná v manažmente plánovania. Tam je možné nájsť ako presne tieto stretnutia prebiehali.

Mimo týchto stretnutí sme sa stretávali aj v iných dňoch a riešili problémy, na ktoré sme narazili a bolo potrebné ich vyriešiť čo najskôr alebo jednoducho pracovali. Tieto stretnutia prebiehali zväčša vo VGG alebo v knižnici.

### 3.1.2 KOMUNIKAČNÉ NÁSTROJE

Na našu komunikáciu sme najčastejšie používali dva nástroje (Slack a Google Drive). V prvotných začiatkoch prác na projekte sme komunikovali prostredníctvom skupinového Facebook chatu. To sa však ukázalo ako neefektívne bolo potrebné nájsť vhodný nástroj.

Slack je aplikácia určená na komunikáciu v tímoch a zvýšenie ich produktivity. Je možné vytvoriť komunikačné kanály čo umožňuje rozdeliť komunikáciu do tematických celkov. V rámci kanálov je možné nastavovať notifikácie pre jednotlivých členov alebo skupinu. To má veľké využitie obzvlášť pri mobilnej aplikácii, ktorá je tiež k dispozícii. Používateľ tak môže nastaviť upozornenia iba na správy kde bol označený a vie, že je potrebné reagovať čo najskôr.

Pomocou tejto aplikácie je možné nahrávať súbory, zdieľať ukážky zdrojového kódu alebo upozorňovať pri commite vetvy.

Google Drive sme používali ako tímové úložisko, kde sme uchovávali dokumentáciu, záznamy zo stretnutí a rôzne iné dokumenty potrebné pre tím. Rovnako sme tu mali uložené datasety pre náš projekt.

Menej často sme používali Skype alebo Gmail, na ktorom sme mali založený tímový alias odkiaľ sa pošta preposielala každému členovi.

## 3.2 MANAŽMENT VÝVOJA A INTEGRÁCIE

V našom projekte je úloha manažéra vývoja a manažéra integrácie rozdelená medzi dvoch členov. Ich úlohy pozostávajú z manažmentu verzií, integrácie súčastí do jedného veľkého celku ale aj konfigurácia systému a iné. Manažér integrácie je priamo zodpovedný za repozitáre v TFS a udržiava vetvy projektu. Rieši konflikty a pridáva novú funkcionálnosť do hlavnej vetvy. Viac o manažovaní projektových vetiev a postup ako ich v projekte 3Drecon udržiavame sa nachádza v dokumente Metodika verziovania.

Aby sme dokázali udržiavať prehľadný kód a projekt používame TFS (Team Foundation Server) kde medzi sebou zdieľame vetvy projektu, manažujeme

úlohy aj zapisujeme bugy. Tento nástroj je priamo integrovaný do Visual Studia, ktoré používame na vývoj a preto je pre nás TFS najlepšou možnou alternatívou. Nástroj ponúka jednoduchú synchronizáciu a aktualizovanie zmien, ktoré priamo prenáša medzi našim úložiskom aj s prípadnými správami.

Každý z členov tímu je zodpovedný za časť systému, ktorú vyvíja teda prípadné nedostatky a chyby v týchto častiach musí primárne vyriešiť jej tvorca. Tvorca nie je zodpovedný iba za funkcionálnu stránku ale aj za výstup a úpravu, teda za celkové spracovanie (správne okomentovaný a zadokumentovaný kód). Následná integrácia do projektu je vykonaná integrátorom. Tvorca s ním túto skutočnosť odkomunikuje podľa pravidiel manažmentu komunikácie a poskytne integrátorovi potrebnú súčinnosť pri práci.

### **3.3 MANAŽMENT DOKUMENTÁCIE**

Písanie dokumentácie v našom tíme prebieha priebežne. Dokumentácia je vytváraná spôsobom spoločného dokumentu, typu GOOGLE DOC, ktorý máme uložený na spoločnom úložisku GOOGLE DRIVE. Nakoľko dokumentácia sa skladá z viacerých častí, na GOOGLE DRIVE sú postupne pridávané dokumenty, ktoré si následne prejdú viacerí členovia tímu a po spoločnej konzultácii sa dohodnú, či je dokument vyhovujúci alebo je nutné ho prerobiť. V prípade, že je dokument vyhovujúci je následne zaintegrovaný do finálnej dokumentácie. Aby sme predišli rôznym kolíziám typu odlišné riadkovanie alebo font písma, dohodli sme sa dodržiavať pravidlá písania dokumentácie, ktoré sú spísané v dokumente menom Metodika písania dokumentácie.

### **3.4 MANAŽMENT PLÁNOVANIA**

Predpokladom úspešného plánovania je naplnený tzv. product backlog aspoň na 2 šprinty vopred. Product backlog sa vytváral na základe požiadaviek nášho zadávateľa, ktorý je v tomto prípade zároveň náš vedúci tímu. Vzhľadom na požiadavky, boli hierarchicky vytvorené časti softvéru (Epics, Features), pre ktoré sa vytvárali backlog itemy. Tie sa radia vždy podľa priority – dôležitosť s akou treba daný backlog item spraviť, aby sme naplnili požiadavky čo najefektívnejším spôsobom.

Proces plánovania prebieha na začiatku každého šprintu. Podľa priority sú ku šprintu priradené backlog itemy, na ktorých sa pracuje počas najbližšieho

šprintu. Backlog item má určité kritéria akceptácie, ktoré musia byť dodržané, aby bola úloha splnená. Na naplnenie týchto kritérií je väčšinou potrebné viacero úloh (taskov).

Následne sa ohodnotia jednotlivé úlohy vzhľadom na prvú ohodnotenú úlohu, ktorá bola vopred určená (base). Na každom stretnutí sa hrá tzv. *Scrum Poker*, pri ktorom sa odhaduje náročnosť úloh pomocou hlasovania, čo sprevádza diskusia o úlohách o ktorých náročnosti sa hlasuje. Touto aktivitou je zabezpečený čo najlepší odhad k jednotlivým úlohám. Vzhľadom na minulé šprinty prispôsobujeme náročnosť ďalšieho šprintu, preto je veľmi dôležité zabezpečiť čo najlepší odhad.

Následne sú zadané úlohy do manažovacieho nástroja TFS, ktoré sú buď priradené jednotlivým členom tímu, alebo si ich členovia tímu vyberajú počas šprintu. Progres v jednotlivých úlohách sa taktiež sleduje v nástroji TFS.

### **3.5 MANAŽMENT RIADENIA**

Riadenie v našom tíme bolo založené na agilnej metóde vývoja SCRUM. V časti manažment plánovania sa môžete dozvedieť ako sme pomocou tejto metódy plánovali postup v našom projekte a ako sme určovali úlohy. Tieto úlohy sme potom evidovali v nástroji TFS. V každom šprinte sme vytvorili definované backlog items a k nim patriace úlohy. Každý člen mal definované svoje úlohy a ich hodnotu (ako sa táto hodnota získava nájdete v časti manažment plánovania). Niektoré úlohy nemali pridelenú zodpovednú osobu nakoľko nebolo jasné kto to bude robiť a nebolo ani potrebné to definovať. Dohodnuté však bolo že sa to musí urobiť a ten kto mal najviac času takúto úlohu pripísal v TFS sebe a urobil ju.

Pre úlohy existujú tri základné stavy: To do, In progress, Done. Logicky je cieľom, každú z definovaných úloh dopracovať do štádia done. Ak by sa však niektoré úlohy v šprinte nestihli urobiť, presúvajú sa do ďalšieho šprintu kde majú najvyššiu prioritu.

Dohľad nad vykonávaním úloh a nad ich rozdeľovaním má vedúci tímu. Ten je zodpovedný za správne rozdelenie úloh, prípadne riešenie problémov v tejto oblasti.



## 4 SUMARIZÁCIA ŠPRINTOV

### 4.1 ŠPRINT 1

V prvom šprinte sme sa primárne snažili urobiť všetko podľa odporúčaní odborníkov. Narazili sme však na niekoľko problémov, ktoré sa odzrkadlili aj na jeho kvalite. No práve vďaka týmto chybám teraz vieme omnoho lepšie ako na šprintoch pracovať. Veľkým problémom bolo, že sme sa museli naučiť fungovať ako tím a dokázať tak zvýšiť efektivitu našej práce. Druhým problémom bolo, že pokračujeme v projekte po minuloročnom tíme a na začiatku sme mali minimálne skúsenosti a znalosti v oblasti, ktorej sa projekt venuje. Preto bolo potrebné dlhé analyzovanie, pretože bez neho nebolo možné ďalej implementovať novú funkcionálnosť. Fakt, že celý projekt je výskumného charakteru túto situáciu sťažoval ešte o trochu viac. So SCRUM-om sme v tíme skúsenosti nemali a s jeho použitím na výskumnom projekte už vôbec nie.

Začali sme používať nové nástroje ako SLACK na komunikáciu a TFS na evidenciu úloh. Nejaký čas nám trvalo aj naučiť sa pracovať s týmito nástrojmi, hlavne s TFS.

V prvom šprinte sme definovali 4 backlog items.

1. Zrýchlenie výpočtov ukladaním medzivýsledkov
2. (bug) Slabá viditeľnosť procesov
3. Analýza celkového kódu
4. (bug) Zlyhanie programu pri načítaní konfiguračného súboru (.xml)

### 4.2 ŠPRINT 2

Pri retrospektíve prvého šprintu sme sa všetci členovia tímu jednohlasne zhodli, že našou najväčšou slabinou bolo TFS a jeho správne používanie, čo sme sa rozhodli do ďalšieho šprintu zmeniť. V druhom šprinte sme sa po dohode s vedúcou projektu a s členom minuloročného tímu Lukášom Hudecom rozhodli pustiť do reštruktulizácie celého programu. Vedeli sme, že tento krok nás spomalí v napredovaní rozširovania funkcionálnosti ale na druhej strane sme vedeli, že ak si to navrhne a prerobíme podľa seba, bude to mať pozitívny účinok do budúcnosti. Najprv sme vytvorili návrh class diagramu nového projektu, ktorý sme následne diskutovali s vedúcou cez Skype hovor. Na konci hovoru, po odsúhlasení štruktúry projektu sme si rozdelili úlohy a postupne sme na nich začali pracovať. Počas šprintu sme narazili na zopár problémov pri rozbehávaní projektu na notebookoch, ktoré sa nám podarilo našťastie vyriešiť. Na konci šprintu sme zistili, že sme v stanovenom čase

nedokázali spraviť všetku prácu, ktorú sme si naplánovali a ,že v tom budeme musieť pokračovať aj v treťom šprinte.

## 4.3 ŠPRINT 3

Pri retrospektíve druhého šprintu sme zistili, že naša práca s TFS je už na vyššej úrovni ako to bolo pri prvom šprinte no stále sme nerobili niektoré veci tak ako by sme mali respektíve ako by sme si predstavovali. Zistili sme, že pri taskoch nezadáваме správne hodnoty v story pointoch a, že to má výrazný vplyv na nežiadúci výzor nášho Burndown chart diagramu. Taktiež sme zhodnotili že by sme mali v TFS riešiť aj neprogramátorske úlohy a mohli by sme vylepšiť aj spôsob robenia nášho code review. Tieto problémy sme si na stretnutí vydiskutovali a navrhli sme vhodné riešenie na ich odstránenie. Pri vytváraní Backlog Items pre tretí šprint sme takmer všetok čas nášho šprintu investovali do dokončenia úloh z predchádzajúceho šprintu, no taktiež sme už načrtli víziu do budúcnosti a identifikovali sme si ciele ako vyriešiť outliers a taktiež možnosť pracovať na našom projekte pomocou Kinectu. Pri hraní Scrum Pokeru sme si určili vzorový task na základe, ktorého sme pridelovali hodnoty ostatným úlohám. V rámci tretieho šprintu bolo nutné dokončiť dokumentáciu.

## 5 METODIKY

### 5.1 METODIKA ZADÁVANIA ÚLOH DO TFS

V rámci tohto projektu (3dRecon) sa úlohy manažujú v 3 fázach:

1. *Tvorba Backlogu*
2. *Plánovanie šprintu*
3. *Práca na taskoch počas šprintu*

Manažment úloh sa v tomto prípade týka **Backlog Items**, **Bugs** a **Tasks**. Epics a Features nie sú opísané, keďže sa s nimi manipuluje iba zriedka.

**Najdôležitejšia časť** metodiky pre členov tímu je *Práca na taskoch počas šprintu*.

SLOVNÍK

BLI - Backlog Item

PO - Product Owner

### 5.1.1 TVORBA BACKLOGU

Tvorba Backlogu je činnosť, ktorá prebieha stále. Vytvoriť nový Backlog Item alebo Bug je možné hocikedy a nemusí sa čakať s vytvorením na konzultáciu s ostatnými členmi tímu alebo s Product Ownerom, aj keď práve PO dáva najviac podnetov na BLI vzhľadom na tému.

#### Životný cyklus BLI

1. *Vytvorenie BLI*
2. *Schválenie BLI Product Ownerom*
3. *Odhadnutie Effortu BLI*
4. *Práca na BLI*
  - 4.1. *Priradenie do sprintu*
  - 4.2. *Odloženie (pozastavenie) BLI*
5. *Dokončenie BLI*

(body 4 a 5 sú opísané v kapitolách *Plánovanie šprintu* a *Práca na taskoch počas šprintu*)

#### 5.1.1.1 VYTVORENIE BACKLOG ITEMU

Aj keď je možné vytvoriť BLI hocikedy je dôležité aby sa zadali všetky nižšie popísané atribúty BLI-u aby sa neskôr nebrzdili procesy zadávaním týchto atribútov. Tiež v čase ich tvorby ich vieme opísať rýchlo, lebo sa nemusíme rozpamätávať.

#### **Kroky tvorby Backlog Itemu:**

1. **Vhodne nazvať BLI - Title** - Názov Backlog Itemu by mal byť dostatočne popisný aby vystihoval prácu ktorú predstavuje, pričom by nemal obsahovať názvy tried, funkcií atď. pokiaľ nie sú nevyhnutné pre popis aby sme neobmedzovali tasky vo voľbe implementácie. (*miesto názvu funkcie alebo triedy skôr použiť opis funkcionality*). Zlý názov “Vytvorenie triedy Exporter” - Dobrý názov “Vytvorenie exportu segmentačných dát”
2. **Pridať “Description” - Description** - Popis by mal byť natoľko podrobný, aby umožnil členom tímu odhadnúť *effort* a určiť tasky na jeho implementáciu.
3. **Určiť “Acceptance Criteria” - Acceptance Criteria** - Kritérium akceptácie reprezentuje “*Definition of done*”. Slúži na verifikáciu dokončenosti BLI. Mala by byť definovaná minimálne 1 akceptačná podmienka.

4. **Určiť “Value area”** - Ak sa BLI týka zmeny štruktúry programu a nepridáva funkcionality tak je *Architecture*, inak *Business*.
5. **Určiť “Priority”** - Pri určení priority je dobré vychádzať z priorit ostatných BLIs a určiť ju vzhľadom na ne.
6. **Priradiť ku “feature” (pridať vzťah patent)**
7. **Priradiť predchodcov (pridať vzťah successor)** - Je dôležité priradiť predchodcu BLI, aby sa pri plánovaní šprintu vedelo, či bude možné BLI implementovať (či nebude musieť BLI čakať na predchodcu).
8. **Usporiadať BLI podľa priority**

V rámci Backlog treba premiestniť nový BLI tak, aby boli Backlog Items v zozname usporiadané podľa priority.

#### 5.1.1.2 SCHVÁLENIE BACKLOG ITEMU PRODUCT OWNEROM

Po schválení BLI Product Ownerom (väčšinou na stretnutiach) je potrebné zmeniť jeho stav z *New* na *Approved*. Ak sa BLI vytvára na podnet Product Ownera, tak po jeho vytvorení tiež treba zmeniť stav na *Approved*.

#### 5.1.1.3 ODHADNUTIE EFFORTU BACKLOG ITEMU

Odhadnutie effortu BLI by sa malo vykonať na tímových stretnutiach pomocou *planning pokeru*, pričom sa môže upraviť aj jeho popis ak si ho počas *planning pokeru* členovia tímu lepšie upresnia.

### 5.1.2 PLÁNOVANIE ŠPRINTU

Pred začatím plánovania šprintu by mali mať všetky BLIs **určený effort**, aby sa vedelo podľa *velocity* z predošlého šprintu odhadnúť aké BLIs vybrať aby sa stihli spraviť do konca šprintu.

#### ***Kroky plánovania šprintu:***

1. **Určiť Backlog Itemy pre nasledujúci šprint.** - Backlog itemy pre nasledujúci šprint sa určujú na základe *velocity* z predošlého šprintu a závislosti Backlog Itemov (niektoré môžu byť vykonané až po druhých)
2. **Určiť a vytvoriť tasky pre Backlog Itemy**

2.1. *Title* - Názov tasku by mal konkrétne popisovať úlohu, ktorá sa ma vykonať. V prípade *Development* tasku by sa mali používať aj názvy funkcií a tried.

2.2. *Description* - Popis tasku **netreba** pokiaľ je názov dostatočne opisný.

2.3. *Activity* treba určiť podľa typu tasku

### 3. Odhadnúť *Remaining Work* tasku

### 4. Určiť vzťahy medzi taskmi

4.1. *Predecessor/Successor* - potrebné pre všetky typy taskov, ak sa dá

4.2. *Related* - vhodné pre tasky ako *REVIEW* a pod. ktoré môžu bežať súbežne a ovplyvňujú sa (súvisia) istým spôsobom

4.3. *Tests/Tested By* - tasky typu *Testing* (Activity) majú vzťah *Tests* s taskami, ktoré testujú.

## 5.1.3 PRÁCA NA TASKOCH POČAS ŠPRINTU

Počas šprintu sa vykonávajú nasledujúce aktivity:

1. **Práca na tasku** - Pri začatí práci na tasku sa člen tímu priradí do daného tasku pokiaľ tak už neurobil a presunie task (zmení mu stav) do *In progress*.
2. **Dokončenie práce na tasku** - Ak sa dokončí práca na tasku je nutné ho presunúť do *Done*. Tiež sa skontroluje Backlog Item tasku, či má dokončené všetky tasky. Ak áno a sú splnené ***Acceptance Criteria***, tak sa nastaví stav Backlog Itemu na *Done*.
3. **Obnovenie práce na tasku** - V prípade ak sa ukáže, že task nie je dokončený, napr. ak neprejde cez testy, zmení sa jeho stav späť na *In progress*.
4. **Objavenie chyby** - Ak sa objaví chyba vytvorí sa nový *Bug* do Backlogu a podľa závažnosti a potreby sa môže pridať do aktuálneho šprintu. Pri jeho vytváraní sa nastaví atribúty *Severity*, *Effort*, *Remaining Work* a *Activity*. (atribúty *Effort* a *Remaining Work* sa nastaví na rovnakú hodnotu)
5. **Pridávanie potrebných taskov** - Počas šprintu sa často stáva, že sa zistí, že treba vytvoriť nový task, lebo pri plánovaní sa s ním nerátalo, alebo sa objavil bug a treba vykonať tasky na jeho opravu. V takých prípadoch sa môže vytvoriť task a pridať do BLI alebo Bugu pre daný šprint. Nevadí,

že Burdown Chart stúpne, pretože podľa toho ako často a o koľko stúpne sa vie vyhodnotiť, ako dobre sa plánuje, prípadne ako často nastávajú chyby.

## 5.2 METODIKA PREHLIADOK KÓDU

Tento dokument pojednáva o spôsobe vykonávania prehliadok naším tímom. Každý *pull-request* musí prejsť schválením, ktorému predchádza prehliadka kódu.

### 5.2.1 ZÁKLADNÉ USTANOVENIA

1. Schváleniu predchádza prehliadka kódu, ktorú vykonáva vopred určený člen tímu. V nevyhnutných prípadoch je možné, aby prehliadku spravil niekto iný, avšak je potrebné dostatočne vopred oznámiť, že prehliadku daný človek nebude môcť vykonať, pričom treba uviesť relevantný dôvod.
2. Nikto by nemal robiť prehliadku pre kód dlhší ako **400** riadkov kódu. V prípade, že je kódu viac, je potrebné, aby bola prehliadka vykonaná viac ako jedným členom.
3. Prehliadka by mala trvať maximálne **1 hodinu**.

### 5.2.2 PRIEBEH PREHLIADKY

Pri prehliadke je potrebné pozrieť sa na kód z viacerých pohľadov, za účelom odhaliť čo najviac chýb a tak zabezpečiť vyššiu kvalitu kódu.

#### ARCHITEKTÚRA

- **Single responsibility principle** - každá trieda je zodpovedná za **práve** jednu konkrétnu vec. Tento princíp je možné uplatniť aj pri metódach. Ak metóda robí 2 veci, nie je to správne.
- **Open/Closed principle** - každá trieda musí byť otvorená pre rozširovanie (dá sa rozširovať, je možné pridávať novú funkcionálnosť...), ale uzavretá pre modifikáciu (existujúca implementácia, je možné použiť inými triedami/modulmi)
- **Duplikácia kódu** - ak si všimneme duplikovaný kód, treba zvážiť či nie je možné vytvoriť metódu, ktorú je potom možné využiť na viacerých miestach.

- **Error handling** - používané metódy je potrebné ošetriť proti zlyhaniu (napr. načítanie .pcd súboru - *FileNotFoundException* - je ošetrená?). Inými slovami, vždy musí byť alternatívny scenár, ktorý sa vykoná v prípade neúspechu primárneho priebehu.
- **Efektívnosť** - treba sa zamyslieť či daný algoritmus nie je možné urýchliť. Napr. iterovanie celého listu v prípade, že to nie je potrebné (hash tabuľka, dictionary apod.)
- **Potenciálne chyby** - táto časť je jednoznačne najťažšia, ale je potrebné sa zamyslieť aj nad ňou. Každý kód by mal prejsť samozrejme testovaním, avšak, je možné, že sa nachádza v kóde nejaká logická chyba? Ak si ju všimneme je výborné nato poukázať.

## ŠTÝL

Vzhľadom na skutočnosť, že pokračujeme v minuloročnom projekte, je vhodné aby sme dodržiavali code conventions, ktoré boli zavedené minulý rok. Táto metodika sa nachádza v minuloročnej dokumentácii a je možné si ju prezrieť. Pri prehliadke postupujeme preto podľa tejto metodiky.

V minuloročnej metodike sú konkrétne návody ako písať kód. Uvediem preto zopár konvencií, ktorých je všeobecne správne sa držať a je vhodné nedostatky týchto konvencií hľadať pri prehliadkach kódu.

- **Mená metód a premenných** - mená musia byť jednoznačné. Ak napadne pri prehliadke niekomu lepšie pomenovanie, nebojte sa napísať svoj návrh.
- **Dĺžka metódy** – 50 riadkov
- **Dĺžka triedy** – 200-300 riadkov
- **Komentovaný kód** - komentovaný kód nemá čo hľadať pri žiadosti o merge novú funkcionality. Ak chcete dať návrh, že niečo treba spraviť, niečo inak atď. Napíšte to do komentára a diskutujte to s ostatnými.
- **Komentáre** - sú dostatočne okomentované všetky metódy, premenné atď? Ak nie, treba nato upozorniť. Je veľmi dobré komentovať aj parametre, ktoré do metódy vstupujú, nie len samotnú funkcionality.

### 5.2.3 TESTING

- **Testy** - pre každú funkcionality, musí byť dokončené testovanie. Ak nie je, treba počkať dokedy bude.
- **DFD** - spĺňa kód *Definition of Done* (zabezpečuje funkcionality, ktorú má?)

## 5.2.4 AKO ROBIŤ PREHLIADKU

1. Pýtajte sa
2. Diskutujte (označte tagom v *tfs* daného človeka)
3. Argumentujte
4. Dajte návrhy nato, ako danú vec zlepšiť
5. Nezapudnite pochváliť, ak je to zaslúžené :)

Na záver ešte jedno zlaté pravidlo.

**Skúste vždy pred pull requestom spraviť code review sám sebe.**

## 5.3 METODIKA VERZIOVANIA KÓDU

Na manažment verzií využívame TFS od spoločnosti Microsoft, ktorý je priamo integrovaný do Visual Studia.

SLOVNÍK

- **Branch** – vetva s projektovým kódom, prípadne jeho časťou.
- **Commit** – odoslanie zmien do úložiska.
- **Merge** – Zlúčenie vetiev kódu, vyriešenie konfliktov.
- **Konflikt** – vznikne v prípade že rovnaká vetva bola na rovnakom mieste upravená dvoma spôsobmi, resp. v dvoch vetvách.
- **Pull-Request** - teda získanie zmien zo vzdialeného repozitára a ich zamergovanie do hlavnej vetvy.

### 5.3.1 BRANCHES

Projekt je vetvený do 2 hlavných vetiev (*branches*).

- **master** – vetva obsahujúca iba skontrolovanú a overenú funkcionálnosť. Je to hlavná projektová vetva. O túto vetvu sa stará integrátor projektu a iba on môže pripojiť do tejto vetvy novú funkcionálnosť. Z tejto vetvy sa aplikácia nasadzuje.
- **dev** – je to developerská vetva určená pre verziovanie kódu. Do takejto vetvy je možné poslať iba funkčný kód po teste programátorom. Do tejto vetvy môže pridať funkcionálnosť každý developer, musí však dodržať pravidlo, že všetka funkcionálnosť



bude bez problémov fungovať aj s ostatnou v tejto vetve. Ostatný developeri teda nebudú mať problém použiť ju pri práci na svojej úlohe.

- **Pomocné vetvy** – každý developer môže vytvárať svoje pomocné vetvy a pracovať v nich. Každá vetva sa však vytvára za účelom práce na konkrétnom tasku alebo backlog iteme. Avšak ak chce túto vetvu pridať do nadradenej, musí dodržať určité pravidlá spísané nižšie v dokumente. V prípade, že celý tím bude pracovať na konkrétnom okruhu funkcionality, môže sa vytvoriť dočasná developerská vetva, z ktorej si všetci urobia svoju vetvu a pri dokončení sa tieto vetvy spoja. V jednej vetve pracuje jeden človek. Výnimkou však môžu byť určité kedy bude v jednej vetve pracovať viac ľudí ale je potrebný ďalší manažment.

V prípade, že sa developer rozhodne pridať novú funkcionality do master vetvy musí mať kód patrične zdokumentovaný podľa pravidiel dokumentovania. Kód musí byť okomentovaný a otestovaný. Ak je všetko splnené informuje o tom integrátora. Ten projekt skontroluje a ak bude všetko v poriadku spojí vetvy, ak nie, ohlási nájdené chyby developerovi a ten ich musí opraviť.

Medzi vetvami je možné prepínať sa iba v prípade, že všetky zmeny boli odoslané alebo ak majú vetvy rovnakú verziu. Ak nemajú rovnakú verziu nebude možné prepnúť sa medzi vetvami.

### 5.3.2 PULL-REQUEST

Pull-Request vetvy podlieha nasledujúcim pravidlám:

1. Funkcionalita vetvy je otestovaná a nenašli sa žiadne chyby.
2. Vetva je bez problémov skompilovateľná.
3. Nadradené sú vždy pravidlá vetvy, do ktorej sa pod-vetva pridáva.
4. Ďalšie pravidlá sú definované v dokumente manažment prehliadok, ktorým musí prejsť každá nová funkcionality.

Každý Pull-Request podlieha procesu schvaľovania tzv. *code review*.

#### **Nikto si nesmie svoj Pull-Request schváliť sám!**

Požiadavka o kontrolu, teda nový Pull-Request sa objaví v komunikačnom nástroji. Jeden z developerov (nie autor) si vezme úlohu z boardu ohľadne schvaľovania tejto vetvy a skontroluje či je všetko splnené. Ak áno povolí Pull-Request, ak nie oznámi autorovi pridaním komentáru, prípadne aj v komunikačnom nástroji, aké našiel chyby a nedostatky a ten je povinný ich napraviť.

Ak pri Pull-Requeste nastanú kolízie je potrebné informovať o tom integrátora, ktorý ich vyrieši a urobí merge vetiev.

### 5.3.3 COMMIT

Commit je vlastne nahranie zmien do repozitára, čím vzniká nová revízia v histórii. Všetky zmeny urobené v počítači konkrétneho developera sa zapíšu do úložiska. Každý commit obsahuje správu ktorá pozostáva zo zoznamu zmien vo funkcionalite spolu s ich krátkym opisom, novú funkcionalitu s opisom a prípadné nezrovnalosti alebo možné budúce chyby a problémy. Zo správy musí byť jasné čo je nové a prečo to bolo vytvorené.

### 5.3.4 SÚHRN PRAVIDIEL ODOSIELANÉHO KÓDU

Kód, ktorý chce developer pridať do nadradenej vetvy podlieha nasledujúcim pravidlám.

Odosielala sa výlučne:

- fungujúci kód
- otestovaný kód
- okomentovaný kód
- stabilný kód

## 5.4 METODIKA TESTOVANIA

Tento dokument určuje spôsob akým je nutné testovať funkcionalitu pred tým ako sa zmena v projekte schváli a integruje sa do nadradenej branch. Keďže pracujeme na výskumnom projekte, pre ktorý nie je možné efektívne vyvinúť automatické testovanie, je nutné aby testovanie prebiehalo na čo najľahšie testovateľných častiach.

POJMY

- **Unit testovanie** – unit testovanie, alebo testovanie komponentov je testovanie, ktoré sa zameriava na špecifickú časť kódu, resp. konkrétnu funkcionalitu
- **Integračné testovanie** – cieľom integračného testovanie je overenie správnej interakcie jednotlivých modulov systému.

- **White box testovanie** – Testovania softvéru zamerané na vnútornú logiku a štruktúru systému.

#### 5.4.1 ZÁKLADNÉ PRAVIDLÁ

- Testovanie je nutné vykonať pre každý pull request
- Každý programátor je zodpovedný za testovanie svojej pridanej funkcionality
- Je nutné vykonať unit test na pridanú funkcionality
- Po unit testoch programátor vykoná integračné testovanie
- Po úspešnom otestovaní môže pokračovať vo vytváraní pull request

**Programátor nesmie vytvoriť pull request bez otestovania svojej práce.**

#### 5.4.2 PRIEBEH TESTOVANIA

Testovanie prebieha v podobe unit testov, ktoré vykonávajú samotný programátori. Pre každý task z backlog si zodpovedný programátor vytvorí branch. V rámci vytvorenej branch pracuje na tasku a po dokončení programovania a okomentovaní kódu otestuje svoju prácu. Testovanie prevedie hlavne na svoju časť kódu, ale taktiež otestuje celú integritu kódu po vytvorenej zmene.

## 5.5 METODIKA PÍSANIA DOKUMENTÁCIE

Tento dokument slúži ako návod na písanie dokumentácie.

### 5.5.1 ZÁKLADNÉ POŽIADAVKY

Dokumentácia je vytváraná v nástroji MS WORD(Microsoft Word) od spoločnosti Microsoft, tento nástroj je zahrnutý v balíku MS Office, ktorý musí byť licencovaný. Dokumentácia musí byť písaná v slovenskom jazyku a v prípade anglických výrazov je nutné aby boli v dokumente preložené. Dokumentácia musí byť písaná pomocou diakritiky. V prípade, že dokonale neovládate slovenský pravopis alebo ho ovládať nechcete využívajte pomocný nástroj na kontrolu pravopisu, ktorý je integrovaný v MS WORD.

### 5.5.2 ZÁKLADNÉ PRAVIDLÁ

- Font písma – Arial
- Veľkosť písma – 11
- Farba písma - čierna
- Riadkovanie – 1,5
- Hlavný nadpis – na titulný nadpis dokumentu používajte nadpis typu Názov
- Podnadpis – na podnadpis dokumentu používajte nadpis typu Nadpis 1
- Zoznam – v prípade zvolenia zoznamu využívajte odrážky
- Obrázky – Obrázok musí mať vždy priradený popis a musí byť aj správne očíslovaný
- Diagramy – Diagram musí mať vždy priradený popis a musí byť správne očíslovaný

- Tabuľka – Tabuľka musí mať vždy priradený popis a musí byť správne očíslovaná
- Popis – Popis k obrázkom, diagramom a tabuľkám musí byť v tvare objekt. Číslo – opis
- Umiestnenie popisu – popis musí byť umiestnený vždy pod objektom, v prípade, že sa nezmestí je nutné objekt s obrázkom vložiť na ďalšiu stranu
- Kapitola – každá nová kapitola musí začínať na novej strane, je nutné aby pred ňou bol zlom strany
- V prípade odkazovania sa na nejaký dokument je potrebné aby odkaz obsahoval hyperlinku

### 5.5.3 SPRÁVA DOKUMENTÁCIE

Každý dokument musí byť nahraný na spoločné úložisko Google Drive, aby bola prístupná aj ostatným členom tímu, Na tomto úložisku sa nachádza finálny dokument, do ktorého bude váš dokument v prípade, že je vyhovujúci zaintegrován. Dokument na Drive nazývajte spôsobom čosomspravil\_ktotospravil. V prípade nejasností kontaktuje manažéra dokumentácie, ktorý s vami bude problémy riešiť.

## 6 GLOBALNA RETROSPEKTIVA

V rámci celkovej retrospektívy sme do tohto bodu semestra narazili na viacero problémov, ktoré sme sa snažili riešiť v každom šprinte. Celkovo môžeme vytýčiť niekoľko skutočností, ktoré ovplyvnili vývoj nášho tímu najviac.

1. **Práca s TFS** – jednoznačne jeden z najväčších problémov. Keďže sme doteraz nemali možnosť pracovať v tíme pozostávajúcom z viac ako dvoch ľudí, je to pre nás nová skúsenosť. Nemali sme žiadne skúsenosti s používaním systémov na manažment úloh. Taktiež spadajú pod tento problém aj iné náležitosti, ako napr. tvorba backlogu či zadávanie estimate-u pri taskoch a vlastne celkové poznatky v oblasti manažmentu úloh.

V rámci 3. šprintu vidieť značný progres v tejto časti. Každý pracuje s tfs, interaguje s tabuľou, zadáva bugy a pod. Výsledok je vidieť na burn-down charte, ktorý naberá správny spád a taktiež aj na velocity, ktorá sa postupne zvyšuje.

2. **Plánovanie** – v rámci plánovania šprintu sme učinovali viacero chýb. Nemali sme vytvorený backlog, z ktorého by sme len podsúvali podľa priority item-y do šprintov. Bolo to ovplyvnené hlavne tým, že pokračujeme po minuloročnom tíme a nevedeli sme presne kam sa chceme pohnúť a navyše je projekt výskumného charakteru, čo znamená, že backlog sa nám neustále mení vzhľadom na výsledky jednotlivých prototypov. Planning poker sme nehrali korektne a nezapisovali sme celý jeho opis rovno do description pri jednotlivých taskoch.

V rámci 3. šprintu sme v tejto časti spravili jednoznačne najväčší krok vpred. Aj keď je pravdou, že výskumný projekt nie je možné naplánovať na pol roka dopredu, lebo nikdy nie je isté čo-ako dopadne. Napriek tomu sme sa posnažili naplniť backlog minimálne na najbližšie 3 šprinty. Z týchto item-ov sme podľa priority vybrali tie s najvyššou a presunuli sme ich do šprintu. Je cítiť oveľa väčší prehľad v tom čo robíme.

Planning poker v 3. šprinte bol jednoznačne prínosnejší ako tie predtým. Hlasovalo sa o všetkých taskoch a diskusia sa písala rovno do ich opisu. Brainstoriming zabezpečil lepší odhad a taktiež aj presný opis toho čo je treba v budúcnosti urobiť.

V neposlednom rade môžeme vyzdvihnúť našu komunikáciu. Od začiatku používame *Slack*, štruktúrujeme komunikáciu na jednotlivé témy a všetci sa aktívne zapájame. Taktiež využívame *Google Drive* na zdieľanie jednotlivých dokumentov. Okrem iného, sa stretávame osobne, mimo nášho oficiálneho stretnutia – to hodnotíme kladne a určite v tom chceme pokračovať aj naďalej.

## **7 EXPORT ÚLOH**



Projekt: 3Drecon Server: 3drecon.visualstudio.com\3drecon Query: 3Drecon Team - Sprint 1 - Backlog List type: Tree						
IP	Work Item Type	Title 1	Title 2	Step	Assigned To	Remaining Work
19	Product Backlog Item	Zrýchlenie výpočtov ukladaním medzivýsledkov		Done	Mario Hunka	
10	Task	Pridanie checkboxu pre export medzivýsledkov		Done	Mario Hunka	
7	Task	Ukladanie medzivýsledkov do formátu .pcd		Done	Mario Hunka	
31	Task	Načítavanie .pcd uložených medzivýsledkov.		Done	Mario Hunka	
21	Bug	Slabá viditeľnosť		Done	Jakub Ginter	
13	Task	Vytvorenie loading screenov		Done	Jakub Ginter	
8	Product Backlog Item	Analýza celkového kódu		Done		
11	Task	Vyhľadanie nástroja na zapisovanie chýb		Done	Mario Hunka	
14	Task	Analýza modulu 3DReconstruction/Methods		Done	Viktor Košťan	
22	Task	Analýza modulu 3DReconstruction/Objects		Done	Jakub Ginter	
23	Task	Analýza modulu 3DReconstruction/Primitives		Done	Jakub Ginter	
26	Task	Analýza zvyšných tried v module DataHandling		Done	Mario Hunka	
28	Task	Analýza modulu gui/GUI		Done	Mario Hunka	
29	Task	Analýza modulu gui/Widgets		Done	Mario Hunka	
39	Task	Prefarbenie bodov		Done	Jakub Ginter	
40	Task	odstánenie zbytočných kopírovaní PointCloud		Done	hascicm.mh	
25	Task	Analýza modulu		Done	hascicm.mh	
9	Bug	Zlyhanie programu pri načítaní		Done	Viktor Košťan	
32	Task	Implementovanie bezpečného načítania konfiguračných		Done	Viktor Košťan	

Project: 3Drecon Server: 3drecon.visualstudio.com\3drecon Query: 3Drecon Team - Sprint 2 - Backlog List type: Tree						
ID	Work Item Type	Title 1	Title 2	Assigned To	Description	Remaining Work
75	Product	Refaktoring tried segmentácie		Richard Pinter		
76	Task		Identifikácia častí kódu týkajúcich sa segmentácie	Richard Pinter		
77	Task		Osamostatnenie kódu týkajúceho sa segmentácie	Richard Pinter		
74	Product	Vytvorenie abstraktnej triedy pre Segmentáciu		Richard Pinter		
81	Task		Vytvorenie triedy pre segmentáciu	Richard Pinter		
72	Product	rekonštrukcie pre nový model		hasicim.mh		
83	Task		prepracovanie triedy do nového projektu	hasicim.mh		
97	Task		objektov do nového projektu	hasicim.mh		
71	Product	refaktoring tried rekonštrukcie		hasicim.mh		
84	Task		refaktoring tried	hasicim.mh		
65	Product	Vytvorenie nového projektu Interior3DRecon		Viktor Košťan		
66	Task		Vytvorenie a nastavenie projektu Interior3DRecon	Viktor Košťan	projektov 3Dreconstruction, DataHandling a Exporter.	
51	Product	Refactoring načítania dát zo súborov		Viktor Košťan		
82	Task		Premiestnenie DataReader do Interior3DRecon	Viktor Košťan		
50	Product	Refactoring triedy ClosedSpace		Viktor Košťan		
67	Task		Vytvorenie základnej verzie triedy ClosedSpace	Viktor Košťan		

Product Backlog Item	Vytvorenie projektu pre GUI	Mario Hunka	rozhranie. Preto je potrebné vytvoriť nový projekt pre GUI a nastaviť všetko tak, aby aplikácia bežala bez problémov.
Task	Integrovanie QT do projektu	Mario Hunka	iný framework -QT. Z čoho vyplýva, že je nutné nastaviť cesty ku zdrojovým súborom QT. Taktiež nainštalovať addon do MCVS 2013, pomocou, ktorého je možné prepojiť QT Creator a MCVS.
Task	projektu s ostatnými projektami	Mario Hunka	Nastavenie dependencies pre GUI (iné moduly, ktoré GUI používa)
Task	Vytvorenie základného používateľského rozhrania	Mario Hunka	Vytvorenie triedy pre gui a základného template.
Task	Vytvorenie nového GUI pomocou QT	Mario Hunka	Design nového GUI.
Product Backlog Item	Prepojenie GUI s projektom	Mario Hunka	Je potrebné zabezpečiť prepojenie GUI s logickou časťou projektu.
Task	EventHandler pre rekonštrukciu	Mario Hunka	Handler pre elementy, ktoré súvisia s rekonštrukciou scény.
Task	EventHandler pre vizualizáciu	Mario Hunka	Handler pre elementy, ktoré súvisia s vizualizáciou.
Task	EventHandler pre načítanie súboru	Mario Hunka	Handler pre elementy, ktoré súvisia s načítaním/exportovaním súboru.
Task	Vytvorenie novej triedy pre vizualizer	Jakub Ginter	Vytvorenie projektu pre refaktoring starého kódu Vizualizácie priestoru
Task	Odstránenie nepotrebného kódu	Jakub Ginter	Zmazanie starého kódu, ktorý sa nepoužíva

Project: 3drecon Server: 3drecon.visualstudio.com\3drecon Query: 3drecon Team - Sprint 3 - Backlog List type: Tree						
ID	Work Item Type	Title 1	Title 2	Assigned To	Description	Remaining Work
146	Product Backlog Item	Dokumentácia k inžinierskemu dielu				
156	Task	Štruktúra dokumentu		Jakub Ginter	Vytvorenie nového dokumentu na google drive. Návrh štruktúry - kapitoly, podkapitoly.	
157	Task	Kapitola úvod		Jakub Ginter		
158	Task	Kapitola ciele		Jakub Ginter		
145	Product Backlog Item	Dokumentácia riadenia				
147	Task	Štruktúra dokumentu		Mario Hunka	Vytvorenie nového dokumentu na google drive. Návrh štruktúry - kapitoly, podkapitoly.	Documentation
148	Task	Finalizácia dokumentu		Mario Hunka	Po dokončení dokumentácie je potrebné naformátovať celý dokumentáciu do jedného štýlu.	1 Documentation
149	Task	Kapitola úvod		Mario Hunka	Napísanie kapitoly úvod.	
150	Task	Integrácia dokumentácie		Mario Hunka	Vzhľadom na to, že dokumentáciu robia všetci členovia tímu. Je potrebné dať všetko do jedného súhrného dokumentu.	Documentation
151	Task	Sumarizácia šprintu 1		Jakub Ginter		
152	Task	Export evidencie úloh		Jakub Ginter		
153	Task	Manažment komunikácie		Jakub Ginter		
154	Task	Manažment vývoja a integrácie		Jakub Ginter		
155	Task	Manažment riadenia		hascicm.mh		
159	Task	Globalna retrospektiva		Mario Hunka		
160	Task	Manazment planovania		Mario Hunka		
142	Bug	Zlá konverzia stringu pri načítavaní nového		Viktor Košťan		3 Testing

143	Task	Oprava konverzie stringu pri načítavani (TEST)	Viktor Košťan			Development
144	Task	Oprava konverzie stringu pri načítavani			1	Development
140	Product Backlog Item	Webová prezentácia tímu	Mario Hunka			
141	Task	Update priebežnej dokumentacie na webe	Mario Hunka			Documentation
137	Product Backlog Item	Tvorba dokumentu na logovanie odrobených hodín	Viktor Košťan	na taskoch, aby mal produkt owner prehľad o odrobenej práci, keďže tfs nedovoľuje zdarma prístup do projektu pre viac ako 5 ľudí.		
138	Task	Vytvorenie excel sheetu na logovanie práce	Viktor Košťan	Je potrebné vytvoriť excel sheet na logovanie práce a vložiť do zdieľaného priečinku na drive.		Documentation
75	Product Backlog Item	Refaktoring tried segmentácie	Richard Pinter			
78	Task	Odstárenie nepotrebného kódu	Richard Pinter		5	
55	Product Backlog Item	Vizualizácia spracovaných dát	Jakub Ginter	Všetky dáta, ktoré načítavame a spracovávame je potrebné vizualizovať.		
91	Task	Vizualizácia Point cloud vo formáte XYZ	Jakub Ginter	Vizualizácia Point cloud vo formáte XYZ pre naskenovaný priestor	5	
93	Task	Vizualizácia Point cloud s farebným odlíšením	Jakub Ginter	Vizualizácia Point cloud s farebným odlíšením pre naskenovaný priestor	5	
52	Product Backlog Item	Refaktoring ukladania (exportu) dát do	Viktor Košťan			
94	Task	Premiestnenie triedy Exporter do Refactoring triedy Exporter v	Viktor Košťan			
95	Task	Refactoring triedy Exporter v	Viktor Košťan		3	
50	Product Backlog Item	Refactoring triedy ClosedSpace	Viktor Košťan			
88	Task	Vytvorenie segmentačného	Viktor Košťan		4	
106	Product Backlog Item	Finalizácia refaktoringu		Posledná časť refaktoringu, v ktorej sa vykoná merge všetkých branches refaktoringu do jednej výslednej.		

107	Task	Vytvorenie metódy calculate pre triedu (MERGE)	Richard Pinter	5	Development
111	Task	Vytvorenie metódy calculate pre (TEST)		1	Development
110	Task	Vytvorenie metódy calculate pre (REVIEW)		2	Testing
112	Task	Vytvorenie metódy calculate pre (MERGE)		2	Development
108	Task	Vytvorenie metódy calculate pre (MERGE)	hascim.mh	8	Development
113	Task	Vytvorenie metódy calculate pre (TEST)		1	
114	Task	Vytvorenie metódy calculate pre (REVIEW)		2	
115	Task	Vytvorenie metódy calculate pre (MERGE)		2	Development
109	Task	Integrácia segmentačných funkcií	Viktor Košťan	3	Development
116	Task	(MERGE) Integrácia segmentačných funkcií		1	Development
117	Task	(TEST) Integrácia segmentačných funkcií		2	Testing
118	Task	(REVIEW) Integrácia segmentačných funkcií		1	Development
119	Task	Handler pre GUI	Mario Hunka	3	Development
120	Task	(MERGE) Handler pre GUI		1	Development
121	Task	(TEST) Handler pre GUI		2	Testing
122	Task	(REVIEW) Handler pre GUI		1	Development
123	Task	Vizualizácia segmentovaných dát	Jakub Ginter	8	Development
124	Task	Interaktor vizualizéru	Jakub Ginter	8	Development
125	Task	(MERGE) Vizualizér		1	Development

126	Task		(TEST) Vizualizér			5	Testing		
127	Task	(REVIEW) Vizualizér						2	Development
131	Product Backlog Item	Tvorba metodík							
132	Task	Manažment code review		Mario Hunka	Metodika na to ako robiť prehliadky kódu.			Documentation	
133	Task	Manažment verzii		Jakub Ginter	Metodika verziovania projektu			Documentation	
134	Task	Manažment testovania		hascicm.mh				Documentation	
135	Task	Manažment úloh		Viktor Košťan				Documentation	
136	Task	Manažment dokumentácie		Richard Pinter				8	Documentation
100	Product Backlog Item	Migrácia existujúcej implementácie segmentácie outlayerov							
128	Task	Migrácia existujúcej implementácie (TEST) Migrácia existujúcej		Mario Hunka	Presun logiky segmentácie outlayerov do nového projektu.			13	Development
129	Task							5	Testing
130	Task	(REVIEW) Migrácia existujúcej						2	Development

## **8 PREBERACICE PROTOKOLY**

Tímový projekt 2016/2017

Tím č. 11 – 3DRecon

**Predmet odovzdávania:**

Dokumentácia riadenia – prvých troch šprintoch

Projektová dokumentácia – po prvých troch šprintoch

Vedúci tímového projektu: Ing. Vanda Benešová, PhD.

Podpisom potvrdzuje prevzatie vyššie uvedených častí dokumentácie

V Bratislave

Dátum

Podpis



## 9 MOTIVAČNÝ DOKUMENT

Dobrý deň,

dovoľte mi predstaviť náš tím pre tímový projekt. Skladá sa zo 6 absolventov bakalárskeho štúdia na FIIT, pričom sme všetci študovali v odbore informatika. Menovite sa v tíme nachádza Jakub Ginter, Miroslav Haščič, Mário Hunka, Viktor Košťan, Richard Pintér a Matej Kollár. Nakoľko neštudujeme všetci rovnaký odbor sú v našom rozvrhu zastúpené takmer všetky odborné predmety inžinierskeho štúdia, ktoré nám poskytnú znalosti pre prácu na širokej škále projektov. Tím sme však neskladali náhodne. Pracovali sme spolu na viacerých menších projektoch v rámci štúdia, nie sme len kolegovia, ale aj kamaráti. Využívali sme rôzne technológie, primárne však prevládajú v našom tíme skúsenosti s vývojom v jazyku Java. S týmto jazykom majú široké skúsenosti všetci členovia nášho tímu. V rámci databázových technológií sú v našom tíme zastúpené PostgreSQL a MySQL ale tak isto aj MariaDB či Hibernate. Náš tím dokáže efektívne pracovať na vývoji ako webových aplikácií tak aj mobilných. Pokročilé skúsenosti máme s vývojom v Pythone(Django), C/C++ a už spomínaná Java. Pri tvorbe webovej aplikácie vieme využiť naše skúsenosti v HTML, CSS, JavaScript(AngularJS). V rámci školských povinností sme si vyskúšali prácu s použitím technológií v oblasti počítačových sietí, databáz, paralelného programovania alebo grafiky a videnia. V rámci bakalárskych prác sme pracovali na témach - vývoj aplikácie pre mobilné zariadenia s platformou Android, analýza správania používateľa, vizualizáciu dát v obohatenej realite a iné.

Počas mimoškolských aktivít sme nabrali skúsenosti v oblasti vývoja i testovania softvéru, tvorby webových stránok či v analýze dát.

Doposiaľ dosiahnuté vedomosti dokážeme efektívne využívať a rozdeliť si prácu, aby každý člen pracoval na tom čo ho zaujíma. Našou najsilnejšou stránkou je tímový duch a chuť pracovať na projekte spoločne a tým vytvoriť čo najzaujímavejší produkt s potenciálom pre reálne využitie praxi.