Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

Projektová dokumentácia

Tím 1: LEGENDRONE

Akademický rok	2016/17
Predmet	Tímový projekt
Študenti	Bc. Tomáš Čičman Bc. Dušan Drábik Bc. Dušan Hatvani Bc. Patrik Kadlečík Bc. Mário Keszeli Bc. Martin Mocko Bc. Lukáš Visokai
Vedúci tímu	Ing. Viktor Šulák

Obsah

DOKUMENTÁCIA K RIADENIU

Dokumentácia k riadeniu

1 Úvod

Dokument vznikol v rámci predmetu Tímový projekt a opisuje prácu na projekte *Simulácia správania bezpilotných lietadiel v roji*. Obsahom tejto dokumentácie je opis manažmentu v tíme. Špecifikácia vytváraného systému je bližšie opísaná v dokumentácii inžinierskeho diela.

V kapitole 2 sú predstavení jednotliví členovia tímu, sú v nej definované zodpovednosti každého člena tímu a tiež podiel práce na tomto dokumente. Kapitola 3 bližšie špecifikuje procesy manažmentu projektu. Kapitola 4 sumarizuje priebeh a výsledky všetkých piatich šprintov vykonaných v priebehu zimného semestra. Poskytuje tiež stručné zhodnotenie príslušného šprintu. Ďalšia piata kapitola je venovaná súhrnu metodík definovaných a využívaných v tíme. Dokumentáciu k riadeniu uzatvára kapitola 6 globálnou retrospektívou práce v zimnom semestri. V retrospektíve je stručne zhrnuté čo chceme začať robiť do budúcnosti, V čom chceme pokračovať a s čím chceme skončiť.

V prílohe A sa nachádzajú všetky metodiky, na ktoré odkazuje kapitola 5, v plnom znení. Príloha B obsahuje export evidencie úloh z nástroja TFS.

2 Členovia tímu a ich roly

2.1 Predstavenie členov tímu

- **Bc. Tomáš Čičman**
  Absolvent bakalárskeho štúdia na FIIT odbor Internetové technológie. Bakalársku prácu mal zameranú na návrh vizualizátora pozícií UAV vo FANET sieťach (C#). Počas práce si prehĺbil znalosti jazyka C#. Počas štúdia pracoval na projektoch ako program na kooperačné kreslenie viacerých používateľov (Java). Aktuálne pracuje ako tester.

  Skúsenosti: C#, C, SQL, Java

- **Bc. Dušan Drábik**
  Absolvent bakalárskeho štúdia na FIIT odbor Internetové technológie. Vo svojej bakalárskej práci sa venoval vývoju aplikácia pre platformu iOS a využitie technológie iBeacon. Počas štúdia pracoval na projektoch ako databázový systém (PHP, MySQL, Angular.js) či sieťovom komunikátore (Python). Momentálne pracuje ako softvérový tester.

  Skúsenosti: Swift, C#, PHP, JS, Angular.js, SQL, Python

- **Bc. Dušan Hatvani**
  Absolvent bakalárskeho štúdia na FIIT odbor Internetové technológie. V rámci bakalárskej práce sa venoval návrhu modulu pre softvérovo-definované siete. Počas štúdia pracoval na projektoch ako databázový systém (Java, MySQL), paketový analyzátor (Java), softvérový prepínač a smerovač (Java). Momentálne pracujem ako tím líder zodpovedný za kvalitu na projekte zameranom na poskytovanie sieťových služieb.

  Skúsenosti: C, Java, SQL, Python, ITIL, CCNA

- **Bc. Patrik Kadlečík**
  Absolvent bakalárskeho štúdia na FIIT odbor Internetové technológie. V rámci bakalárskej práce sa venoval návrhu automatizovaného systému pre meranie magnetických vlastností materiálov (LabView). Počas štúdia pracoval na projektoch ako databázový systém (Java, PostgreSQL), paketový analyzátor (Java), softvérový prepínač a smerovač (C#).

  Skúsenosti: LabView, C, C#, Java, SQL, CCNA

- **Bc. Mário Keszeli**
  Absolvent bakalárskeho štúdia na FIIT odbor Informatika. Vo svojej bakalárskej práci sa venoval spracovaniu obrazu pre snímanie identifikačných kariet mobilom. Počas štúdia pracoval na projektoch ako paketový analyzátor (C#, Java) či databázový systém (Java, PostgreSQL, Hibernate). Momentálne pracuje ako imaging developer.

  Skúsenosti: Java/Android, JNI C++, C++, OpenCV, C#, PostgreSQL, Hibernate

- **Bc. Martin Mocko**
  Absolvent bakalárskeho štúdia na FIIT odbor Informatika. Vo svojej bakalárskej práci sa venoval objavovaniu trendov v používaní webového portálu (Java, Python). Počas štúdia pracoval na projektoch umelej inteligencie (C, Java) či webovej stránke s využitím technológií Jekyll, GitHub Pages. Taktiež počas štúdia sa naučil robiť objektovo-orientované aplikácie v Jave a vyskúšal si, ako sa vyvíjajú enterprise aplikácie. V programovacom jazyku C si vyskúšal paralelné programovanie (OpenMP, MPI). V Pythone si zvykne občas písať menšie skripty na uľahčenie roboty. Implementoval aj jednoduchú 2.5D hru “Arkanoid” v jazyku C++.

  Skúsenosti: XML/JSON, DocBook, XSLT

- **Bc. Lukáš Visokai**
  Absolvent bakalárskeho štúdia na FIIT odbor Internetové technológie. V bakalárskej práci sa venoval téme automatickej zmeny pozícií UAV na základe kvality linky (C#). Počas štúdia pracoval na databázovom systéme (Java, PostgreSQL) či softvérovom smerovači a prepínači (C#). Aktuálne pracuje na štátnom projekte, ktorý sa zameriava na vytvorenie webovej aplikácie s prepojením do národného cloudu a RIS.

  Skúsenosti: C#, PostgreSQL, C, Java/Android

2.2 Rozdelenie manažérskych úloh

	Zodpovednosť za manažment v tímovom projekte	
Člen tímu	**Dlhodobá**	**Krátkodobá**
Bc. Tomáš Čičman (manažér testovania)	administrátor webového sídla a administrátor Team Foundation Server	zapisovateľ, tvorca retrospektív, dohľad na dodržiavanie termínov
Bc. Dušan Drábik (manažér architektúry)	správca architektúry projektu	analýza existujúceho riešenia, návrh nového riešenia
Bc. Dušan Hatvani (manažér zberu požiadaviek a technickej podpory)	správca vizualizátora, technická podpora v sieťovom odvetví	zapisovateľ, informovanie o aktuálnom stave projektu
Bc. Patrik Kadlečík (manažér úloh)	administrátor a správca Microsoft Team Foundation Server	Scrum master, tvorca web stránky, zapisovateľ, tvorca šablóny retrospektív, príprava rešerší
Bc. Mário Keszeli (manažér tvorby dokumentácie)	správca projektovej dokumentácie	analýza existujúceho riešenia, úprava web stránky, tvorba šablóny zápisníc
Bc. Martin Mocko (manažér vývoja)	programátor, správca repozitára	metodiky práce v tíme, buildy, udržiavanie aktuálnych informácií o stave projektu
Bc. Lukáš Visokai (manažér komunikácie)	programátor, dohliadanie na komunikáciu v tíme, dodržiavanie úloh a zodpovedností	Scrum master, zapisovateľ, príprava rešerší, podporná práca, komunikácia s predchádzajúcim tímom

2.3 Podiel práce na jednotlivých častiach dokumentácie

- **Bc. Tomáš Čičman**
  Dokumentácia k riadeniu:
  - Manažment testovania
  - Metodika testovania
  - Implementácia
  - Testovanie

- **Bc. Dušan Drábik**
  Dokumentácia inžinierskeho diela
  - Celkový pohľad na systém, diagram nasadenia, diagram tried a sekvenčné diagramy

- **Bc. Dušan Hatvani**
  Dokumentácia k riadeniu:
  - Sumarizácia šprintov
  - Manažment zberu požiadaviek

  Dokumentácia inžinierskeho diela
  - Návrh komunikácie medzi prostredím a vizualizátorom
  - Návrh vizualizátora
  - Implementácia

- **Bc. Patrik Kadlečík**
  Dokumentácia k riadeniu:
  - Globálna retrospektíva
  - sumarizácia šprintov

  Dokumentácia inžinierskeho diela
  - Export evidencie úloh z TFS
  - Metodika manažmentu úloh

- **Bc. Mário Keszeli**
  Dokumentácia k riadeniu:
  - Úvod (dokumentácia k riadeniu), predstavenie členov tímu, sumarizácia šprintov

  Dokumentácia inžinierskeho diela
  - Analýza - analýza existujúceho riešenia, analýza použitia bezpilotných lietadiel
  - Metodika dokumentácie

- **Bc. Martin Mocko**
  Dokumentácia k riadeniu:
  - Aplikácie manažmentov - komunikácie, dokumentácie, plánovania, kvality
  - Metodika prehliadok kódu a verziovania, komunikácie, dokumentácie, písania kódu

- **Bc. Lukáš Visokai**
  Dokumentácia k riadeniu:
  - Rozdelenie manažérskych úloh
  - sumarizácia šprintov

  Dokumentácia inžinierskeho diela
  - Úvod (dokumentácia inžinierskeho diela), ciele projektu pre zimný semester, analýza existujúceho riešenia, analýza použitia bezpilotných lietadiel
  - Globálna retrospektíva
  - Metodika písania kódu, revidovanie metodiky

3 Aplikácie manažmentov

Manažment dokumentácie

V rámci manažmentu dokumentácie vytvárame priebežné a jednotlivé dokumenty. Do priebežných dokumentov zaraďujeme zápisnicu z každého tímového stretnutia, retrospektívu po každom šprinte a dokumentáciu zdrojového kódu. Tieto dokumenty sa buď priebežne menia alebo sa priebežne vytvárajú nové verzie podľa potreby. Do jednotlivých dokumentov zaraďujeme tímové metodiky, dokumentáciu riadenia a dokumentáciu inžinierskeho diela. Každý z vytváraných dokumentov v rámci tímového projektu má zadefinovanú šablónu, podľa ktorej sa tento dokument vytvára.

Manažment komunikácie

V rámci tímovej komunikácie môžeme komunikáciu rozdeliť na komunikáciu na stretnutí a komunikáciu mimo stretnutia. V rámci komunikácie na stretnutí má každý člen tímu priestor povedať o svojich výsledkoch a problémoch, ktoré nastali, a ak je to nutné, otvoríme priestor pre hlbšiu diskusiu o vzniknutých problémoch. Komunikácia mimo stretnutia v rámci tímu funguje hlavne v nástroji Slack. Pre rôzne účely sme vytvorili viacero kanálov, podľa druhu témy, ktorej sa daná komunikácia týka.

Ďalej je dôležité mať zavedené efektívne fungovanie komunikácie s produktovým vlastníkom, ktorým je zároveň aj náš vedúci tímu. Na túto komunikáciu používame taktiež nástroj Slack, pričom máme neformálne dohodnutie s vedúcim, že pokiaľ potrebujeme, aby reagoval na naše správy, označíme ho v nástroji spôsobom “@viktor_sulak”. Vedúceho taktiež informujeme o aktuálnom stave projektu a úloh na pravidelných týždenných tímových stretnutiach.

Manažment zberu požiadaviek

V manažmentu zberu požiadaviek priebežne vytvárame požiadavky pre náš projekt spolu s produktovým vlastníkom. Zber požiadaviek prebieha väčšinou počas priamej komunikácie s produktovým vlastníkom. Iná forma zberu požiadaviek je nepriamo pomocou diagramov prípadov použitia (angl. Use Case), ktoré počas tímových stretnutí navrhujeme a pomocou nich identifikujeme požiadavky. Pokiaľ sú navrhnuté nejaké požiadavky, musia byť dôkladne analyzované. Až po analyzovaní sa môže rozhodnúť, či budú požiadavky akceptované v tej forme, aká bola navrhnutá. Ak nastane prípad, že požiadavky z istého dôvodu nemôžu byť naplnené, je nutné ich pozmeniť, respektíve od nich úplne upustiť.

Manažment plánovania

Plánovanie v rámci tímu. Na začiatku semestra sme vytvorili základné používateľské príbehy produktu aj s popisom, teda minimálne požiadavky, ktoré by mali byť splnené. Na stretnutí pred začiatkom šprintu vždy podľa priority používateľských príbehov zoberieme ich časť, pre ktorú sa zaviažeme, že ju stihneme spraviť. Tieto používateľské príbehy následne rozdelíme na menšie úlohy, ktoré obsahujú všetko to, čo je potrebné splniť pre úspešné vykonanie používateľského príbehu. Jednotlivé úlohy sa pridelia členom tímu tak, aby každý člen tímu mal určenú robotu na šprint a taktiež tak, aby sme vedeli, že má predstavu o tom, čo bude robiť. Po začatí šprintu je o týždeň stretnutie v rámci šprintu, kde rozoberáme všetko, čo sa podarilo za uplynulý týždeň členom tímu vytvoriť/vyriešiť a taktiež ťažkosti, s ktorými sa stretli alebo museli vysporiadať a navrhujeme možné riešenia. Ak je to potrebné, vytvoríme ďalšie úlohy, ktoré pri prvotnom plánovaní nebolo zrejmé, že treba vyriešiť.

Stav projektu. Stav projektu je reflektovaný v stavoch úloh, ktoré sú v produktovom backlogu. Tieto stavy úloh sú pravidelne aktualizované. Stav aktuálneho šprintu je reflektovaný v burndown grafe. Všetky úlohy, používateľské príbehy, epické príbehy, ako aj záznamy o chybách môžeme nájsť v nástroji TFS.

Stav úloh. Ku každej úlohe je uvedený jej názov, popis (description), priorita úlohy a odhad (v počte story pointov), koľko bude táto úloha trvať. Pred tým, ako sa začne robiť na úlohe, musí mať úloha jasne určené, kto na nej bude pracovať.

Zhodnotenie šprintu a prírastku do systému. Po každom šprinte na stretnutí tím prezentuje produktovému vlastníkovi nový prírastok k produktu a aj všetky splnené a nesplnené úlohy.

Manažment testovania

V rámci testovania môžeme testovanie rozdeliť na dve časti, konkrétne na testovanie kódu a testovanie vizuálneho výstupu. Testovanie kódu je zabezpečované samotným vývojárom, ktorý daný kód implementoval. Toto testovanie bude prebiehať pomocou unit testov implementovaných v kóde. Testovanie vizuálneho výstupu v sebe zahŕňa napríklad testovanie plynulosti zobrazovania, prekrývania objektov a iných vizuálnych častí v simulácii.

Manažment úloh

V rámci manažmentu úloh je vytvorená metodika manažmentu úloh, ktorá opisuje postupy na správu úloh v nástroji TFS. Je potrebné mať určené ako vytvárať, priraďovať a dokončovať úlohy. Úlohy sa vytvárajú na základe konzultácii pred začiatkom šprintu. Metódou “planning poker” určujeme náročnosť úloh. Úlohy sú priraďované členom tímu počas plánovania šprintu a vykonávajú sa podľa priority. Nižšie číslo určuje väčšiu prioritu. Manažér úloh je zodpovedný za správu všetkých úloh v projekte v nástroji TFS.

Manažment kvality

Na zabezpečenie dostatočnej kvality jednotlivých úloh máme zadefinované rôzne metodiky, ako napr. metodiku prehliadky kódu a verziovania, metodiku písania kódu a taktiež tímový "definition of done".

"Definition of done" pre používateľský príbeh (user story):
Product Backlog Item (PBI) spĺňa tímovú "definition of done" vtedy, keď:

- Prejde unit testami, ktoré boli pre dané úlohy vytvorené
- Kód bol posúdený (angl. "reviewed") aspoň jedným ďalším členom tímu, ktorý sa nepodieľal na tvorbe kódu
- PBI musí spĺňať požiadavky používateľského príbehu (akceptačné kritériá od produktového vlastníka)

"Definition of done" pre úlohy:

- Kód implementovaný pre danú úlohu musel prejsť cez posúdenie kódu (angl. "code review")
- Ak sú pre túto úlohu vytvorené unit testy, kód musí prejsť aj nimi

4 Sumarizácie šprintov

4.1 Šprint 1

Pri prvom tímovom stretnutí sme sa zoznámili s tímom, ktorý počas minulého akademického roka vypracovával projekt, na ktorý tento tímový projekt nadväzuje. Oboznámili sme sa s existujúcim riešením, diskutovali sme s bývalým tímom o ďalšom smerovaní projektu.

Oboznámili sme sa tiež s vývojovým prostredím a nástrojmi využívanými pri práci na projekte, pri manažmente a tiež pri komunikácii v tíme. Vytvorili sme a nasadili webovú stránku tímového projektu. Začalo sa pracovať na vytváraní jednotlivých metodík.

Všetky úlohy sa nám nepodarilo dokončiť včas, takže sme boli nútení niektoré z nich presunúť do nasledujúceho šprintu.

Počas retrospektívy sme zhodnotili celý šprint jednotlivo, ako aj tímovo. Každý člen tímu zhodnotil zo svojho pohľadu to, čo išlo podľa jeho predstáv a naopak veci, ktoré nešli podľa jeho predstáv. Nakoniec každý navrhol, čo by bolo vhodné v budúcich šprintoch vylepšiť alebo prestať robiť.

Spolu s nedokončenými úlohami sme vytvorili úlohy na ďalší šprint a pridelili ich jednotlivým členom tímu.

4.2 Šprint 2

Počas druhého šprintu sme sa zamerali na detailnú analýzu existujúceho riešenia bývalého tímu. Okrem analýzy existujúceho riešenia sme sa zamerali aj na hlbšiu analýzu použitia bezpilotných lietadiel a ich senzorov.

Úspešne sme dokončili všetky zostávajúce metodiky. V druhom šprinte sa nám poradilo splniť všetky zadané úlohy a žiadne neboli prenášané do nasledujúceho šprintu.

Po dôkladnej analýze existujúceho riešenia a diskusii sme sa kvôli existujúcim problémom a nedostatkom jednomyseľne zhodli, že nebudeme pokračovať v rozširovaní projektu bývalého tímu, ale navrhneme a implementujeme od základov nové riešenie.

Počas retrospektívy sme opäť prediskutovali klady a zápory uplynulého šprintu. Vecí, ktoré chceme prestať robiť, bolo v tomto šprinte podstatne menej ako v predchádzajúcom.

4.3 Šprint 3

Po dôkladnej analýze riešenia minulého projektu sme sa rozhodli začať robiť projekt úplne od začiatku. Konzultovali sme s produktovým vlastníkom požiadavky na systém, aby sme mohli vytvoriť návrh nového systému.

Identifikované úlohy boli zamerané väčšinou na návrh systému. Vytvorili sme potrebné diagramy (diagram tried, sekvenčné diagramy, diagram komponentov). Navrhli sme formát správ posielaných pri komunikácii medzi UAV a komunikácii medzi prostredím a vizualizátorom. Navrhli sme tiež formát záznamov (tzv. log súborov), ktoré slúžia na kontrolu správania UAV v simulácii. Navrhli sme prostredie simulácie, reprezentáciu mapy prostredia a pôsobenie poveternostných podmienok. Vytvorili sme metodiku testovania a dokončili sme dokumentáciu k prvému kontrolnému bodu. V rámci tímovej réžie sme spustili kontinuálnu integráciu v TFS. Všetky úlohy priradené k tomuto šprintu sa nám podarilo dokončiť.

V retrospektíve sme prediskutovali šprint. Identifikovali sme klady a zápory a navrhli sme zlepšenia do ďalšieho šprintu. Taktiež sme identifikovali veci, ktorým sa chceme v budúcnosti vyvarovať.

4.4 Šprint 4

Štvrtý šprint bol venovaný implementácii základnej verzie finálneho produktu (angl. minimum viable product – MVP). Na začiatku bolo potrebné vytvoriť základnú kostru celej aplikácie a dokončiť rozpracované sekvenčné diagramy. Komponenty pochádzajúce z predchádzajúceho projektu, u ktorých sme uznali, že sú funkčné a nie je potrebné ich nanovo vyvíjať, sme znovupoužili v našej novej implementácii. Počas práce na MVP sme postupne implementovali základy modulov prostredia, simulácie, komunikácie a modely UAV. Taktiež bola v tejto fáze implementácie vyvinutá inicializácia jednotlivých modulov na základe konfiguračných JSON súborov.

Počas retrospektívy sme prediskutovali plnenie jednotlivých úloh v tomto šprinte. Všetky zadefinované úlohy sa nám podarilo splniť.

4.5 Šprint 5

V závere zimného semestra sme pracovali na dokončovaní MVP. Identifikovali sme súčasti, od ktorých závisí fungovanie aplikácie ako celku vrátane grafického používateľského rozhrania.

Rozšírili sme simuláciu o aplikovanie poveternostných vplyvov na pohyb UAV a pridali sme prepojenie s grafickým používateľským rozhraním vizualizátora. Taktiež sme dokončili spracovanie konfiguračných súborov, pomocou ktorých je inicializované prostredie, mapa a objekty UAV. Implementovali sme základnú kostru TCP spojenia na strane vizualizátora. Pripravili sme a sfunkčnili posielanie správ medzi UAV. V rámci šprintu sme tiež dokončovali finálnu dokumentáciu pre druhý kontrolný bod v zimnom semestri.

Po poslednej retrospektíve v zimnom semestri, ktorá bola venovaná piatemu šprintu, sme si spravili globálnu retrospektívu k celému zimnému semestru a tiež k MVP, aby sme si zhrnuli, čo sa nám podarilo implementovať a čo je v budúcnosti potrebné dokončiť.

4.6 Šprint 6

V prvom šprinte letného semestra sme si na prvom stretnutí overili a zistili ďalšie požiadavky na produkt od Produktového vlastníka. Od získaných požiadaviek sa odvíjali ďalšie šprinty.

Počas stretnutia sme odprezentovali progres na produkte produktovému vlastníkovi. Ďalej sme naplánovali úlohy šprintu.

Celý šprint sa zaoberal implementáciou základnej simulácie. Na stretnutí sme navrhli viacero možností a z nich sme zostavili úlohy šprintu. Úlohy šprintu riešili inicializáciu spojenia medzi simulátorom a vizualizátorm. Vytvorili sme základnú logiku pre logovanie správ posielaných medzi objektami UAV a implementovali sme deserializáciu objektu UAV na formát JSON správ. Implementovali sme TCP klienta na strane vizualizátora a vyriešili sme chybu duplikovaného spúšťania UNITY pri viacnásobnom spustení simulátora. Posledná úloha riešila inicializačnú obrazovku systému, kde sme implementovali možnosť spúšťania simulátora s vizualizátorom a bez neho.

Počas retrospektívy sme prediskutovali plnenie jednotlivých úloh v tomto šprinte. Všetky zadefinované úlohy sa nám podarilo splniť.

4.7 Šprint 7

Druhý šprint letného semestra zastrešoval vizualizačnú časť produktu. V šprinte sme sa zaoberali hlavne JSON správami posielanými medzi simulátorom a vizualizátorom. Na stretnutí počas šprintu sme diskutovali a navrhovali prácu s vláknami, pričom požiadavka od produktového vlastníka bola implementovať každý objekt UAV v samostatnom vlákne. Pustili sme sa do jednoduchého návrhu kolízií objektov v simulácii.

Úlohy v šprinte riešili logiku preposielania JSON správ a parsovanie správ na strane vizualizátora UNITY. Implementovali sme potrebnú funkcionalitu na preposielanie správ a ich parsovanie. Implementovali sme funkcionalitu na prácu s vláknami. Vyriešili a opravili sme chybu spôsobenú ukončením programu spolu s UNITY. Implementovali sme základné funkcie, ktoré riešili kolízie objektov UAV v simulácii.

Počas retrospektívy sme prediskutovali plnenie jednotlivých úloh v tomto šprinte. Všetky zadefinované úlohy sa nám podarilo splniť.

4.8 Šprint 8

V tretom šprinte letného semestra sme sa zaoberali riešením kolízií objektov a správaním objektov po kolízii. Taktiež sme začali s návrhom smerovacieho protokolu. Počas implementácie funkcionality poskytujúcej spúšťanie simulátora a UNITY cez LAN sieť vznikla chyba, ktorú bolo potrebné odstrániť. Úlohy šprintu riešili návrh a základnú implementáciu smerovacieho protokolu, kolízie objektov a posielanie správ objektmi UAV iba do definovanej

vzdialenosti podľa použitej komunikačnej technológie. Implementovali sme základnú funkcionalitu smerovacieho protokolu. Odstránili sme chybu vznikajúcu pri spúšťaní simulátora a vizualizátora cez LAN sieť. Implementovali sme pokročilú logiku kolízií objektov. Pridali sme vplyv vetra na pohyb objektov UAV a implementovali sme posielanie správ objektmi UAV iba do definovanej vzdialenosti podľa použitého komunikátora.

Počas retrospektívy sme prediskutovali plnenie jednotlivých úloh v tomto šprinte. Všetky zadefinované úlohy sa nám podarilo splniť.

4.9 Šprint 9

V štvrtom šprinte sme sa zaoberali komunikáciou medzi objektmi UAV. Podľa požiadaviek sme pridali do systému možnosť pridávania vlastnej IP adresy pri spúšťaní systému cez LAN a doplnili sme logiku logovania správ posielaných medzi objektmi UAV. Do systému sme teda implementovali nami navrhnutý smerovací protokol. Implementovala sa všetka logika pre logovanie správ posielaných UAV objektmi. Pridala sa funkcionalita poskytujúca používateľovi zadať vlastnú IP adresu pri použití simulátora cez LAN. Navrhli a implementovali sme logiku pohybu UAV po kolízii s iným UAV alebo s objektom prostredia.

Počas retrospektívy sme prediskutovali plnenie jednotlivých úloh v tomto šprinte. Všetky zadefinované úlohy sa nám podarilo splniť.

4.10 Šprint 10

V poslednom desiatom šprinte letného semestra sme dokončovali a odovzdali produkt tretej strane na testovanie. Poznámky a postrehy z testovania sme zapracovali do finálneho produktu. Dokončili sme finálnu verziu dokumentácie, do ktorej sme pridali používateľskú a inštalačnú príručku. Vygenerovali sme dokumentáciu zo zdrojového kódu a pridali ju do finálnej dokumentácie. Dokumentáciu sme spolu s produktom odovzdali vedúcemu a aktualizovali sme tímovú web stránku.

Počas retrospektívy sme diskutovali o celom priebehu letného semestra a o splnení cieľov projektu.

5 Používané metodiky

Pre správne fungovanie tímu a jasne zadefinované základné postupy sme sa rozhodli vytvoriť pre náš tím metodiky tímovej práce, ktoré počas práce na projekte dodržiavame. V tejto kapitole sa nachádza prehľad použitých metodík, ako aj stručný opis toho, o čom daná metodika pojednáva. Plné znenie príslušnej metodiky sa nachádza v prílohe tohto dokumentu.

5.1 Metodika dokumentácie

Táto metodika definuje rozsah dokumentácie vytváranej v rámci tímového projektu a tiež štruktúru a štýly pre jednotlivé dokumenty. Okrem dokumentácie riadenia a dokumentácie inžinierskeho diela sa v rámci tímového projektu pravidelne na každom stretnutí vytvárajú zápisnice, po ukončení šprintov zas zápisy retrospektív. Prostredníctvom nástrojov a postupov definovaných v metodike sa generuje dokumentácia zdrojového kódu. Pre jednotlivé druhy dokumentov boli vytvorené šablóny uložené v tímovom repozitári.

5.2 Metodika písania kódu

V rámci tejto metodiky sme si zadefinovali, akým spôsobom budeme písať názvy premenných v príslušnom programovacom jazyku. Metodika je zadefinovaná pre dva najdôležitejšie jazyky, ktoré sa budú v rámci tímového projektu využívať, a tými sú C# a Python. V týchto jazykoch máme taktiež zadefinovanú metodiku komentovania kódu.

5.3 Metodika komunikácie

Metodika komunikácie má za cieľ definovanie procesov komunikácie, ktoré v rámci tímového projektu uplatňujeme. Medzi hlavné nástroje na komunikáciu v tíme patrí náš systém na manažment projektu (TFS) a nástroj na komunikáciu v tíme (Slack) s rôznymi komunikačnými kanálmi. Pre neformálnu komunikáciu použijeme tímovú skupinu na sociálnej sieti Facebook. Je tu popísané, v akých prípadoch (podľa typu problému, úlohy, alebo závažnosti) môžeme dané nástroje využiť.

5.4 Metodika prehliadok kódu a verziovania

Po prvotných neúspechoch so systémom verziovania typu TFVC sme sa rozhodli prejsť na systém verziovania Git. V tejto metodike je popísanie, ako postupovať od pridelenia novej úlohy, cez commitovanie, požiadanie o pull request, až po spojenie novej vetvy pre danú úlohu s vyššou (master) vetvou. Je tu dôkladne opísaný postup, ktorým musia autori kódu prejsť, aby vytvorili nový pull request (čím požiadajú o prehliadku kódu - tzv. “code review”). Následne v metodike písania kódu sú definované dve základné pravidlá, ktoré musí kód spĺňať, aby mohol byť akceptovaný.

5.5 Metodika testovania

V rámci tejto metodiky sme jasne zadefinovali testovanie projektu. Uviedli sme kto vykonáva jednotlivé testy, kedy ich vykonáva a taktiež spôsob, akým majú byť vykonané. Pri testovaní kódu sme zadefinovali uplatnenie využitia unit testov a spôsob ich implementácie. Taktiež sme definovali spôsob testovania vizuálnej simulácie.

5.6 Metodika úloh

Účelom tejto metodiky je opis a definovanie postupov práce s úlohami v tíme. Ide o postupy pri tvorbe, zmene, odstraňovaní a úprave úloh v nástroji TFS (Team Foundation Server). Metodika je určená primárne pre manažéra úloh, ale slúži všetkým členom tímu. Manažovanie úloh prebieha v nástroji TFS a manažér úloh je zodpovedný za všetky úlohy, ktoré sa vyskytnú v projekte.

6 Globálna retrospektíva

Pri vývoji projektu sme aplikovali agilnú metódu SCRUM, ktorú sme sa snažili poctivo dodržiavať počas celého vývoja. Počas každého semestra sme absolvovali 5 šprintov, z toho 4 šprinty trvali 2 týždne a posledný šprint trval jeden týždeň. Na začiatku každého šprintu sme si spolu s produktovým vlastníkom zadefinovali cieľ šprintu. Po skončení každého šprintu prebiehala retrospektíva.

Prvé dva šprinty sme sa venovali analyzovaniu starého projektu, pričom sme sa na konci druhého šprintu počas retrospektívy zhodli, že budeme projekt vyvíjať nanovo. Pri plánovaní tretieho šprintu sme definovali nové požiadavky na MVP pre zimný semester.

Podarilo sa nám implementovať nasledujúce body z MVP:

- navrhnutie modulárneho systému pre jednoduché pridávanie nových funkcionalít
- inicializácia prostredia
- základný pohyb UAV spolu s aplikovaním vplyvov poveternostných podmienok
- definovanie komunikácie medzi UAV, odosielanie a prijímanie správ
- vizualizácia správania sa UAV prostredníctvom grafického používateľského rozhrania, resp. záznam prostredníctvom log súborov

Následne sme MVP obohatili o:

- logiku kolizii a vplyvov prostredia,
- integraciu simulatora a vizualizatora vratane ich komunikacie prostrednictvom TCP protokolu,
- komunikaciu medzi jednotlivými UAV,
- posielanie a smerovanie správ,
- logovanie pozície UAV v simulácii a zaznamenávanie komunikácie

6.1 Retrospektíva po šprintoch

Z práce v priebehu zimného semestra a po vyhodnotení procesov vyplynulo niekoľko postehov na zlepšenie.

Čo chceme začať robiť do budúcnosti

- Dohodnúť niekoľko odborných konzultácií s externým odborníkom
- Dôkladnejšie písanie Unit testov
- Analyzovať vytvorené testy (či pokrývajú čo majú)
- Zlepšiť dodržiavanie testovania
- Dôkladné dodržiavanie všetkých metodík
- Rozšíriť metodiku komunikácie o nástroj Slack
- Zlepšiť metodiku manažmentu úloh
- Vytvoriť metodiku zberu požiadaviek
- Včas plánovať šprinty
- Analyzovať dokončené úlohy

- Delegovať úlohy tak, aby viacero ľudí nemuselo pre svoje úlohy upravovať rovnaké triedy
- Dôkladnejšie opisovať úlohy
- Dôkladne dokumentovať zdrojový kód
- Dokumentovať diagramy prípadov použitia

V čom chceme pokračovať

- Písanie akceptačných testov
- Dobrá komunikácia tímu a plánovanie úloh celého tímu
- Teambuilding
- Prezentáciách progresu na úlohách
- Priebežne plnenie úloh
- Rozširovanie a skvalitňovanie metodík

S čím chceme skončiť

- Komunikačný šum pri priamych stretnutiach
- Komunikácia v súvislosti s konkrétnymi podrobnosťami návrhu a implementácie na sociálnej sieti Facebook

Po skončení letného semestra hodnotíme výrazný posun k lepšiemu. Pozitívne hodnotíme hlavne zlepšenie manažmentu nástroja na správu úloh TFS, čo výraznou mierou prispelo k efektívnejšej práci na projekte. Podarilo sa nám tiež zaznamenať pokrok v dokumentovaní zdrojového kódu. Celý zdrojový kód bol veľmi podrobne zdokumentovaný, vďaka čomu sa takisto zefektívnila práca a zlepšilo sa porozumenie pri prehliadke kódu.

Priestor na zlepšenie vidíme v striktnejšom dodržiavaní vývoja riadeného testami, čo bol jeden z kľúčových nedostatkov, ktoré sa nám v letnom semestri nepodarilo odstrániť.

Prílohy

A Metodiky

STU FIIT

LEGENDRONE

Metodika dokumentácie

Tím:	#1, LEGENDRONE
Vedúci tímu:	Ing. Viktor Šulák
Členovia tímu:	Tomáš Čičman, Dušan Drábik, Dušan Hatváni, Patrik Kadlečík, Mário Keszeli, Martin Mocko, Lukáš Visokai
Akademický rok:	2016/2017
Autor:	Mário Keszeli
Verzia číslo:	1.1
Dátum poslednej zmeny:	4.11.2016

STU FIIT

LEGENDRONE

Obsah

STU FIIT

LEGENDRONE

1. Úvod

Dokumentovať kód je dôležité hlavne kvôli tomu, aby si ľudia, ktorí majú záujem používať naše riešenie, prípadne v ňom pokračovať, mali odkiaľ naštudovať, čo a ako funguje. Je dôležité mať miesto, kam sa dá pozrieť a vedieť si nájsť ako sa volá funkcia, ktorá má nejaký význam/účel, a taktiež si pozrieť aké má vstupy a aké poskytuje výstupy. Takýto technický popis modelu nášho projektu poskytuje technická dokumentácia, ktorej pravidlá je dobré pre správne fungovanie tímového projektu zadefinovať. Taktiež je dôležité zadefinovať, akým spôsobom vytvárať iné druhy dokumentácií, a to tie, ktoré nebudú generované z nami vytváraného kódu, ale tie, čo vytvárame samostatne. Na to, aby bola forma vždy jednotná, je potrebné určiť si základné pravidlá písania dokumentácie.

2.Metodika generovania dokumentácie

Základné dokumentačné tagy:

- <summary> a <remarks>
- <param>
- <returns>
- <see> a <seealso>
- <c> a <code>
- <example>

Podrobný opis dokumentačných tagov s príkladmi ich použitia:

<summary> a <remarks>

Opis metód a tried.

```
/// <summary>
/// Sem patrí primárny opis metódy, resp. triedy. Opis je povinný pre každú
metódu a triedu.
/// </summary>
/// <remarks>
/// Voliteľné doplňujúce informácie k opisu metódy, resp. triedy.
/// </remarks>
```

<param>

Opis vstupných parametrov funkcie.

```
/// <summary>Opis metódy (pozri vyššie).</summary>
///
/// <param name="value">Tu bude opis parametra value metódy MyMethod.</param>
public void MyMethod(int value) { … }
```

<returns>

Opis výstupných hodnôt funkcie.

```
/// <summary>Opis metódy.</summary>
///
/// <returns>Tu sa opíše návratová hodnota metódy MyMethod.</returns>
public bool MyMethod() { … }
```

<see> a <seealso>

Umožňuje pridávať do dokumentácie odkazy (hyperlink). Pre externé odkazy sa využíva parameter href="…", pre interné odkazy v projekte sa používa cref="…".

```
/// <summary>Ak sa chceš v opise odkázať napríklad na nejakú metódu, typ,
udalosť a pod., použi see <see cref="M:Example.MyClass.MyMethod"/>.</summary>
/// <seealso href="http://www.w3.org/2001/XMLSchema.xsd">W3C XML Schema
2001</seealso>
```

<c> a <code>

Umožňuje v opisoch nastaviť formátovanie textu ako kód – vhodné pre opis príkladu použitia funkcie (pozri tag example nižšie).

<example>

Príklad použitia metódy, resp. triedy.

```
/// <summary>Opis metódy.</summary>
///
/// <example>
/// Objekt triedy <see cref="T:Example.MyClass"/> sa inicializuje nasledovne:
/// <code>
/// MyClass obj = new MyClass();
/// </code>
/// </example>
public class MyClass() { … }
```

3.Metodika tvorby dokumentácie

Nasledujúca metodika bude ponímať tvorbu dokumentácie, ktorá nie je generovaná, ale priamo tvorená nejakým členom (členmi) tímu. Dokumentovanie zahŕňa všetky typy dokumentov zahrnutých v dokumentácií k inžinierskemu dielu a v dokumentácií k riadeniu.

3.1. Pravidlá formy dokumentácie

Hlavné kapitoly majú nadpis typu 1, podkapitoly majú nadpis typu 2. Ďalší štýl, resp. úroveň nadpisov je na uvážení tvorcu dokumentácie. Typ písma textu normálneho paragrafu je Arial, veľkosť je 11. Riadkovanie použijeme veľkosti 1.05.

Použité číslovanie je v tvare:

1. Prvý nadpis
   1.1. Prvý podnadpis
2. Druhý nadpis
   2.1. Prvý podnadpis
   2.2. Druhý podnadpis

Odrážky majú nasledovný tvar:

- Prvá odrážka
  - Prvá pododrážka
- Druhá odrážka
  - Prvá pododrážka

Každá dokumentácia musí spĺňať podmienku správnej hlavičky dokumentu, ktorá obsahuje názov univerzity, fakulty, názov témy (dokumentácie), názov tímu, vedúceho tímu a členov tímu.

Použité skratky je nutné vysvetliť v kapitole „Použité skratky“. Použité cudzie alebo technické výrazy je nutné vysvetliť v kapitole „Slovník výrazov“.

Pre citácie a bibliografické údaje sa používa norma ISO 690 a ISO 690-2.

Výsledná finálna forma dokumentácie musí mať formát pdf. Pred odovzdaním a tlačou je nutné skontrolovať výzor a štruktúru dokumentácie.

STU FIIT

LEGENDRONE

Metodika písania kódu

Tím:	#1, LEGENDRONE
Vedúci tímu:	Ing. Viktor Šulák
Členovia tímu:	Tomáš Čičman, Dušan Drábik, Dušan Hatváni, Patrik Kadlečík, Mário Keszeli, Martin Mocko, Lukáš Visokai
Akademický rok:	2016/2017
Autor:	Lukáš Visokai
Verzia číslo:	1
Dátum poslednej zmeny:	3.11.2016

STU FIIT

LEGENDRONE

Obsah

STU FIIT

LEGENDRONE

1. Úvod

Cieľom tejto metodiky je zjednotiť menné konvencie a formátovanie kódu napísaného v rôznych programovacích jazykoch (C#, Python) do jednotnej štandardizovanej podoby. Tak, aby bol vytvorený kód jednoduchšie čitateľný a možno aj pochopiteľný pre vývojárov. Táto metodika je určená pre všetkých členov tímu, vzhľadom na to, že všetci sa určitým spôsobom podieľajú na vývoji.

1.1. Formátovanie kódu

Základným nástrojom na prácu v projekte pre nás bude Visual Studio 2015. Na formátovanie kódu sme sa rozhodli využiť plug-in do Visual Studia 2015, Resharper. V tomto plugine sme si nastavili profil podľa potreby tak, ako je možné vidieť neskôr pri kapitole 2.Metodika pomenovaní a taktiež 3.Metodika komentárov.

1.2. Štandard kódu

Kód musí spĺňať tímové štandardy – a teda musí byť:

- Ľahko testovateľný
- Málo závislý od iných tried/modulov

2.Metodika pomenovaní (naming conventions)

Všeobecne platí, že názvy sa píšu anglicky, tak, aby čo najlepšie vystihovali objekt, akciu, rozhranie..., ktoré zastupujú, sú zakázané názvy, ktoré nemajú zmysluplný význam. (jednopísmenové názvy premenných a pod.).

2.1. C#

Entity	Spôsob
Triedy, menný priestor	UpperCamelCase
Rozhrania	IUpperCamelCase
Typy parametrov	TUpperCamelCase
Metódy, vlastnosti a udalosti	UpperCamelCase
Lokálne premenné	lowerCamelCase
Lokálne konštanty	lowerCamelCase
Parametre	lowerCamelCase
Polia (nie privátne)	UpperCamelCase
Inštancie polí (privátne)	_lowerCamelCase
Statické polia (privátne)	_lowerCamelCase
Konštantné polia (nie privátne)	UpperCamelCase
Konštantné polia (privátne)	UpperCamelCase
Statické readonly polia (privátne, nie privátne)	UpperCamelCase
Enum	UpperCamelCase
Všetky ostatné entity	UpperCamelCase

2.2. Python

Entity	Spôsob
Triedy	UpperCamelCase
Všetko ostatné	lowerCamelCase

3.Metodika komentárov

Jazyk C# tak ako viaceré jazyky poskytuje 3 typy komentárov, blokový komentár, viac riadkový komentár so štandardnou syntaxou /**/ a komentár na dokumentovanie kódu, ktorý sa automaticky generuje po zadaní /// v IDE VS.

V programovacom jazyku Python sa zas v prevažnej väčšine používajú jednoriadkové komentáre, ktoré začínajú symbolom # (mriežka). Takýmto spôsobom teda treba dávať na každý riadok jednu mriežku, ktorá signalizuje komentár.

Nasledujúca tabuľka prehľadne ukazuje spôsob komentovania:

C#

Komentár	Notácia
Jednoriadkové	//
Viacriadkové	/* ..... */
Nad premenné a metódy	///

Python

Komentár	Notácia
Jednoriadkové	#
Nad premenné a metódy	”’ ….. ”’

STU FIIT

LEGENDRONE

Metodika pre komunikáciu

Tím:	#1, LEGENDRONE
Vedúci tímu:	Ing. Viktor Šulák
Členovia tímu:	Bc. Tomáš Čičman, Bc. Dušan Drábik, Bc. Dušan Hatváni, Bc. Patrik Kadlečík, Bc. Mário Keszeli, Bc. Martin Mocko, Bc. Lukáš Visokai
Akademický rok:	2016/2017
Autori:	Bc. Martin Mocko revidoval. Bc. Lukáš Visokai
Verzia číslo:	2
Dátum poslednej zmeny:	29.11.2016

LEGENDRONE

Obsah

1. Úvod

Účelom tejto metodiky je opísať a zadefinovať postupy komunikácie v tíme. Poskytuje informácie o správnom spôsobe komunikácie, určuje kanály (nástroje) pre rôzne typy komunikácie a uvádza postup pre správne použitie týchto kanálov. Metodika je určená pre všetkých členov tímu.

Komunikácia v tíme prebieha primárne prostredníctvom komunikačnej služby Slack, Skype, prostredníctvom nástroja na správu projektov TFS (Team Foundation Server), ďalej prostredníctvom skupiny na Facebooku a prostredníctvom komentárov v zdrojovom kóde.

2.Prehľad komunikačných kanálov

Nasledujúca tabuľka zobrazuje nástroje používané pri práci na projekte a ich využitie pri konkrétnych situáciách.

Činnosť/nástroj	Facebook	Slack	Telefón	TFS	Skype
Komunikácia s celým tímom	X	X			X
Manažovanie tímu		X			
Reportovanie o dokončenej činnosti		X		X	
Reportovanie o prebiehajúcej činnosti		X		X	
Nastavenie nástrojov		X		X	
Oboznámenie o novej metodike		X			
Pomoc pri probléme s implementáciou	X	X			X
Upresnenie si podrobností o úlohe	X	X			
Urgentný problém	X		X		X
Žiadosť o revíziu kódu		X		X	
Žiadosť o pomoc s úlohou		X		X	
Komunikácia k úlohe		X		X	X
Vytvorenie novej metodiky	X	X			

3.Komunikačné nástroje

Slack

Našim primárnym komunikačným nástrojom bude nástroj nazývaný Slack. Tento nástroj pokrýva veľkú väčšinu nepriamej komunikácie prebiehajúcej v tíme. Prostredie Slacku, ktoré využívame, je rozčlenené do niekoľkých hlavných komunikačných kanálov, pričom každý je zameraný na inú problematiku:

- #general
  - Dohodnutie stretnutia a upresnenie termínov
  - Dôležité upozornenia
- #random
  - Slúži na komunikáciu, ktorá sa týka vecí, ktoré nesúvisia priamo s tímovým projektom
- #tfs_logs
  - Slúži na automatizované rýchle získavanie informácií zo systému manažmentu projektov TFS
  - Každý člen tímu bude dostávať najnovšie informácie z diania na projekte
  - Prehľadnosť, kto kedy a čo robí
- #tp-2016-2017
  - Slúži na komunikáciu akýchkoľvek informácií o tímovom projekte, ktoré nezapadajú k inému typu kanálu
- #pull_request
  - “chcem poprosiť o code review“
  - „čítam si pripomienky k môjmu riešeniu úlohy a nerozumiem, ako boli myslené, ale nechcem spamovať TFS“
- #metodiky
  - “nerozumiem metodike”
  - “našiel som chybu v metodikách”
  - V metodikách niečo chýba
  - Chcel by som niečo napísať/povedať/urobiť, ale neviem kde a metodika o tom nehovorí
- #bugy
  - „našiel som chybu“
  - na komunikáciu bugov taktiež slúži aj nástroj TFS
- #otazky_k_implementacii
  - pri implementácii som narazil na problém s ktorým si neviem pomôcť
  - pri implementácii sa rozhodujem ako to spraviť ale sa neviem presne rozhodnúť akým spôsobom to budem implementovať

Facebooková komunikácia

Cez Facebook sa v spoločnej tímovej skupine riešia neformálne záležitosti okolo tímového projektu. Taktiež sa môže využiť ako sekundárny nástroj na komunikáciu akýchkoľvek informácií o tímovom projekte, pokiaľ sa takýmto spôsobom dá dospieť k rýchlejšej alebo efektívnejšej komunikácii ako pri nástroji Slack.

Telefonická komunikácia

Telefonicky sa riešia urgentné problémy a núdzové situácie, na ktoré je potrebná okamžitá spätná väzba. Telefonicky dostupný by mal byť počas dňa každý člen tímu, ak by nastala potreba vyriešiť bezodkladnú situáciu.

TFS komunikácia

Systém TFS je určený na komunikáciu primárne ohľadom stavu a priradenia úloh a taktiež projektového backlogu, burndown chartu a tímovej rýchlosti (velocity). Komunikácia ku konkrétnym úlohám sa však rieši v nástroji Slack.

Typy situácií riešených v TFS:

- Žiadosť o revíziu úlohy (kódu)
- Komunikovanie postupov práce ostatným členom tímu (commit komentáre pri commitovaní)

Komunikácia cez Skype

Skype môže nahrádzať fyzické stretnutia pri rôznych stretnutiach poprípade pri rôznych problémoch na projekte. Pomocou Skypu sa môžu zúčastňovať spoločných stretnutí aj členovia tímu, ktorí nemôžu byť fyzicky prítomný. Skype umožňuje zdieľanie obrazovky čo uľahčuje riešenie problémov na diaľku.

Žiadosť o revíziu úlohy

Po dokončení úlohy je v niektorých činnostiach potrebná kontrola od iného člena tímu. Pre upovedomenie člena tímu o tejto skutočnosti je potrebné použiť TFS, zároveň napísať aj do kanála #code-review. Pre viac informácií si prečítajte metodiku pre code review.

Do metodík patrí:

- Používanie podporných nástrojov (IDE, verziovanie softvéru, TFS, …),
- Spôsob písania dokumentov,
- Pomenovanie súborov,

- Pravidlá pri programovaní,
- Každá činnosť, ktorú tím musí vykonávať jednotne.

Pri pridávaní novej metodiky je potrebné postupovať nasledovne:

- Použiť vytvorenú šablónu na metodiky stiahnuteľnú zo stránky tímu
- Pridať odkaz na dokument s metodikou na stránku tímu
- Upovedomiť členov tímu o novej metodike prostredníctvom Slack kanála #metodiky

Pri úprave metodiky:

- Upraviť obsah dokumentu s metodikou
- Upovedomiť členov tímu o zmene metodiky prostredníctvom Slack kanála #metodiky

LEGENDRONE

Metodika pre code review a verziovanie

Tím:	#1, LEGENDRONE
Vedúci tímu:	Ing. Viktor Šulák
Členovia tímu:	Tomáš Čičman, Dušan Drábik, Dušan Hatváni, Patrik Kadlečík, Mário Keszeli, Martin Mocko, Lukáš Visokai
Akademický rok:	2016/2017
Autor:	Martin Mocko
Verzia číslo:	2
Dátum poslednej zmeny:	26.11.2016

STU FIIT

LEGENDRONE

Obsah

STU FIIT

LEGENDRONE

1. Úvod

Cieľom tejto metodiky je zosumarizovanie pravidiel a vytvorenie postupu pre vykonávanie code review. Vzhľadom na to, že kód aj po odtestovaní môže byť problematický, alebo je ho možné ešte upraviť do podoby, ktorá by viacej vyhovovala podmienkam projektu, je potrebné vykonávať kontrolovanie kódu.

Code review je pre fungovanie tímu veľmi dôležité, pretože prináša do tímu väčšiu istotu ohľadom implemenácie nejakej úlohy alebo príslušného riešenia. Je dôležité nielen z pohľadu kódera, ktorý kód vytvoril, aby dostal spätnú väzbu k jeho kódu, ale slúži taktiež na to, aby sme mali minimálne jedného ďalšieho člena tímu, ktorý je oboznámený s príslušným riešením danej úlohy. Takýmto spôsobom sa zdieľajú poznatky a pokiaľ by prišlo k výpadku člena tímu, nie je to pre tím kritické.

1.1. Roly vývojárov

Na vytváraní zdrojového kódu sa podieľajú vývojári, ktorých úlohy môžeme rozdeliť do dvoch základných rolí a to:

- Autor kódu – je osoba, ktorá vytvorila daný zdrojový kód a tým sa aktívne podieľala na vývoji systému
- Posudzovateľ – je osoba, ktorá zodpovedá za kontrolovanie zdrojového kódu, ktorý vytvoril „autor kódu“. Kontrolovanie môže prebiehať rôznymi formami, najčastejšie je to formou poznámok, ktoré sú následne predložené autorovi kódu

1.2. Slovník pojmov

Pojem	Význam
Branch (vetva)	Verzia/strom objektov, ktoré máme v záujme sledovať. Je robená na to aby sa na viacerých objektoch (súboroch) dalo pracovať paralelne.
Push	Skopírovať informácie z lokálnej vetvy do vzdialenej (od nás do nejakej inej).
Pull	Natiahnutie informácií zo vzdialenej vetvy do lokálnej vetvy (z nejakej inej ku nám).
Pull request	Slúži na formalizovanie (overenie) dokončenej úlohy. Úloha musí prejsť reviewom kódu a je potrebné vyriešiť konflikty.
Code review	Prehliadka kódu, ktorá určí, či je kód pripravený na mergnutie do rodičovskej vetvy.
Commit	Zachytenie nejakého "stavu" kódu.

2.Pravidlá code review

Vzhľadom na to, že existuje veľké množstvo spôsobov a pravidiel na vykonávanie code review, je potrebné si stanoviť pravidlá, ktoré zachovajú konzistentnosť a tým zvýšia efektivitu. Je dôležité aby pravidlá boli jednoduché a zrozumiteľné.

2.1. Pravidlá

Posudzovateľ

Posudzovateľ je zodpovedný za kontrolovanie kódu a informovanie o chybách autora kódu. Posudzovateľ by mal byť osoba, ktorá sa doteraz aktívne nepodieľala na vytváraní kódu. Je to z toho dôvodu, aby nebola ovplyvnená doterajším vývojom a tým mohla priniesť do problematiky vlastný názor. Code review by taktiež nemala vykonávať osoba, ktorá nemá o danej problematike žiadne vedomosti.

Časová následnosť

Posudzovať kontroluje a hodnotí kód až keď je vytvorený a odtestovaný. Je to z toho dôvodu, aby posudzovateľ nestrácal zbytočne čas kontrolovaním kódu, ktorý nefunguje. Úlohou posudzovateľa nie je opraviť chyby kódu, ale ich detekovať, prípadne navrhnúť riešenie.

2.2. Sumarizácia

- Posudzovateľ sa aktívne nepodieľa na vytváraní kódu
- Posudzovateľ posudzuje kód až po jeho vytvorení a odtestovaní
- Posudzovateľ neopravuje chyby, ale ich detekuje (výnimka v prípade, že identifikované chyby sú triviálne a oprava nezaberie viac ako 5 minút)
- Posudzovateľ by mal mať minimálne vedomosti o danej problematike

3.Ako požiadať o code review

Postup notifikácie člena tímu o žiadosti o code review. Nasledujúce kroky robí autor kódu:

1. Keď je úloha hotová, pushnúť commity na server cez Sync – Push outgoing commits
2. Po tom, ako splnil úlohu si autor kódu musí pullnúť najnovšiu verziu „master“ vetvy do svojej lokálnej „master“ vetvy
3. Túto lokálnu „master“ vetvu autor kódu mergne do svojej branche, na ktorej pracoval
4. Ak je to potrebné, autor kódu vyrieši merge konflikty
5. Po úspešnom vyriešení merge konfliktov môže autor kódu vytvoriť nový pull request cez Pull requests – New pull request (pre lepšiu predstavu pozrieť kapitolu 5.2 Vytvorenie pull requestu)
6. Označiť člena(ov) tímu, ktorý bude zodpovedný za review pull requestu
7. Potvrdiť požiadanie o pull request (tým pádom žiadame aj o code review)

4. Od vytvorenia tasku k zlúčeniu (merge) do projektu

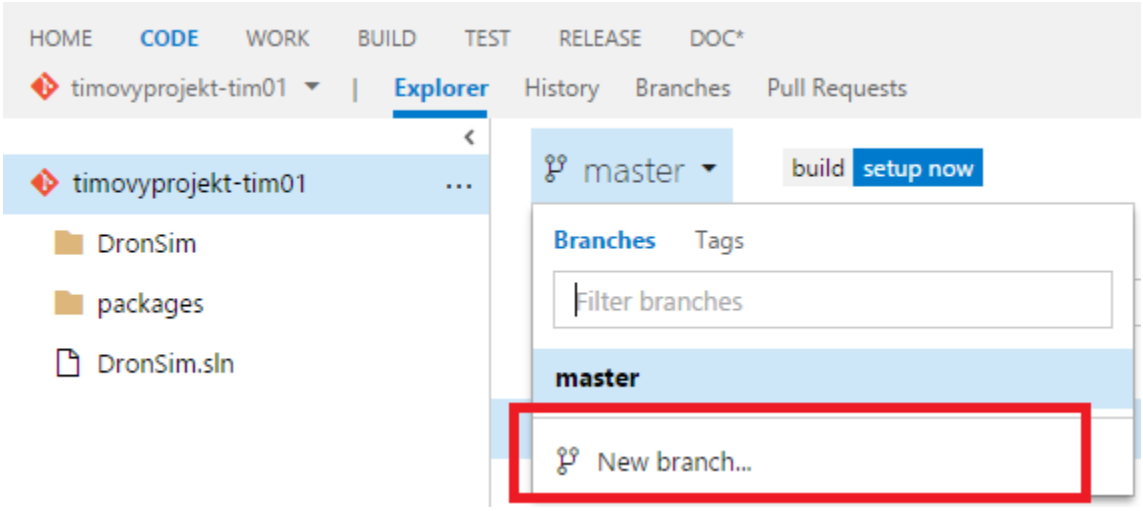
Pre efektívne fungovanie tímu musí taktiež byť v tíme jasne zadefinovaný postup od vytvorenia tasku až po mergnutie zdrojového kódu do projektu.

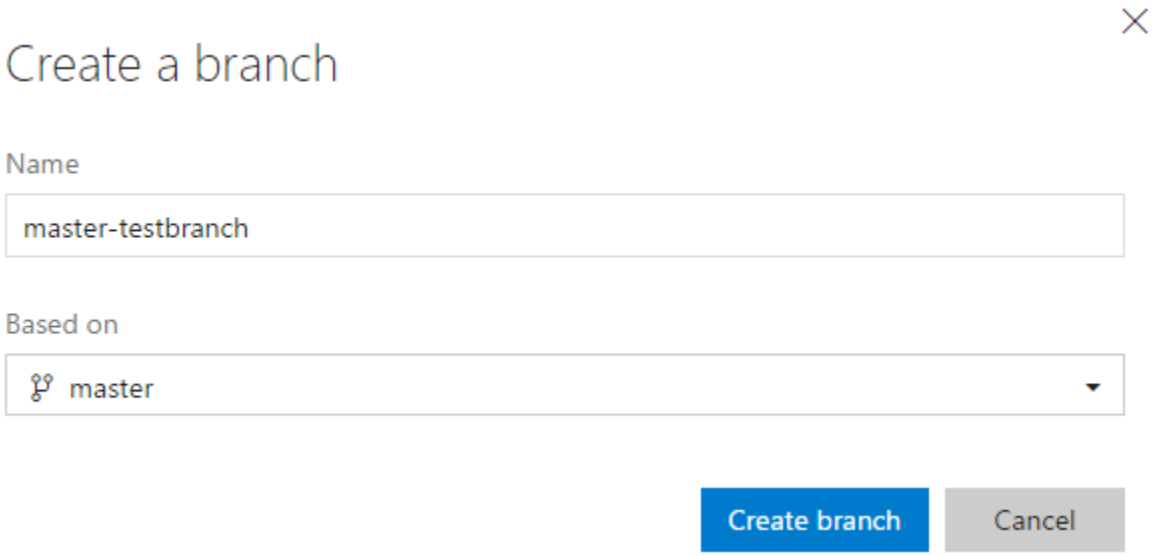
Tento postup bude vyzerať nasledovne:

1. Vytvorí sa úloha, priradí sa konkrétnemu členovi tímu
2. Autor kódu začne na tejto úlohe v systéme TFS pracovať – nastaví úlohu do stavu „In Progress“
3. Autor kódu si vytvorí novú branch z branche “master” a nazve ju spôsobom “<číslo_úlohy>-<názov_úlohy>”. Ak autor kódu implementuje viacej úloh, vytvorí branch spôsobom “<číslo_úlohy_1>-<číslo_úlohy_2>-..<číslo_úlohy_n>-<názov_úlohy>”. Čiže názov pre branch pre dve úlohy by vyzeral napr. **“5132-5135-Implementacia_modelu_UAV”.**
4. Autor kódu pracuje na úlohe, keď ju naimplementuje, môže spraviť commit, resp. check-in
5. Commit uložiť aj s komentárom, ktorý stručne vysvetlí spravené zmeny v kóde
6. Pokračuje sa pravidlami popísanými v 3.kapitole Ako požiadať o code review
7. Posudzovateľ si v nástroji TFS v sekcii “My Work” nájde nový review request
8. Posudzovateľ začne posudzovať review daných commitov, pokiaľ je kód v poriadku, akceptuje zmeny a pokračuje sa krokom 12, pokiaľ nie je kód v poriadku, pokračuje sa krokom 8
    a. Posudzovateľ sa pri posudzovaní sústredí na to, či daná implementácia spĺňa požiadavky úlohy, taktiež či je kvalita kódu dostatočná, či je kód riadne zdokumentovaný – viď metodiku dokumentácie
9. Kód nebol v poriadku, takže je potrebné poskytnúť dodatočnú spätnú väzbu autorovi kódu. Táto spätná väzba môže byť poskytnutá nasledujúcimi spôsobmi:
    a. Do komentára k danému code review requestu, ktorý posudzovateľ dostal
    b. Komentárom do autorovho kódu k častiami kódu, ktoré sú problematické
    c. Poskytnutím poznámok ku kódu iným spôsobom (napr. Napísať do Slacku)
10. Posudzovateľ vráti autorovi úlohu na prerobenie
11. Autor kódu upraví kód podľa pripomienok posudzovateľa. Pokračuje sa bodom 8.
12. Autor kódu, prípadne iný zainteresovaný človek vyrieši prípadné konflikty, ktoré mohli nastať pri pull requeste
13. Posudzovateľ akceptuje pull request, implementácia úlohy spĺňa požiadavky
14. Vytvorené a skontrolované riešenie sa mergne do branche “master”

5.Práca s branchami v TFS

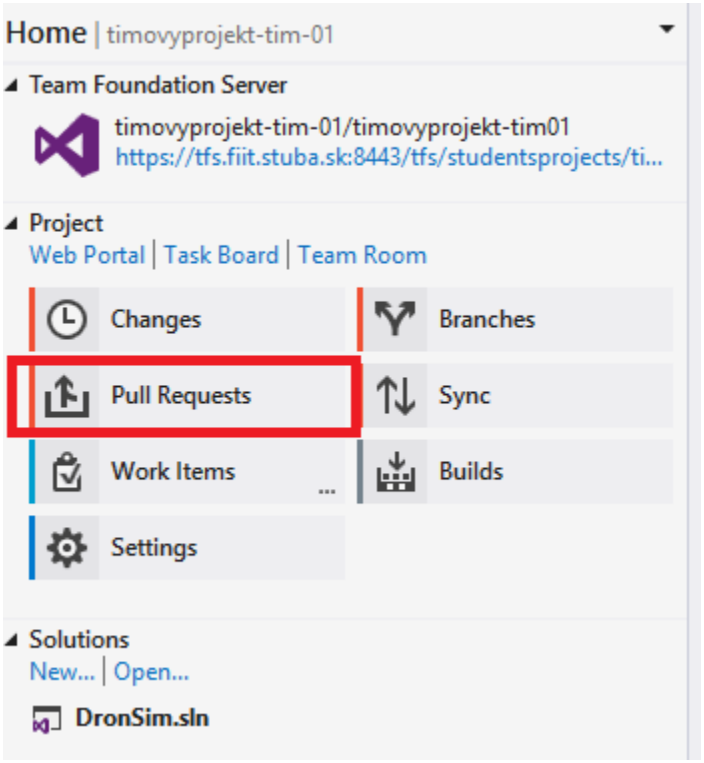
5.1. Vytvorenie novej branche

1. Ísť do CODE → Explorer → Zvoliť git repozitár projektu (napr. timovyprojekt-tim01) → Kliknúť na názov branche v ktorej sa aktuálne nachádzame (napr. „master“) → Kliknúť na New branch
2. 

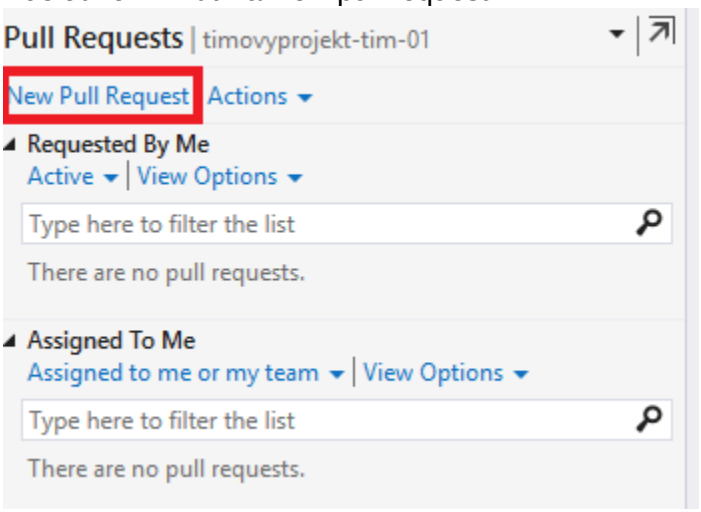
3. Vyplniť kolonku “Name” pre meno branche, zvoliť, z ktorej branche sa má nová brancha vytvoriť
4. 

5.2. Vytvorenie pull requestu

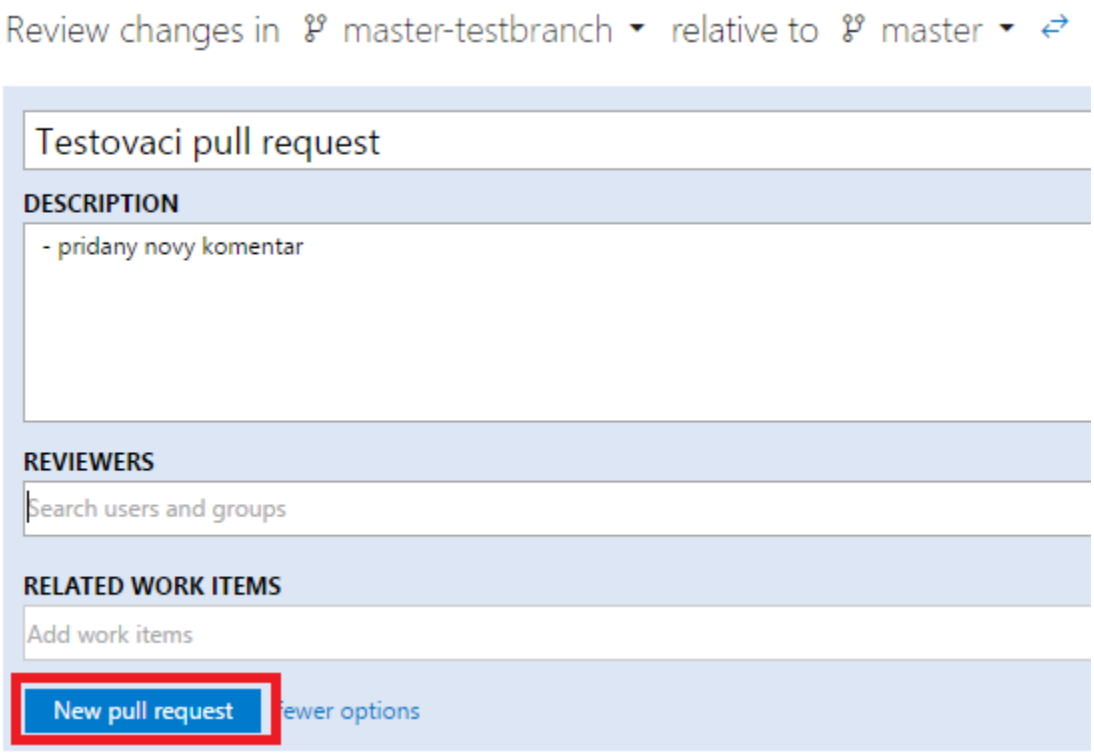
1. Nachádzame sa v Home, treba kliknúť na Pull requests



2.
3. Následne kliknúť na New pull request



4.
5. Otvorí sa webstránka, treba zadať názov pull requestu, kliknúť na more options, vyplniť description, pridať Reviewerov, pokiaľ niesú pridané súvisiace Work itemy, pridať aj tie, následne kliknúť na New pull request



6.
7. Pull request je týmto vytvorený a pripravený ku code review

LEGENDRONE

Metodika testovania

Tím:	#1, LEGENDRONE
Vedúci tímu:	Ing. Viktor Šulák
Členovia tímu:	Tomáš Čičman, Dušan Drábik, Dušan Hatváni, Patrik Kadlečík, Mário Keszeli, Martin Mocko, Lukáš Visokai
Akademický rok:	2016/2017
Autor:	Tomáš Čičman
Verzia číslo:	1
Dátum poslednej zmeny:	13.11.2016

STU FIIT

LEGENDRONE

Obsah

STU FIIT

LEGENDRONE

1. Úvod

Cieľom tejto metodiky je ukázať akým spôsobom bude prebiehať testovanie projektu, spolu s konvenciami, ktoré by mali byť pri testovaní dodržané a použité. V tejto metodike sa uvádza kedy, a akým spôsobom testovať daný projekt.

2. Súvisiace metodiky a manuály

Metodika testovania sa opiera a odkazuje na tieto metodiky:

- Metodika reportovania chýb
- Guide: Writing Testable Code

3.Súvisiace metodiky a manuály

Testovanie produktu bude vykonávané na dvoch úrovniach. Týmito úrovňami je úroveň kódu a vizuálna úroveň celkového riešenia.

3.1. Testovanie kódu

Pre testovanie softvéru z pohľadu kódu budú využité unit testy, ktoré bude developer vytvárať pre vlastný kód. Tieto testy budú dodržiavať nasledovné konvencie:

- budú vytvárané mimo hlavného projektu, konkrétne v projekte s názvom DronSimTests. Tento projekt má referenciu na hlavný projekt DronSim.

- budú vytvárané vo vedľajšom projekte v triede, ktorej názov bude pozostávať z názvu pôvodnej triedy z hlavného projektu, spolu s príponou Tests (napr. EnvironmentTests.cs) Každá z týchto tried bude mať pred svojou definíciou atribút [TestClass] ako to zobrazuje obrázok č. 1.

- budú vytvárané pre jednotlivé metódy, pričom testovacie metódy budú mať názov pôvodnej metódy spolu s príponou Test (napr. InicializeTest) Pred definíciou každej metódy bude uvedený atribút [TestMethod] ako to zobrazuje obrázok č. 1.

```
namespace DronSimTests
{
    [TestClass]
    0 references | 0 changes | 0 authors, 0 changes
    public class EnvironmentTests
    {
        [TestMethod]
        0 references | 0 changes | 0 authors, 0 changes
        public void TestMethod1()
        {
        }
    }
}
```

Obrázok 1 Vzor testovacej triedy

Pri písaní testu sa bude dodržiavať tzv. AAA (Arrange, Act, Assert) vzor, ktorý pozostáva z použitia nasledujúcich sekcií:

- Arrange - v tejto sekcií sa inicializujú objekty a nastavujú jednotlivé hodnoty dát, ktoré sú následne poskytnuté metóde v teste
- Act - táto sekcia pozostáva zo samotného volania testovacej metódy s parametrami z predchádzajúcej sekcie
- Assert - na tomto mieste sa uskutočňuje verifikácia, či výstup testovacej metódy zodpovedá očakávaniu

Príklad takejto metódy je zobrazený na obrázku č. 2.

```
[TestMethod]
public void Withdraw_ValidAmount_ChangesBalance()
{
    // arrange
    double currentBalance = 10.0;
    double withdrawal = 1.0;
    double expected = 9.0;
    var account = new CheckingAccount("JohnDoe", currentBalance);
    // act
    account.Withdraw(withdrawal);
    double actual = account.Balance;
    // assert
    Assert.AreEqual(expected, actual);
}
```

Obrázok 2 Vzor test metódy

Pre testovanie je potrebné pamätať na vhodnú implementáciu kódu pre jednoduché testovanie. Pre zrýchlenie testovania a zlepšenie jeho efektívnosti je potrebné odbremeniť konštruktor vytváranej triedy od vytvárania zbytočných objektov. Z tohto dôvodu budeme implementovať tzv. *dummy* triedy, ktoré budú obsahovať implementáciu iba metódy a dáta výhradne potrebné pre test.

3.2. Vizuálne testovanie

Na základe požiadaviek produkt ownera bude prebiehať testovanie produktu z pohľadu používateľa. Pri tomto testovaní sa bude klásť dôraz na všetky požiadavky zadané product ownerom, ako napríklad plynulosť simulácie, prekrývanie objektov, atď.

STU FIIT

LEGENDRONE

Metodika manažmentu úloh

Tím:	#1, LEGENDRONE
Vedúci tímu:	Ing. Viktor Šulák
Členovia tímu:	Tomáš Čičman, Dušan Drábik, Dušan Hatváni, Patrik Kadlečík, Mário Keszeli, Martin Mocko, Lukáš Visokai
Akademický rok:	2016/2017
Autor:	Patrik Kadlečík
Verzia číslo:	1
Dátum poslednej zmeny:	24.11.2016

STU FIIT

LEGENDRONE

Obsah

STU FIIT

LEGENDRONE

1. Úvod

Účelom tejto metodiky je opísať a zadefinovať postupy práce s úlohami v tíme. Jedná sa o postupy pri vytváraní, menení , vymazávaní , upravovaní úloh v nástroji TFS (Team Foundation Server). Metodika je určená primárne pre manažéra úloh ale môže slúžiť všetkým členom tímu.

Manažovanie úloh prebieha v nástroji TFS a zodpovedný za všetky úlohy , ktoré sa vyskytnú v projekte je manažér úloh.

2.Práca s úlohami v TFS

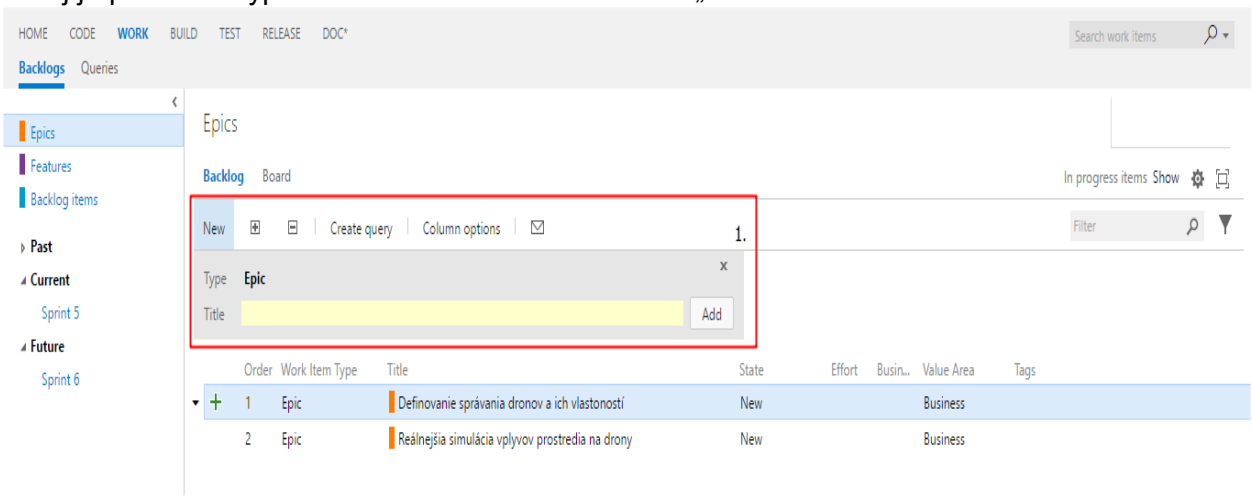
V tejto sekcii sú opísané postupy pre vytváranie, mazanie, upravovanie, popisovanie, správu úloh v TFS.

2.1. Vytvorenie novej úlohy

Pri vytváraní úlohy je potrebné si uvedomiť, aký typ úlohy ideme vytvárať. Pri projekte zameranom na vývoj softvéru máme identifikované 3 rôzne typy. „Epic“ , „Backlog Item – user story“ a „Task“.

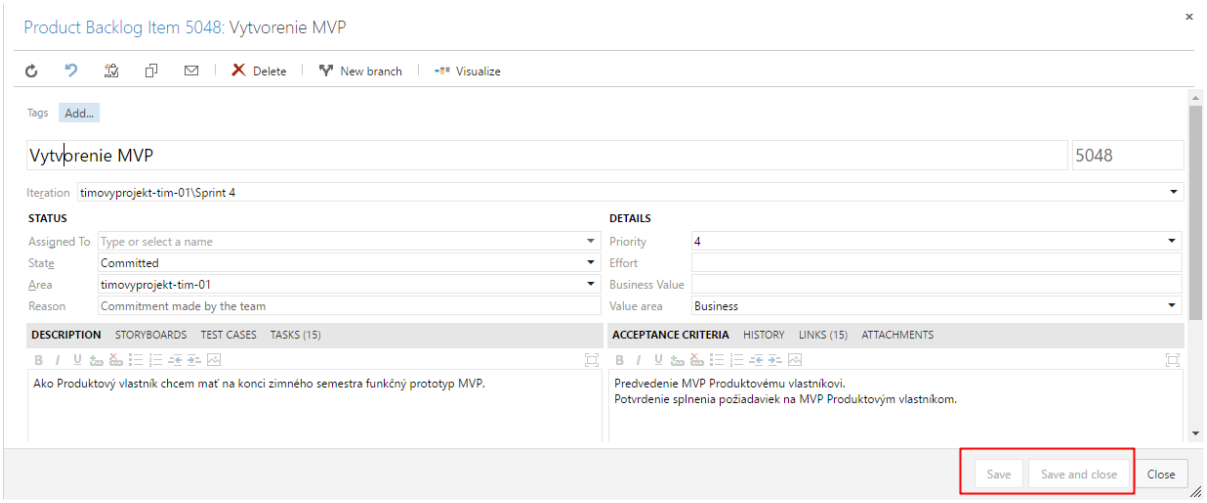
Úlohy sú vytvárané po konzultácii celého tímu s produktovým vlastníkom a scrum mastrom. Pri tejto konzultácii sa dohodnú typy úloh. Ďalej celý tím odhaduje náročnosť úloh. Úlohám je určená priorita a potom sú priradené určitým členom tímu. Všetky tieto atribúty sú zahrnuté do popisu úloh pri jej vytváraní alebo upravovaní.

1. V TFS je potrebné ísť do záložky WOKR -> Backlogs. Na ľavej strane obrazovky máme na výber z 3 typov. Podľa typu úlohy , ktorú chceme vytvoriť si zvolíme typ.
2. Ďalej je potrebné vyplniť názov a kliknúť na možnosť „Add“.



3. Po vytvorení úlohy je potrebné zadefinovať atribúty. Dvojklikom na určený typ úlohy sa zobrazí okno , kde je potrebné vyplniť atribúty , napísať akceptačné kritériá a popis úlohy.
4. Posledný krokom je uloženie zmien, kliknutím na možnosť „Save“ alebo „Save and close“ v pravom dolnom rohu dialógového okna úlohy.

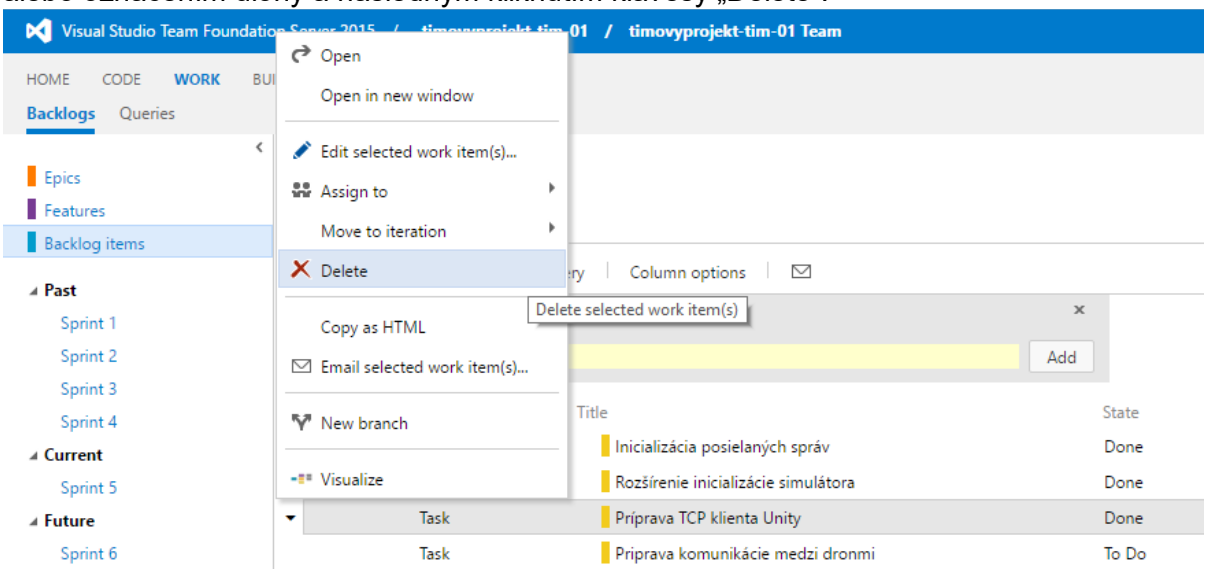
Po týchto krokoch je úloha vytvorená a učená na vykonávanie.



2.2. Zmazanie úlohy

Pri situácii , keď je potrebné zmazať úlohu je potrebné túto zmenu konzultovať s produktovým vlastníkom , scrum mastrom a členmi tímu. Po schválení tohto kroku je možné daný typ úlohy vymazať.

1. Najprv je potrebné si úlohu vyhľadať. V TFS je potrebné ísť do záložky WOKR -> Backlogs. Ak sa jedná o “Task” je potrebné si rozkliknúť „Backlog Item- user story“ v ktorej sa daná úloha nachádza a pravým tlačidlom na myši vybrať z kontextového menu „Delete“ alebo označením úlohy a následným kliknutím klávesy „Delete“.

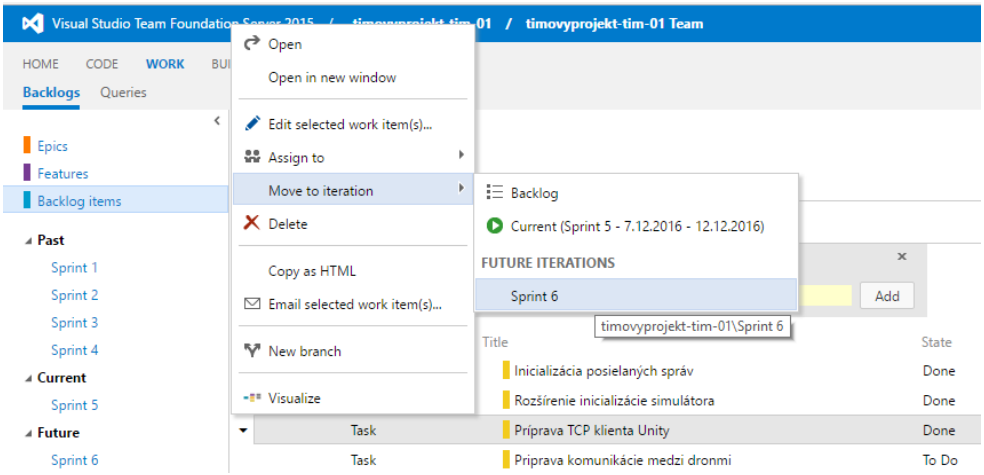


2.3. Správa úloh

V situácii, ak sa nepodarí dokončiť úlohu v danom šprinte je potrebné presunúť úlohu do nasledujúceho šprintu. Najskôr prebehne konzultácia ohľadom úlohy , v ktorej sa rozoberie dôvod nevykonania úlohy. Z výsledkov konzultácie sa určí ďalej , čo sa s úlohou bude diať.

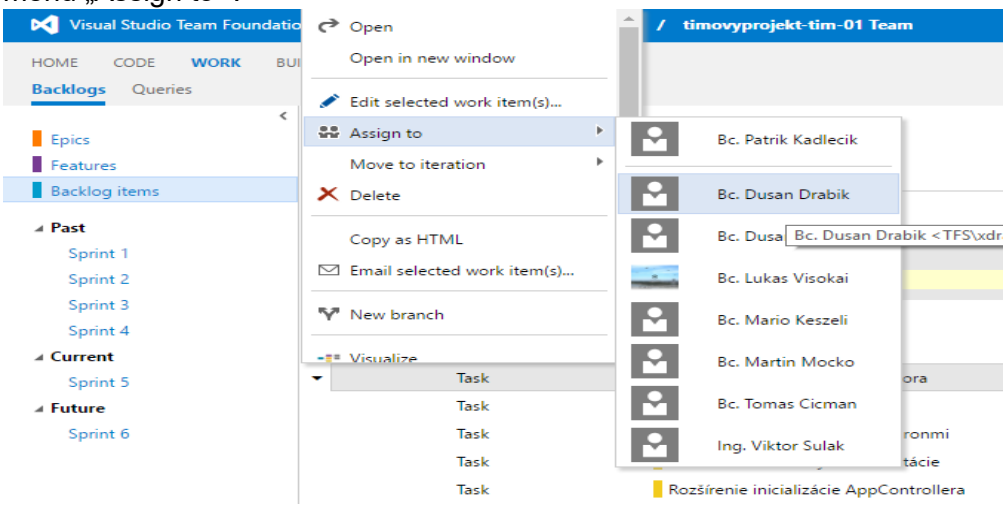
2.3.1 Presunutie úlohy do ďalšieho šprintu

1. Najprv je potrebné si úlohu vyhľadať. V TFS je potrebné ísť do záložky WOKR -> Backlogs. Ak sa jedná o “Task” je potrebné si rozkliknúť „Backlog Item- user story“ v ktorej sa daná úloha nachádza a pravým tlačidlom na myši vybrať z kontextového menu „Move to iteration“ .



2.3.2 Priradenie úlohy členovi tímu

1. Najprv je potrebné si úlohu vyhľadať. V TFS je potrebné ísť do záložky WOKR -> Backlogs. Ak sa jedná o “Task” je potrebné si rozkliknúť „Backlog Item- user story“ v ktorej sa daná úloha nachádza a pravým tlačidlom na myši vybrať z kontextového menu „Assign to“ .



B Export evidencie úloh

Project: timovyprojekt-tim-01 **Server:** tfs.fiit.stuba.sk\StudentsProjects **Query:** query_vsetko **List type:** Tree

ID	Work Item Type	Title 1	Title 2	Assigned To	State	Description	Priority	Created Date
4156	Task	Vytvoriť Class diagram bývalého riešenia - v dokumentácii je spravený naprd			To Do		1	01.10.2016 16:56
4450	Product Backlog Item	Zadefinovanie správania drona			Done		2	11.10.2016 9:13
4779	Task		Inicializácia Enviromentu	Bc. Tomas Cicman	Done		2	25.10.2016 13:56
4781	Task		Analýza správania - Python scripty	Bc. Mario Keszeli	Done	V rámci úlohy je potrebné zanalyzovať a komentovať Python scripty , ktoré slúžia na definovanie správania dronov.	2	25.10.2016 13:57
4782	Task		Analýza komunikácie - Python	Bc. Dusan Drabik	Done	V rámci úlohy je potrebné zanalyzovať a komentovať komunikáciu medzi časťou Enviroment a Simulátor.	2	25.10.2016 13:58
4783	Task		Finalizácia metodík tímu	Bc. Martin Mocko	Done	V rámci úlohy je potrebná finalizácia metodík v tíme , konkrétne finalizácia dokumentov k metodike komunikácie ,	2	25.10.2016 14:00
4848	Bug		Nehýbanie sa drona v UNITY - neposie	Bc. Lukas Visokai	Done		2	02.11.2016 13:03
4456	Product Backlog Item	Riešenie kolízií			Done	Ako Produktový vlastník chcem ovplyvňovanie pohybu dronov v simulácii pomocou kolízií s objektami v prostredí.	2	11.10.2016 9:54
5550	Task		Návrh a jednoduchá implementácia s	Bc. Tomas Cicman	Done	Story point -> 4 V rámci úlohy je potrebné analyzovať , navrhnúť a implementovať základnú funkccionalitu smerovacieho pr	2	06.03.2017 16:58
5805	Task		Kolízie so zemou	Bc. Patrik Kadlecik	Done	Story point -> 2 V rámci úlohy je potrebné implementovať funkcionalitu , ktorá bude riešiť kolíziu objektu UAV so zemou prc	2	23.03.2017 22:43
5806	Bug		Spustenie simulácie a Unity cez LAN	Bc. Dusan Hatvani	Done		2	23.03.2017 22:44
5807	Task		Kolízie objekt - dron	Bc. Patrik Kadlecik	Done	Story point -> 3 V rámci úlohy je potrebné implementovať funkcionalitu , ktorá bude riešiť kolízie objektov v prostredí a obje	2	23.03.2017 22:45
5808	Task		Kontrola kolízie pri inicializácii simulá	Bc. Lukas Visokai	Done	Story point -> 2 V rámci úlohy je potrebné implementovať funkciu na kontrolu kolízií pri inicializácii simulácie. Ak nastane ko	2	23.03.2017 22:47
5809	Task		Posielanie správ iba do zadefinovanej	Bc. Martin Mocko	Done	Story point -> 4 Zadefinovanie rôznych maximálnych vzdialeností pre rôzne typy komunikácií (bt, wifi, mobilne,..) Posielanie	2	23.03.2017 22:48
4457	Product Backlog Item	Spoľahlivá komunikácia		Bc. Dusan Hatvani	Done	Ako používateľ chcem mať spoľahlivú komunikáciu medzi simulátorom a grafickým vizualizátorom Unity (TCP komunikácia)	2	11.10.2016 9:54
4778	Task		Príprava TCP komunikácie na localhos	Bc. Dusan Hatvani	Done	V rámci úlohy je potrebné pripraviť projekt na realizáciu TCP komunikácie.	2	25.10.2016 13:53
4458	Product Backlog Item	Tímová réžia			Done	Ako Produktový vlastník chcem mať vytvorený tímový web , vytvorený komunikačný kanál pre tím.	2	11.10.2016 9:55
4459	Task		Metodika písania kódu	Bc. Lukas Visokai	Done	V rámci úlohy je potrebné vytvoriť dokument , ktorý popisuje metodiku písania kódu v tíme.	3	11.10.2016 9:57
4460	Task		Vytvorenie webovej stránky	Bc. Patrik Kadlecik	Done	V rámci úlohy je potrebné vytvoriť webovú stránku tímu, ktorá je dostupná na: http://labss2.fiit.stuba.sk/TeamProject/2016	2	11.10.2016 9:57
4461	Task		Metodika generovania dokumentácie	Bc. Mario Keszeli	Done	V rámci úlohy je potrebné vytvoriť dokument , ktorý popisuje metodiku generovania dokumentácie.	3	11.10.2016 9:58
4462	Task		Refaktoring zdrojového kódu Environ	Bc. Dusan Drabik	Done	V rámci úlohy je potrebné vykonať refaktoring zdrojového kódu z časti projektu Enviroment .	4	11.10.2016 9:58
4463	Task		Nasadenie webovej stránky	Bc. Tomas Cicman	Done	V rámci úlohy je potrebné nasadiť webovú stránku tímu na server. Webová stránka je dostupná na: http://labss2.fiit.stuba.s	4	11.10.2016 9:58
4464	Task		Refaktoring zdrojového kódu vizualiz	Bc. Dusan Hatvani	Done	V rámci úlohy je potrebné zrefaktorovať zdrojový kód časti vizualizátora Unity.	4	11.10.2016 9:59
4465	Task		Metodiky tímového projektu	Bc. Martin Mocko	Done	V rámci úlohy je potrebné vytvoriť šablónu metodík tímu, a vytvoriť potrebné metodiky pre manažment v tíme.	3	11.10.2016 10:00
4653	Task		Integrácia TFS a Unity	Bc. Lukas Visokai	Done	V rámci úlohy je potrebné vyhľadať možnosti integrácie zdrojového kódu Unity a TFS.	4	18.10.2016 9:46
4679	Task		Rešerš senzorov dronov	Bc. Patrik Kadlecik	Done	V rámci úlohy je potrebné vyhľadať možnosti využitia rôznych typov senzorov v dronoch. Je potrebné vytvoriť dokument o	4	18.10.2016 10:35
4680	Task		Rešerš použitia dronov	Bc. Lukas Visokai	Done	V rámci úlohy je potrebné vyhľadať možnosti a konvencie používania UAV v reálnom živote.	4	18.10.2016 10:36
4857	Task		Migrácia TFS projektu na GIT projekt		Done		2	07.11.2016 14:10
4903	Task		Kontinuálna integrácia	Bc. Martin Mocko	Done	V rámci úlohy je potrebné vyriešiť kontinuálnu integráciu v TFS.	4	09.11.2016 14:20
4897	Product Backlog Item	Vytváranie špecifikácie z požiadaviek			Done	Ako Produktový vlastník chcem vedieť , či moje požiadavky na vytváraný systém boli správne pochopené.	2	09.11.2016 13:37
4898	Task		Návrh formátu Logov	Bc. Mario Keszeli	Done	V rámci úlohy je potrebné navrhnúť formát správ generovaných systémom (Logov).	1	09.11.2016 13:59
4899	Task		Návrh SDK pre Python scripty	Bc. Dusan Hatvani	Done	V rámci úlohy je potrebné vytvoriť návrh SDK pre Python scripty.	2	09.11.2016 14:01
4900	Task		Návrh diagramov systému	Bc. Dusan Drabik	Done		4	09.11.2016 14:03
4901	Task		Návrh formátu správ posielaných pri	Bc. Lukas Visokai	Done	V rámci úlohy je potrebné navrhnúť formát správ , ktoré sa budú posielať medzi dronmi pri ich komunikácii.	3	09.11.2016 14:05
4902	Task		Návrh spôsobu simulácie	Bc. Lukas Visokai	Done	V rámci úlohy je potrebné navrhnúť spôsob simulácie dronov v prostredí simulátora.	2	09.11.2016 14:19
4904	Task		USE CASE diagram	Bc. Patrik Kadlecik	Done	V rámci úlohy je potrebné vytvoriť use case diagram použitia dronov.	1	09.11.2016 14:23
4905	Task		Návrh prostredia	Bc. Patrik Kadlecik	Done	V rámci úlohy je potrebné navrhnúť mapu prostredia , navrhnúť vplyv prostredia (vietor, deň-noc) na správanie dronov,	2	09.11.2016 14:28
4906	Task		Metodika testovania	Bc. Tomas Cicman	Done		1	09.11.2016 14:34
4907	Task		Návrh formátu správ medzi Envirome	Bc. Dusan Hatvani	Done	V rámci úlohy je potrebné navrhnúť formát správ posielaných medzi Enviromentom a Unity.	1	09.11.2016 14:35
4908	Task		Dokumentácia 1. kontrolného bodu	Bc. Mario Keszeli	Done	V rámci úlohy je potrebné dokončiť dokumentáciu k 1. kontrolnému bodu.	3	09.11.2016 15:04
4985	Product Backlog Item	Komunikácia medzi dronmi		Bc. Patrik Kadlecik	Done	Ako Produktový vlastník chcem mať funkčnú komunikáciu a posielanie správ medzi dronmi pri simulácii v prostredí.	2	15.11.2016 9:15
5552	Task		Logovanie spáv	Bc. Mario Keszeli	Done	Story point -> 2 V rámci úlohy je potrebné implementovať funkcionalitu poskytujúcu logovanie správ posielaných objektami	2	06.03.2017 17:01
5906	Task		Implementácia protokolu	Bc. Lukas Visokai	Done	Story point -> 5 V rámci úlohy je potrebné implementovať smerovací protokol.	2	03.04.2017 21:00
5907	Task		Implementácia protokolu	Bc. Tomas Cicman	Done	Story point -> 5. V rámci úlohy je potrebné implementovať smerovací protokol.	2	03.04.2017 21:01
5908	Task		Pohyb dronov po kolízii	Bc. Patrik Kadlecik	Done	Story point -> 3. V rámci úlohy je potrebné implementovať funkcionalitu , ktorá rieši pohyb objektu UAV po detekovaní kolíz	2	03.04.2017 21:03
5909	Task		Možnosť zadania vlastných IP	Bc. Dusan Hatvani	Done	Story point -> 3. V rámci úlohy je potrebné implementovať funkcionalitu zabezpečujúcu zadať vlastnú IP adresu pri spúšťaní	2	03.04.2017 21:05
5988	Task		Pridanie moznosti broadcastu sprav	Bc. Martin Mocko	Done	Pridanie možnosti posielania správy broadcastom, čiže všetkým dronom. Taktiež pre každého drona pridanie, že aké typy kc	2	09.04.2017 10:57
5048	Product Backlog Item	Vytvorenie MVP			Done	Ako Produktový vlastník chcem mať na konci zimného semestra funkčný prototyp MVP.	4	22.11.2016 10:13
5049	Task		Vytvorenie základnej štruktúry MVP	Bc. Dusan Drabik	Done	V rámci úlohy je potrebné vytvoriť základnú kostru projektu. Je potrebné implementovať hlavné triedy spolu s dôležitými at	2	22.11.2016 10:34
5050	Task		Vytvorenie sekvenčných diagramov	Bc. Dusan Drabik	Done	V rámci úlohy je potrebné vytvoriť sekvenčné diagramy a identifikovať základné správy, ktoré sa budú posielať medzi časťam	2	22.11.2016 10:35
5051	Task		Inicializácia Enviromentu	Bc. Tomas Cicman	Done	V rámci úlohy je potrebné implementovať základnú funkcionalitu Enviromentu a prepojiť časť Enviromentu s časťou AppCor	2	22.11.2016 10:35
5052	Task		Inicializácia Simulátora	Bc. Lukas Visokai	Done	V rámci je potrebné implementovať základnú funkcionalitu simulátora, teda aktualizácia polohy UAV.	2	22.11.2016 10:35
5053	Task		Vytvorenie modelu UAV	Bc. Mario Keszeli	Done	V rámci úlohy je potrebné spracovať Python skript obsahujúci správanie drona a sprístupniť jeho metódy C# objektom.	2	22.11.2016 10:36
5054	Task		Vytvorenie komunikátora	Bc. Dusan Hatvani	Done	V rámci úlohy je potrebné pripraviť potrebné triedy a atribúty na implementovanie základného komunikátora medzi simulát	2	22.11.2016 10:36
5055	Task		Inicializácia aplikácie (AppController	Bc. Patrik Kadlecik	Done	V rámci úlohy je potrebné inicializovať spustenie aplikácie. V triede AppController je potrebné rozdeliť hlavný konfiguračný s	2	22.11.2016 10:36
5112	Task		Inicializácia posielaných správ	Bc. Martin Mocko	Done	V rámci úlohy je potrebné implementovať triedu Message a potrebné atribúty. Je potrebné implementovať metódy na prijír	2	29.11.2016 10:04
5185	Task		Rozšírenie inicializácie simulátora	Bc. Lukas Visokai	Done	V rámci úlohy je potrebné rozšíriť funkcionalitu simulátora doplnením základných atribútov a metód na pôsobenie vetra na	2	06.12.2016 10:26
5186	Task		Príprava TCP klienta Unity	Bc. Dusan Hatvani	Done	V rámci úlohy je potrebné pripraviť základnú kostru TCP klienta na strane vizualizátora.	2	06.12.2016 10:28
5187	Task		Priprava komunikácie medzi dronmi	Bc. Dusan Drabik	Done	V rámci úlohy je potrebné pripraviť základnú štruktúru komunikácie medzi dronmi.	2	06.12.2016 10:34
5188	Task		Dokončenie finálnej dokumentácie	Bc. Mario Keszeli	Done	Je potrebné doplniť dokumentáciu o požadované informácie a pripraviť ju na finálne odovzdanie.	2	06.12.2016 10:35
5189	Task		Rozšírenie inicializácie AppControllera	Bc. Patrik Kadlecik	Done	V úlohe je potrebné rozšíriť funkcionalitu AppControllera a vytvoriť príslušné testy.	2	06.12.2016 10:37
5190	Task		Rozšírenie inicializácie Enviromentu	Bc. Tomas Cicman	Done	V úlohe je potrebné rozšíriť funkcionalitu Enviromentu. Je potrebné spracovať JSON objekt reprezentujúci mapu a objekty v	2	06.12.2016 10:38
5191	Task		Posielanie správ	Bc. Martin Mocko	Done	V rámci úlohy je potrebné rozšíriť funkcionalitu posielania a prijímania správ v triede Message.	2	06.12.2016 10:40
5252	Product Backlog Item	Jednoduchá vizualizácia			Done	Ako Produktový vlastník chcem zobrazovať simuláciu pomocou grafického vizualizátora UNITY.	3	15.02.2017 14:34
5548	Task		Preposielanie JSON	Bc. Dusan Hatvani	Done	Story point -> 3 V rámci úlohy je potrebné implementovať prijímanie JSON správ a ich spracovanie.	2	06.03.2017 16:56
5549	Task		Parsovanie JSON na strane UNITY	Bc. Lukas Visokai	Done	Story point -> 3 V rámci úlohy je potrebné implementovať parser na JSON objekty na strane vizualizátora UNITY.	2	06.03.2017 16:57
5551	Task		Riešenie kolízií objektov	Bc. Patrik Kadlecik	Done	Story point -> 3 V rámci úlohy je potrebné implementovať základné riešenie kolízií UAV a objektov v prostredí.	2	06.03.2017 17:00
5553	Task		Návrh a jednoduchá implementácia p	Bc. Martin Mocko	Done	Story point -> 3 V rámci úlohy je potrebné navrhnúť a implementovať základnú funkcionalitu spolupráce viacerých vlákien. K	2	06.03.2017 17:02
5554	Task		Ukončenie program spolu s ukončení	Bc. Dusan Hatvani	Done	Story point -> 1 V rámci úlohy je potrebné implementovať ukončenie simulátora spolu s vizualizátorom UNITY.	2	06.03.2017 17:24
5330	Product Backlog Item	Základná simulácia			Done	Ako Produktový vlastník chcem mať funkčnú základnú simuláciu.	2	20.02.2017 23:40
5331	Task		TCP klient na Unity strane	Bc. Dusan Hatvani	Done	Story-points -> 3. V rámci úlohy je potrebné vytvoriť TCP klienta na strane vizualizátora UNITY 3D.	2	20.02.2017 23:41
5332	Task		Inicializácia spojenia	Bc. Dusan Hatvani	Done	Story-points -> 2. V rámci úlohy je potrebné inicializovať TCP spojenie medzi simulátorom a vizualizátorom UNITY	2	20.02.2017 23:42
5333	Task		Základné logovanie správ	Bc. Mario Keszeli	Done	Story-points -> 2. V rámci úlohy je potrebné pripraviť základnú logiku pre logovanie správ posielaných dronmi.	2	20.02.2017 23:43
5334	Task		Inicializácia simulátora s vizualizátoro	Bc. Lukas Visokai	Done	Story-points -> 3. V rámci úlohy je potrebné inicializovať simulátor a vizualizátor UNITY	2	20.02.2017 23:44
5335	Task		Deserializácia objektu UAV na JSON f	Bc. Martin Mocko	Done	Story-points -> 2. V rámci úlohy je potrebné vytvoriť deserializáciu objektu typu UAV na JSON formát.	2	20.02.2017 23:45
5336	Task		Neduplikované spúšťanie Unity	Bc. Tomas Cicman	Done	Story-points ->3. V rámci úlohy je potrebné odstrániť duplikované spúšťanie UNITY pri viacnásobnom spustení simulátora s v	2	20.02.2017 23:45

5337	Task	Úprava inicializačnej obrazovky systé	Bc. Patrik Kadlecik	Done	Story-points -> 2. V rámci úlohy je potrebné upraviť inicializačnú obrazovku konzolovej aplikácie s možnosťou si vybrať	2	20.02.2017 23:46
6171	Product Backlog Item	Používateľská príručka		Approved	Ako Produktový vlastník chcem mať vytvorenú finálnu dokumentáciu obsahujúcu používateľskú príručku , inštalačnú príruč	3	24.04.2017 16:22
6172	Task	Používateľská príručka	Bc. Dusan Hatvani	Done	Story point -> 1. V rámci úlohy je potrebné napísať používateľskú príručka systému a doplniť ju do dokumentácie.	2	24.04.2017 16:24
6173	Task	Komentovanie kódu	Bc. Patrik Kadlecik	Done	Story point -> 1. V rámci úlohy je potrebné okomentovať celý zdrojový kód systému a vygenerovať dokumentáciu zdrojovéh	2	24.04.2017 16:25
6174	Task	Finalizácia dokumentácie	Bc. Mario Keszeli	Done	Story point -> 3. V rámci úlohy je potrebné doplniť dokumentáciu o potrebné informácie.	2	24.04.2017 16:26
6175	Task	Inštalačná príručka	Bc. Martin Mocko	Done	Story point ->1. V rámci úlohy je potrebné napísať inštalačnú príručku k systému a doplniť ju do dokumentácie.	2	24.04.2017 16:27
6176	Task	Finálne aktualizovanie tímového web	Bc. Tomas Cicman	Done	Story point ->1. V rámci úlohy je potrebné aktualizovať tímovú web stránku a doplniť tam potrebné dokumenty.	2	24.04.2017 16:28
6177	Task	Finalizácia dokumentácie	Bc. Lukas Visokai	Done	Story point -> 2. V rámci úlohy je potrebné doplniť dokumentáciu o potrebné informácie.	2	24.04.2017 20:58