

Slovenská technická univerzita v Bratislave  
Fakulta informatiky a informačných technológií

# Projektová dokumentácia

Tím 1: LEGENDRONE

---

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

# Obsah

## DOKUMENTÁCIA K RIADENIU

---

<b>1 Úvod</b>	<b>4</b>
<b>2 Členovia tímu a ich roly</b>	<b>5</b>
2.1 Predstavenie členov tímu	5
2.2 Rozdelenie manažérskych úloh	7
2.3 Podiel práce na jednotlivých častiach dokumentácie	8
<b>3 Aplikácie manažmentov</b>	<b>10</b>
<b>4 Sumarizácie šprintov</b>	<b>13</b>
4.1 Šprint 1	13
4.2 Šprint 2	13
4.3 Šprint 3	13
4.4 Šprint 4	14
4.5 Šprint 5	14
<b>5 Používané metodiky</b>	<b>15</b>
5.1 Metodika dokumentácie	15
5.2 Metodika písania kódu	15
5.3 Metodika komunikácie	15
5.4 Metodika prehliadok kódu a verziovania	15
5.5 Metodika testovania	16
5.6 Metodika úloh	16
<b>6 Globálna retrospektíva</b>	<b>17</b>

## DOKUMENTÁCIA INŽINIERSKEHO DIELA

---

<b>7 Úvod</b>	<b>20</b>
<b>8 Ciele pre zimný semester</b>	<b>21</b>
<b>9 Analýza</b>	<b>22</b>
<b>10 Návrh</b>	<b>27</b>
<b>11 Implementácia</b>	<b>33</b>
<b>12 Testovanie</b>	<b>36</b>
<b>Prílohy</b>	<b>37</b>
A Metodiky	37
B Export evidencie úloh	77

# **Dokumentácia k riadeniu**

# 1 Úvod

Dokument vznikol v rámci predmetu Tímový projekt a opisuje prácu na projekte *Simulácia správanía 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 kolaboratívne 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 aplikácie pre platformu iOS a využitiu 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 pracuje 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

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

### Manažment komunikácie

Komunikáciu môžeme 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 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 neformálnu dohodu 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 rámci 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 zaviazame, 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 zabezpečené 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, prekryvania 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, priradovať a dokončovať úlohy. Úlohy sa vytvárajú na základe konzultácii pred začiatkom šprintu. Metódou "planning poker" určujeme náročnosť úloh. Úlohy sú priradované 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 zadané 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 prideliť 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 jednomyselne zhodli, že nebudeme pokračovať v rozširovaní projektu bývalého tímu, ale navrhujeme 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 zadané 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.

## 5 Používané metodiky

Pre správne fungovanie tímu a jasne zadané 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 vytváratej 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 v zimnom semestri. Počas zimného 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. Počas práce na projekte sme zaviedli metodiky a procesy, ktoré sme sa snažili dodržiavať. V súčasnosti úspešne dodržiavame procesy verziovania kódu a kontinuálnej integrácie, avšak potrebujeme zlepšiť procesy súvisiace s kvalitou kódu a tvorbou testov.

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

Nepodarilo sa nám aplikovať:

- zahrnutie základne do implementácie

### 6.1 Retrospektíva po šprintoch

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
- Utužovanie kolektívu
- Ranné kávy
- 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

# **Dokumentácia inžinierskeho diela**

## 7 Úvod

Obsahom tejto dokumentácie je pohľad na technickú stránku projektu vypracovaného v rámci predmetu Tímový projekt na tému Simulácia správania bezpilotných lietadiel v roji. Téma je pokračovaním minuloročného projektu. Cieľom je zdokonalenie nástroja na simuláciu správania bezpilotných zariadení v roji, zlepšenie ich vzájomnej komunikácie a rozšírenie simulácie o senzory umožňujúce efektívne plnenie úloh.

Kapitola 8 opisuje ciele, ktoré sme si stanovili pre zimný semester. Obsahuje tiež požiadavky na systém a globálne ciele. Základným cieľom pre zimný semester je vytvorenie MVP prototypu aplikácie.

Obsahom kapitoly 9 je analýza projektu. V jej časti sa venujeme analýze existujúceho projektu spolu s vykonanými zmenami. Analýza obsahuje aj výsledky rešerše využitia bezpilotných lietadiel spolu s rôznymi senzormi, ktorými bezpilotné lietadlá disponujú.

Kapitola 10 sa zameriava na celkový pohľad na nový systém, ktorého súčasťou je architektúra, diagram tried a moduly systému. Jednotlivé časti sú popísané a vyobrazené na diagramoch v tejto kapitole.

# 8 Ciele pre zimný semester

## Hlavné ciele pre zimný semester pred šprintom číslo 3

- analýza preberaného projektu
- pochopenie a zdokumentovanie preberaného projektu (refaktoring, komentovanie zdrojového kódu a príprava na doimplementovanie nových funkcionalít)
- zefektívnenie správania celého systému

Po šprinte č. 2 pri prediskutovaní kladov a záporov projektu bývalého tímu, sme sa rozhodli, po odsúhlasení vedúcim projektu, vytvoriť simulátor nanovo. Po tomto rozhodnutí sme si definovali nové požiadavky na systém a MVP pre zimný semester.

## Požiadavky na projekt

- definovanie UAV (určovanie pozície pomocou GPS, pohyb - rýchlosť, smerový vektor, kapacita a spotreba batérie)
- navrhnutie a vytvorenie spoľahlivej komunikácie (medzi-komponentová)
- definovanie senzorov (implementovať základné senzory, navrhnuť systém pre pridanie rôznych senzorov)
- definovanie prostredia (vytvorenie mapy a objektov reálneho sveta, simulovanie poveternostných podmienok)
- kolízie objektov s UAV
- navrhnuť a implementovať komunikáciu medzi UAV (na aplikačnej vrstve, rádius dosahu komunikácie)

## MVP pre zimný semester

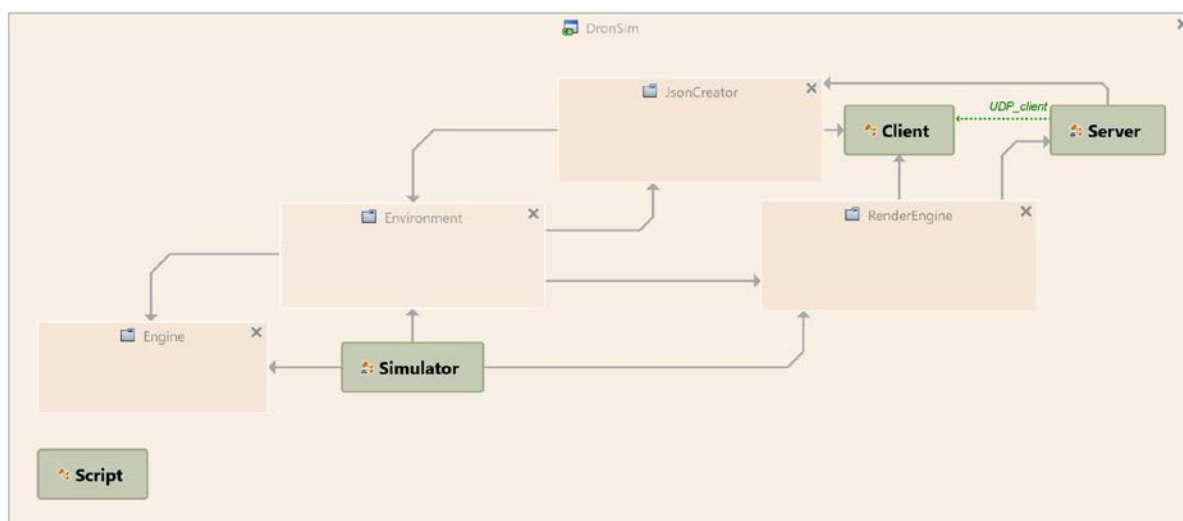
Za cieľ v zimnom semestri sme si zvolili vytvorenie prototypu aplikácie, ktorý bude spĺňať základné funkcionality:

- navrhnutie modulárneho systému pre jednoduché pridávanie nových funkcionalít
- inicializácia prostredia (mapa, základňa, UAV)
- vytvorenie UAV a jeho základný pohyb po mape
- definovanie správ medzi dvoma UAV ako aj medzi UAV a základňou
- simulácia funkčnosti preposielania a zachytávania akýchkoľvek správ
- zobrazenie správneho správania sa aplikácie formou logov a zobrazenie do grafického používateľského rozhrania

# 9 Analýza

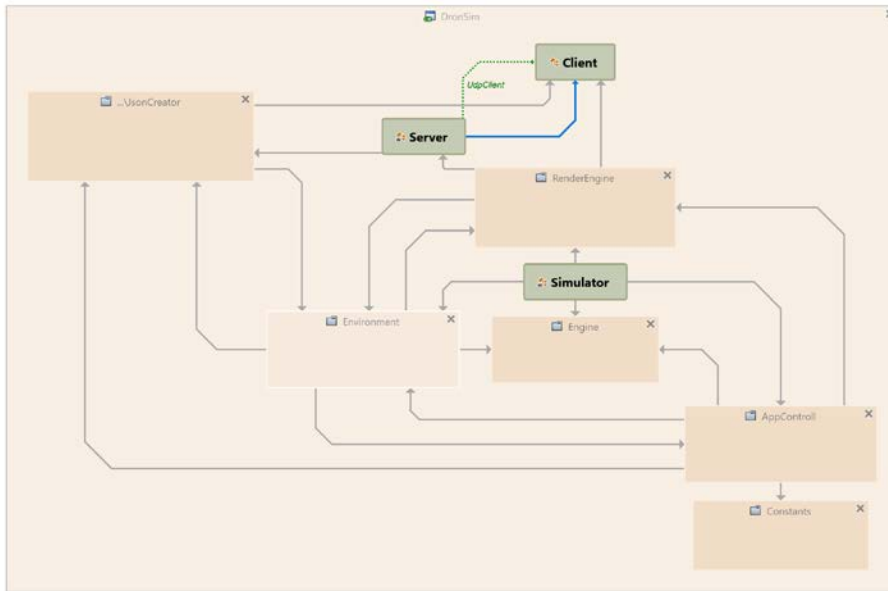
## Analýza existujúceho riešenia

Pri analýze existujúceho riešenia bývalého tímu sme začali s naštudovaním dokumentácie a zdrojového kódu. Pri študovaní dokumentácie sme zistili, že z nezanedbateľnej časti nezodpovedá dodanému projektu a jeho zdrojovému kódu. Diagram tried v dokumentácii taktiež nereprezentoval skutočnú architektúru a štruktúru kódu. Konflikt sme objavili aj pri triede zodpovednej za spúšťanie celého simulátora. Zatiaľ, čo v dokumentácii je uvedené, že trieda Engine je spúšťačom celej aplikácie (konkrétne metóda `onStart(script: Script): int`), po inšpekcii zdrojového kódu triedy Engine sme zistili, že táto trieda je prázdna a nič nespúšťa. Následne sme pokračovali s naštudovaním zdrojového kódu bez dokumentácie. Zistilo sa pri tom, že spúšťačom aplikácie je metóda `Main(string[] args)` v triede Simulator. V tejto triede sa vykonávalo vytvorenie prostredia (Environment) a jeho inicializácia. Následne sme po dôkladnej analýze zdrojového kódu pristúpili k zlepšovaniu kvality kódu (refactoring). Obrázok 1 zachytáva pohľad na pôvodnú vysokoúrovňovú štruktúru modulov a prepojení.



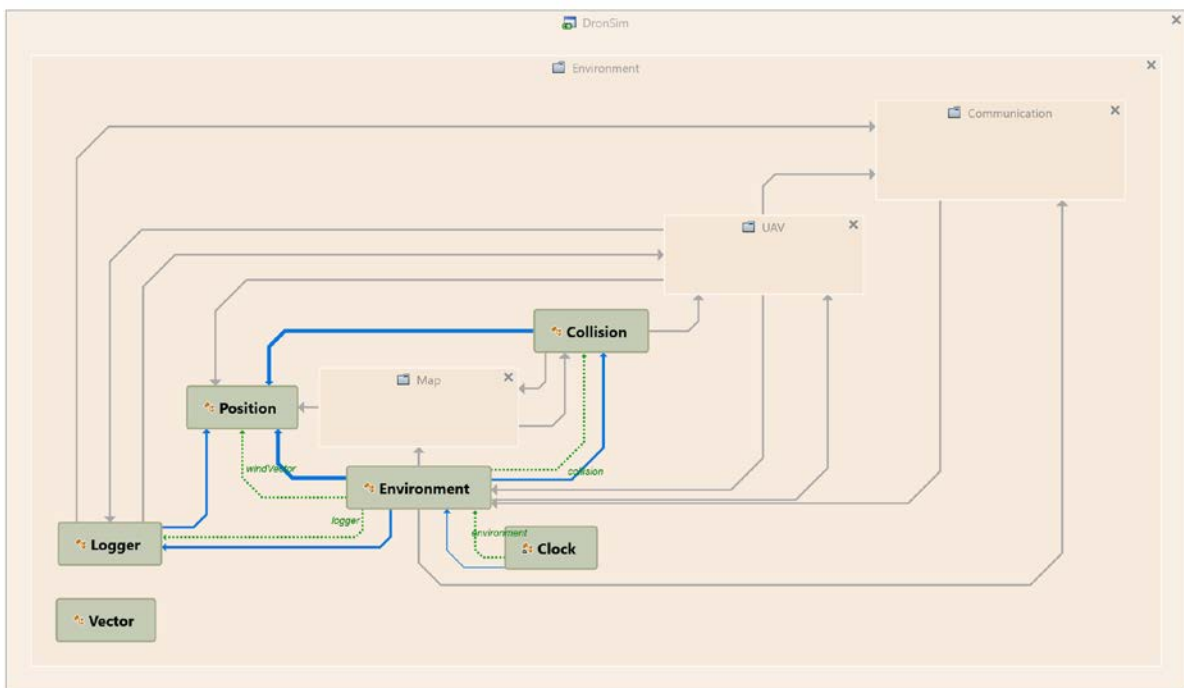
Obr. 1: Celkový pohľad na vysokoúrovňovú štruktúru pôvodného projektu bývalého tímu

Na obrázku 2 vidno prepracovanú štruktúru prepojení a modulov spolu s ich vzťahmi po našej úprave pôvodného kódu. Spúšťačom aplikácie zostala trieda Simulator, ktorá volala metódu `Main(string[] args)`. V nej sa vytvorí objekt triedy AppControl inicializujúci Environment a prípadne aj Unity pre grafické používateľské rozhranie simulátora. Oproti pôvodnej štruktúre pribudli dva moduly AppControl a Constants.



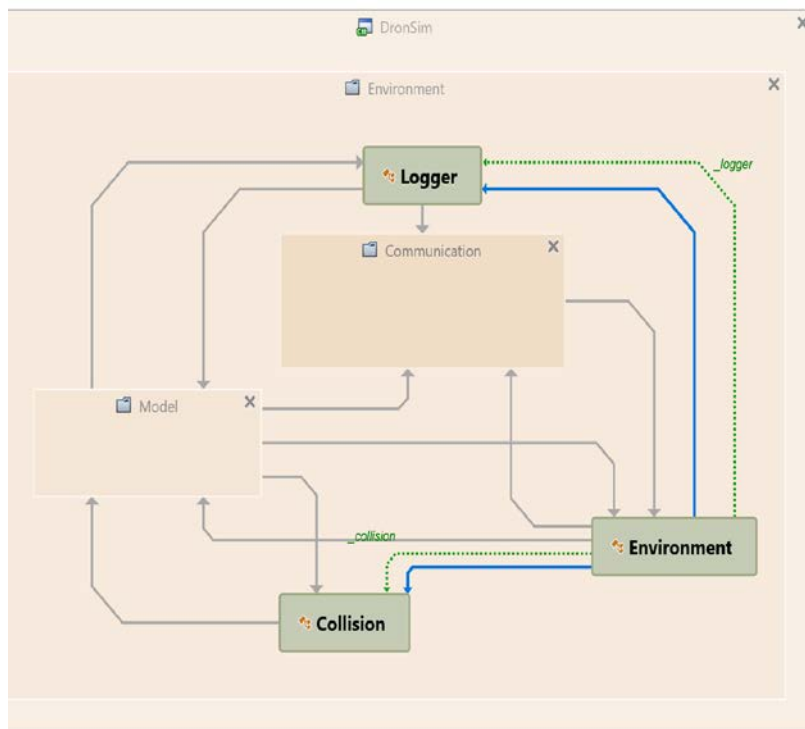
Obr. 2: Celkový pohľad na vysokoúrovňovú štruktúru upraveného projektu

Obrázok 3 znázorňuje pôvodnú štruktúru modulu Environment, ktorý obsahoval navyše zložky UAV, Communication a Map.



Obr. 3: Architektúra modelu Environment pôvodného projektu

Obrázok 4 zobrazuje upravenú modulu Environment. V upravenej časti sme nechali pôvodnú zložku Communication, ale pridali sme novú zložku Model. Spravili sme tak pre lepšiu organizáciu tried do skupín, ktoré spolu súvisia. Zložka Model obsahuje dve ďalšie zložky, ktoré boli pôvodne v modeli Environment - zložky UAV a Map. UAV navyše obsahuje novú súčasť UAV\_State, ktorá definuje aktuálny stav UAV.



Obr. 4: Architektúra modelu Environment po úprave a refaktoringu projektu

## Analýza použitia bezpilotných lietadiel

Bezpilotné lietadlá (angl. unmanned aerial vehicle - UAV) sú v súčasnosti čoraz viac rozšírené v rôznych oblastiach a spôsoboch použitíach. Vývoj dronov bol iniciovaný pre armádne využitie najmä kvôli ochrane vlastných pilotov. Môžu byť pri tom nasadené na špehovanie nepriateľského prostredia pomocou rôznych senzorov, prípadne ochranu hraníc, ale aj ako útočné lietadlá. Použitie na súkromné účely najčastejšie zahŕňa natáčanie videa a fotografovanie. Vo verejnej sfére však pre ne existujú podstatne širšie možnosti nasadenia. Obrovský prínos predstavujú napríklad pre záchranné alebo policajné zložky, ktorým uľahčujú prehľadávanie prostredia aj v náročnom teréne, či pri nevyhovujúcich poveternostných podmienkach. Veľkou výhodou je cena UAV, keďže náklady na jeho obstaranie sú výrazne nižšie ako napríklad pri obstarávaní helikoptéry. Takisto je známe aj využitie týchto zariadení v poľnohospodárstve, či už pri mapovaní poľnohospodárskej pôdy, alebo pri chove zvierat, kde sa pomocou bezpilotných lietadiel môže mapovať pohyb a poloha zvierat a taktiež ochraňovať ich napr. pred pytlíkmi. Taktiež majú veľký význam pre meteorológov. Bepilotné lietadlá bez ohrozenia ľudskej posádky možno vyslať na prieskum hurikánov a analyzovať a spracovávať nazbierané dáta v reálnom čase. V rôznych sférach je tiež užitočné prehľadávanie miestností rojom dronov alebo vytváranie mapy priestorov prípadne aj v spolupráci s UGV (pozemné vozidlo).



## Rešerš použitia senzorov v bezpilotných lietadlách

Existuje množstvo senzorov, ktoré je možné pripojiť k dronom a tým zvýšiť ich efektívnosť pri plnení úloh. Základným prvkom drona je systém, pomocou ktorého sa dokáže orientovať v danom prostredí. Väčšina dronov používa globálny satelitný navigačný systém (GNSS), ktorého hlavnou časťou môže byť GPS (Global Positioning System), satelitná navigácia (INS) alebo laserový diaľkomer. Pri plnení rôznych typov úloh ako simultánna lokalizácia a mapovanie vnútorných priestorov sa používajú miniatúrne laserové senzory (HokuyoURG a IMU Xsens MTI-G) a ultrazvukové senzory. Na orientáciu vo vnútornom prostredí objektu sa používa magnetometer tiež nazývaný elektronický kompas. Ako už bolo spomenuté, bohaté uplatnenie nachádzajú UAV v armáde obzvlášť pri sledovaní cieľov. Úlohy týkajúce sa sledovania cieľov môžeme rozdeliť na dve skupiny:

1. Automatické sledovanie statických cieľov
2. Sledovanie pohyblivých cieľov pohybujúcich sa rýchlosťou do 5 km/h

Pri automatickom sledovaní statických cieľov sa väčšinou používajú tri typy systémov:

1. Systém GALIFIR – hlavnou zložkou systému je infra-kamera s teleobjektívom s dvomi zornými poľami.
2. Systém STIS (Stabilised Thermal Imaging System) – hlavnou zložkou tohto systému je senzor MINI-FLIR s teleobjektívom s tromi ohniskovými vzdialenosťami, ten je doplnený nízkoúrovňovou CCD TV kamerou a laserovým diaľkomerom.
3. MKD 400/600 elektro-optické zariadenie, ktoré obsahuje infra-kameru a CCD TV kameru

Ďalšou možnosťou využitia v armáde je sledovanie pohyblivých cieľov pohybujúcich sa rýchlosťou do 5 km/h. Pri tomto type úlohy sa používajú systémy, ktorých hlavnou zložkou je radar. Často používané systémy na sledovanie pohyblivých cieľov:

1. SAR (Synthetic Aperture Radar) radar
2. TESAR (Tactical Endurance Synthetic Aperture Radar) – systém obsahujúci zariadenie pre automatické rozpoznávanie pozemných cieľov
3. STARLOS (SAR Target Recognition and Location System)
4. EL/M-2055 radar
5. Rádiolokačný senzor AWARD (All Weather Airborne Reconnaissance Drone Sensor)

Armádne nasadenie dronov bolo hlavnou motiváciou k skonštruovaniu dronov. Popri použití v armáde však existuje mnoho ďalších odvetví, kde nájdu svoje uplatnenie. Medzi ne patrí poľnohospodárstvo, monitorovanie ovzdušia, monitorovanie znečistenia pôdy, resp. vody, monitorovanie sopečných aktivít, či monitorovanie poškodenia elektrického vedenia. Drony používané na meranie koncentrácie plynu v ovzduší sú vybavené GSS (Gas Sensing System) systémom, ktorý pracuje na základe MEMS senzoru plynu USM-VOC 3.0 od firmy Unitronic. Tento senzor meria koncentráciu CO<sub>2</sub>, NO<sub>2</sub> a SO<sub>2</sub> v ovzduší. Na monitorovanie nebezpečných chemických a toxických látok v prostredí sa používajú drony obsahujúce LSCAD (Lightweight Stand-off Chemical Agent Detector) detektor, ktorý obsahuje televíznu kameru a infračervený interferometer.

Drony monitorujúce morské prostredie, morské aktivity a riziká (napr. tsunami) sú vybavené video kamerou s vysokým rozlíšením, výškovým sensorom, sensorom barometrického tlaku, sensorom vlhkosti a meteorologickým sensorom. Ďalším dôležitým odvetvím, kde sa drony používajú, je monitorovanie znečistenia vody. Pri tomto type úlohy nesú drony fotoaparát Canon 50D spolu s niekoľkými multispektrálnymi kamerami. Na monitorovanie sopečných aktivít v oblastiach s vysokou pravdepodobnosťou výskytu sopečnej aktivity sa používajú drony s termálnymi kamerami TC36000. Tieto typy kamier sledujú teplotu bahennej pôdy popri sopkách.

Zaujímavým využitím dronov a ich sensorov je použitie na monitorovanie typu vegetácie a pôdy v ťažko dostupných lokalitách. Takéto drony obsahujú multispektrálne kamery, ktoré pracujú na princípe odrazu svetla od rôznych typov vegetácie. Pri monitorovaní fotovoltických systémov a elektrického vedenia sú drony vybavené termovíznymi kamerami, ktoré detegujú praskliny a detegujú oblasti s vysokou teplotou. Drony monitorujúce elektrické vedenie používajú ultrafialové kamery a infračervené kamery typu FLIR, ktoré majú za úlohu merať infračervené emisie v okolí elektrických vodičov.

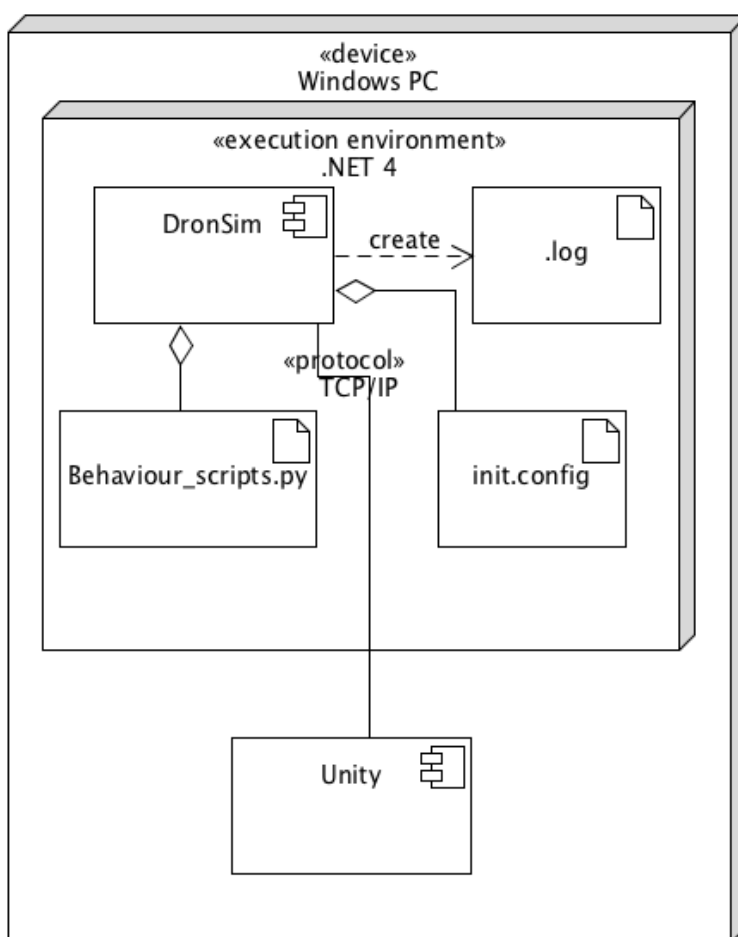
# 10 Návrh

V tejto kapitole rozoberieme podrobný návrh novej architektúry aplikácie.

## Diagram nasadenia

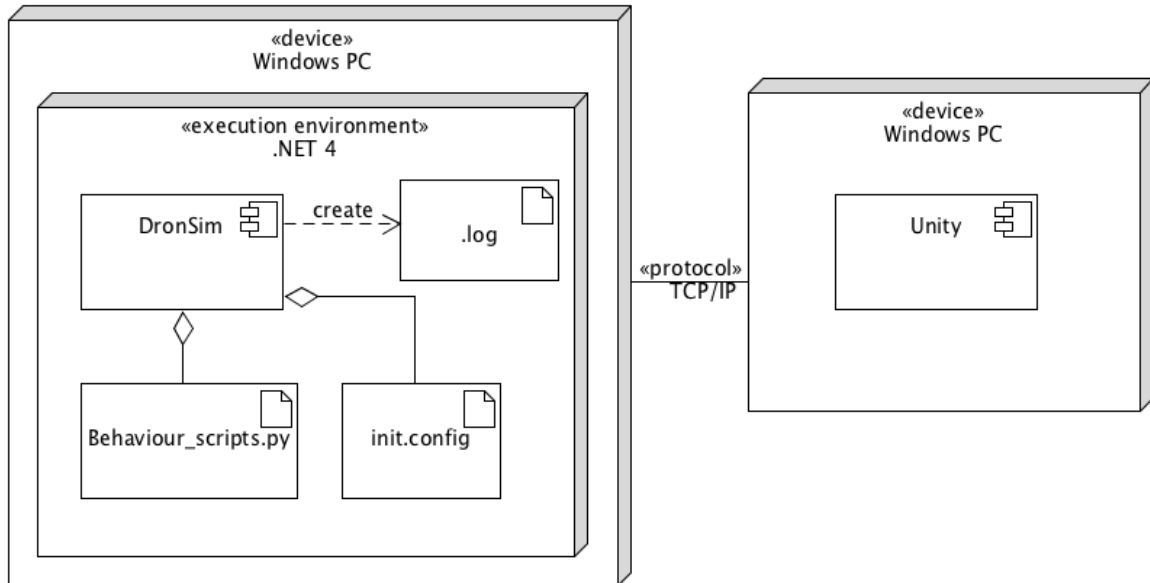
Nami navrhnutá architektúra aplikácie je použiteľná v troch rôznych rozloženiach nasadenia komponentov.

Prvý diagram nasadenia na obrázku 5 popisuje situáciu, kedy sú oba komponenty riešenia nasadené na jednom stroji s operačným systémom Windows. Pre spustenie samotného simulátora je potrebná aj inštalácia programového rámca .NET Framework 4.5.2. Časti riešenia potom komunikujú pomocou protokolu TCP/IP.



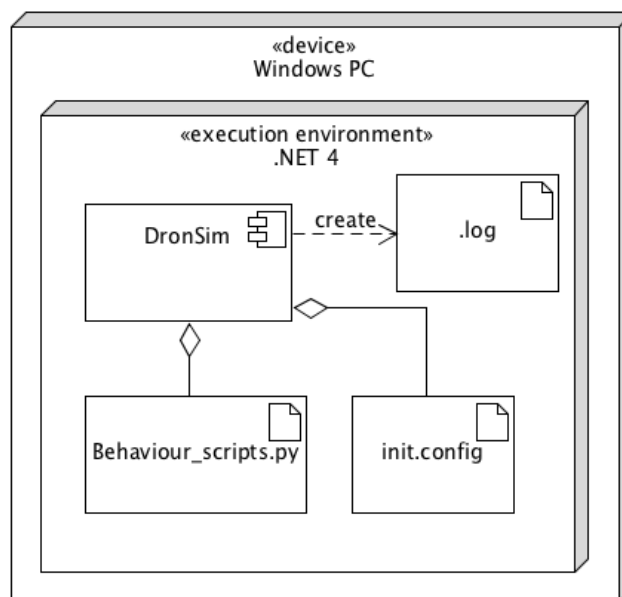
Obr. 5: Diagram nasadenia prvého prípadu

Diagram na obrázku 6. popisuje nasadenie aplikácie na dvoch samostatných zariadeniach s operačným systémom Windows. Tak ako v prvom prípade, na zariadení so simulátorom je pre spustenie potrebný programový rámec .NET Framework 4.5.2.



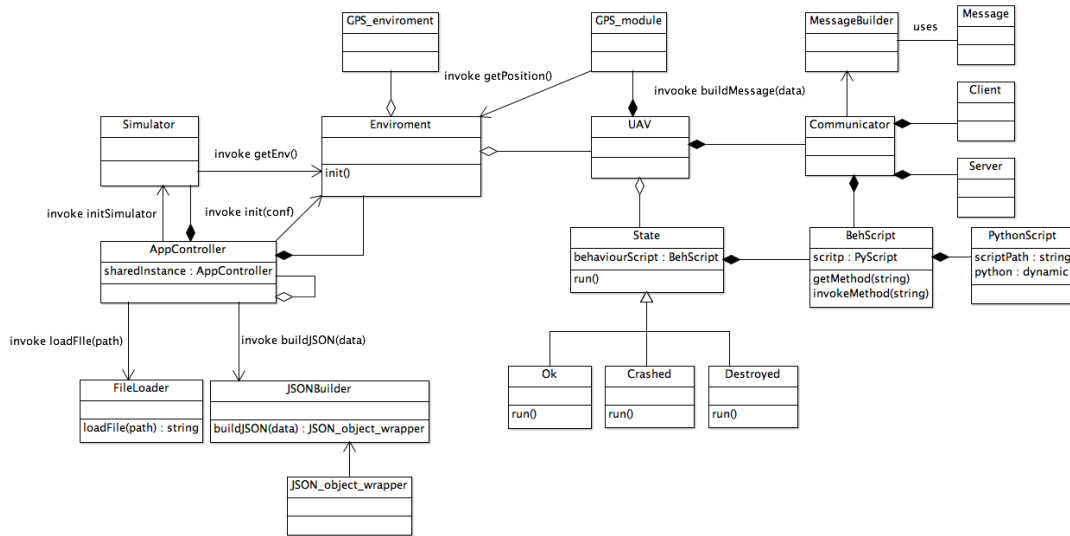
Obr. 6: Diagram nasadenia druhého prípadu

Obrázok 7 opisuje nasadenie aplikácie na jednom zariadení s operačným systémom Windows a programovým rámcom .NET Framework 4 bez nasadeného vizualizátora Unity 3D. V tomto nasadení sa zapisujú len záznamy o testovaní vo forme súborov *názov.log*.



Obr. 7: Diagram nasadenia tretieho prípadu

## Diagram tried



Obr. 8: Návrh diagramu tried novej implementácie

Diagram tried opisuje štruktúru tried novej implementácie aplikácie.

### AppController

Trieda určená na ovládanie toku aplikácie. Načítava konfiguračné súbory a na základe zadaných informácií inicializuje moduly Simulator a Environment. Tiež spúšťa a ovláda simuláciu.

### Simulator

Stará sa o samotnú simuláciu dronov. Aplikuje pohyb a vplyvy prostredia, s ktorými bolo prostredie Environment zinicilizované.

### Environment

Predstavuje prostredie, v ktorom budú drony simulované. Obsahuje informácie o dronoch, prekážkach na mape a mape samotnej.

### UAV

Objekt predstavuje dron podľa používateľskej konfigurácie z konfiguračného súboru, ktorý obsahuje informácie o rozmeroch, tvare a ostatných vlastnostiach dronov.

### **State**

Reprezentuje stav drona, ktorý bude obsahovať správanie drona v jednotlivých stavoch, definovaných v Python skriptoch.

### **Communicator**

Implementuje metódy, ktoré slúžia na komunikáciu dronov s prostredím Environment a prostredia s vizualizátorom Unity 3D.

### **JSON\_object\_wrapper**

Obaľuje JSON objekt a implementuje nad ním metódy na prístup k jeho reprezentácia.

### **GPS\_enviroment**

Modul pre počítanie GPS pozícií na základe pozícií dronov v prostredí a poskytovania pozícií metódami.

### **BehScript**

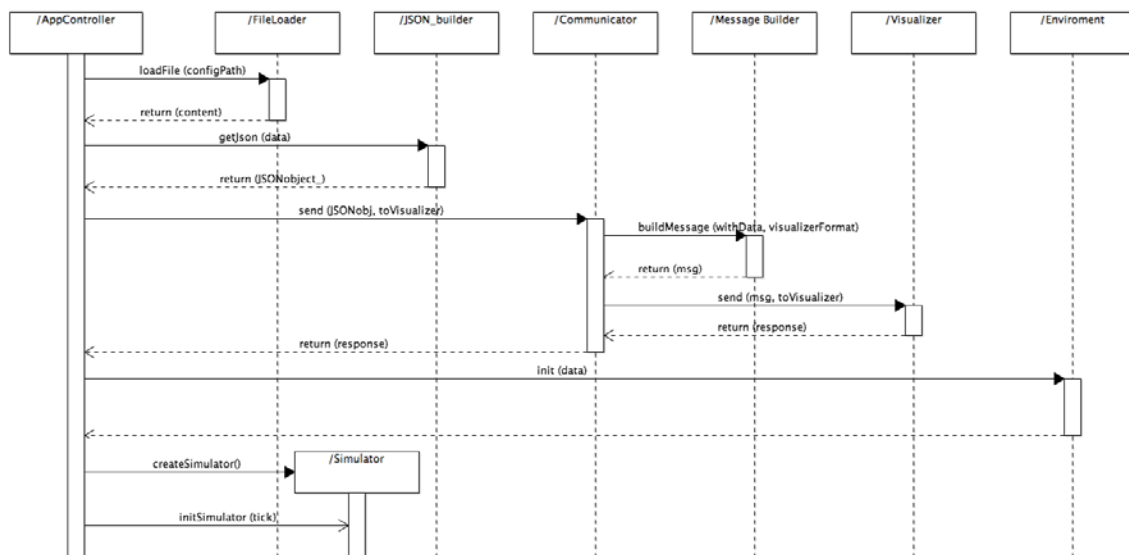
Reprezentuje triedu obsahujúcu Python skripty s opisom správania drona, komunikácie dronov a definícií senzorov dronov. Tiež implementuje metódy na volanie dynamických metód z Python skriptu.

## Sekvenčné diagramy

V tejto kapitole popíšeme navrhnuté sekvenčné diagramy kľúčových funkcií navrhutej aplikácie.

### Inicializácia aplikácie

Obrázok 9 reprezentuje návrh volania metód pri inicializácii prostredia simulátora v novej implementácii projektu. Zahrňuje inicializáciu vizualizátora Unity 3D na základe konfiguračného súboru. Tiež sa tu inicializuje prostredie Environment dronmi a ostatnými parametrami z konfiguračného súboru. Ďalším krokom inicializácie je vytvorenie modulu simulátora Simulator a jeho nastavenie pre potreby scenára v konfiguračnom súbore.



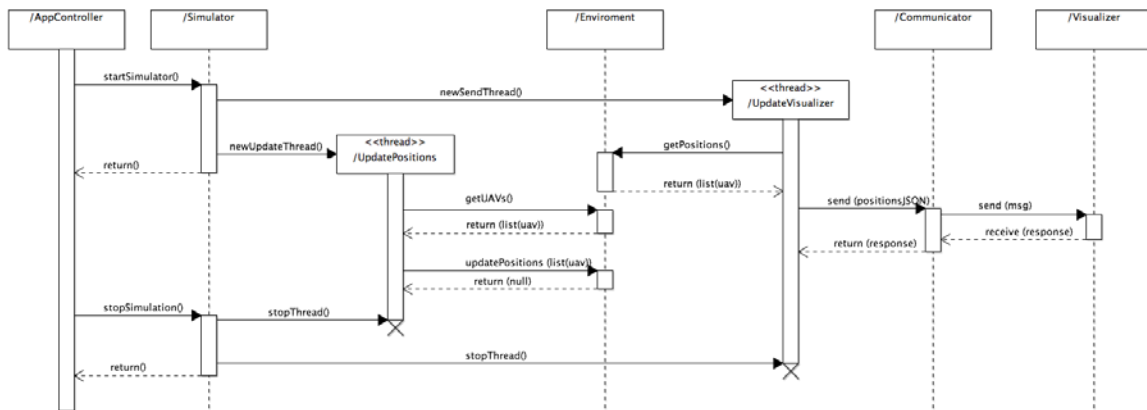
Obr. 9: Sekvenčný diagram inicializácie aplikácie

### Obnovovanie polohy

Na obrázku 10 je sekvenčný diagram návrhu volaní metód v procese simulácie. Obsahuje dve základné časti so samostatnými vláknami, ktoré sa spustia pri spustení simulácie.

Jedným je vlákno na komunikáciu s vizualizátorom Unity 3D, ktoré v určených intervaloch zasiela informácie o polohách dronov a ich zmene.

Ďalším je vlákno pre výpočet pohybu dronov v prostredí na základe scenárov a správania definovaného v skriptoch správania dronov. Zmenené pozície sú uložené v prostredí Environment.



Obr. 10: Sekvenčný diagram obnovovania polohy

## Návrh komunikácie medzi prostredím a vizualizátorom

Podľa nových požiadaviek na komunikáciu systému, v ktorých je definované, že má byť komunikácia rozšírená na komunikáciu po lokálnej sieti, je nutné prerobiť koncept pôvodne navrhutej komunikácie. Momentálne je komunikácia zabezpečená pomocou UDP protokolu na rovnakom stroji (localhost). Keďže má byť komunikácia rozšírená na lokálnu sieť, je potrebné zmeniť protokol komunikácie na TCP.

Pôvodný návrh správ, ktoré si medzi sebou vizualizátor a prostredie vymieňajú, je v dobrom stave a môže byť ponechaný v rovnakej podobe. Formát správ zostáva aj naďalej JSON.

## Návrh vizualizátora

Vizualizátor vypracovaný bývalým tímom je postačujúci a momentálne nie je potrebné ho meniť. Doposiaľ neboli zadané žiadne ďalšie požiadavky pre vizualizátor.



# 11 Implementácia

Aktuálna verzia našej implementácie ponúka dve možnosti simulácie. Prvou možnosťou je simulácia bez grafického rozhrania, pri ktorej je možné jej priebeh sledovať prostredníctvom vytváraných záznamov v textovej forme (tzv. logov).

```
*****
CURRENT TIME: 08.11.2016 09:08:16
CLOCK TICK: 4
CURRENT WIND CONDITIONS: 2, 0, 2
*****

-----
ID | STATUS | POSITION | ENERGY LEFT |
-----
0 | Ok | 10,3, 3, 15,3 | 100 |
*****

*****
CURRENT TIME: 08.11.2016 09:08:16
MESSAGE
FROM: 0
TO: 0
TO REAL: 0
BODY: <<module>.Message instance at 0x000000000000002B>
COMMUNICATION TYPE: 1
*****

*****
CURRENT TIME: 08.11.2016 09:08:17
CLOCK TICK: 5
CURRENT WIND CONDITIONS: 2, 0, 2
*****

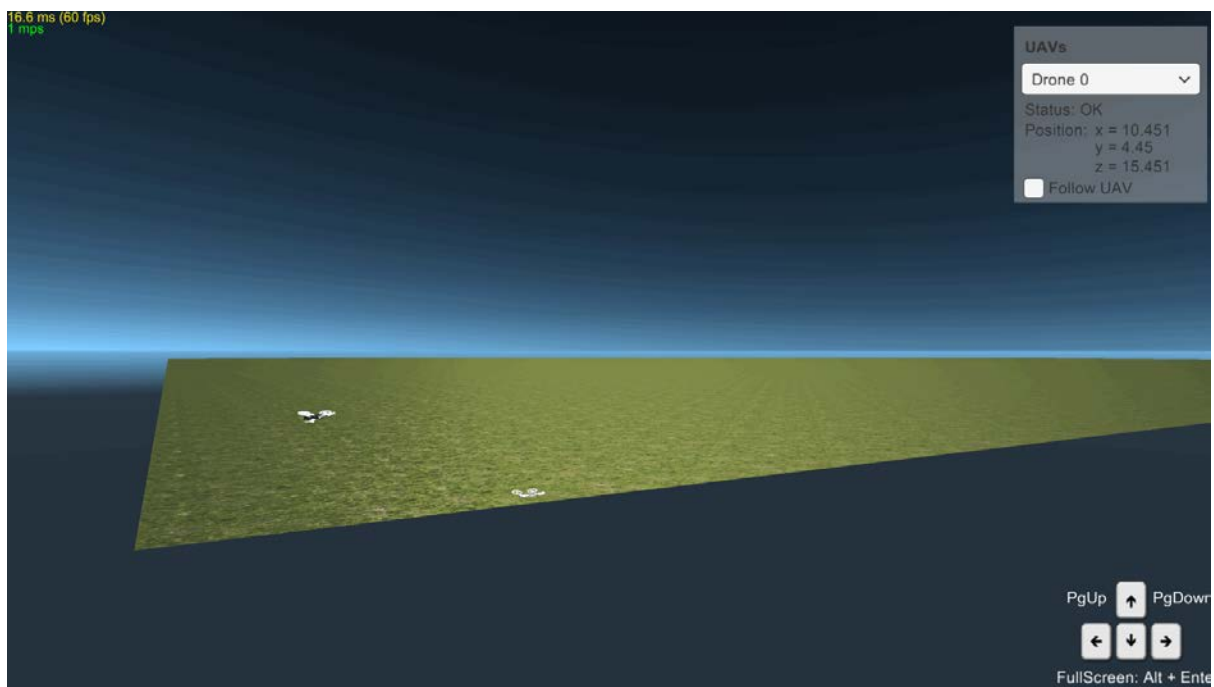
-----
ID | STATUS | POSITION | ENERGY LEFT |
-----
0 | Ok | 10,43, 3,22, 15,38 | 100 |
*****

*****
CURRENT TIME: 08.11.2016 09:08:18
CLOCK TICK: 6
```

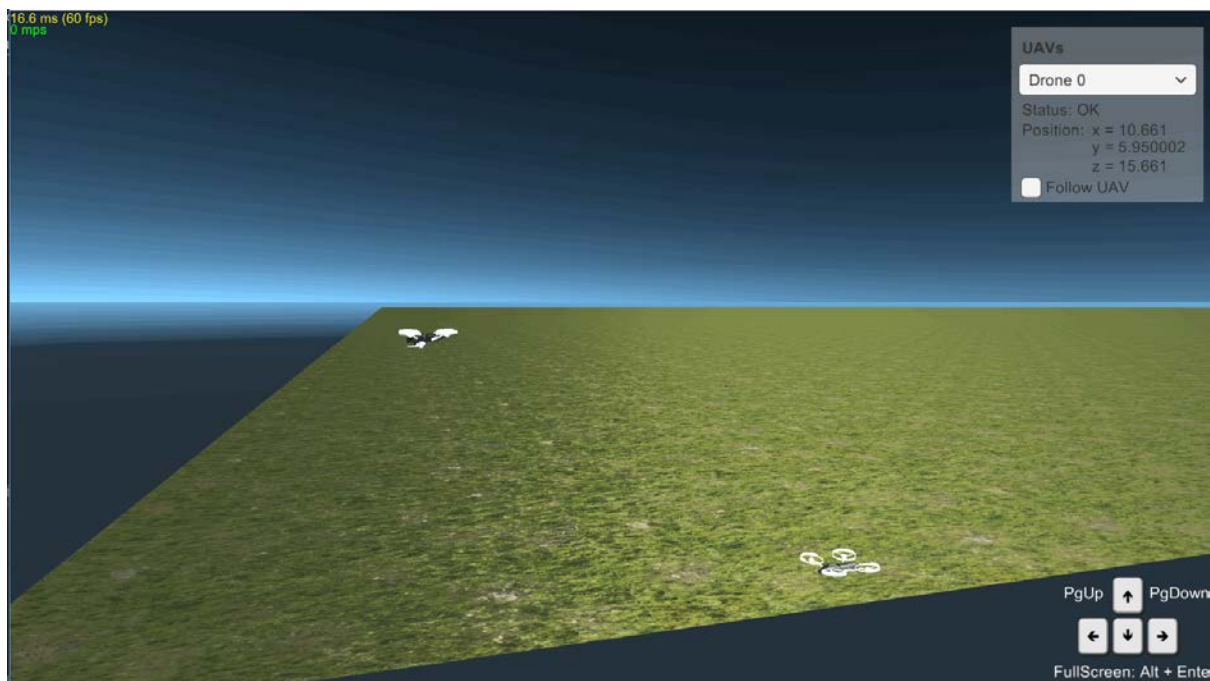
Obr. 11: Záznam vygenerovaný zo spustenej simulácie

Druhou možnosťou je zobrazenie s konfigurovateľným grafickým rozhraním. Používateľ v rámci neho môže nastaviť rozlíšenie, úroveň detailov, či zobrazenie v režime na celú obrazovku, resp. v okne. Vyššie spomenuté súbory so záznamami sú vytvárané aj počas spusteného grafického zobrazenia.

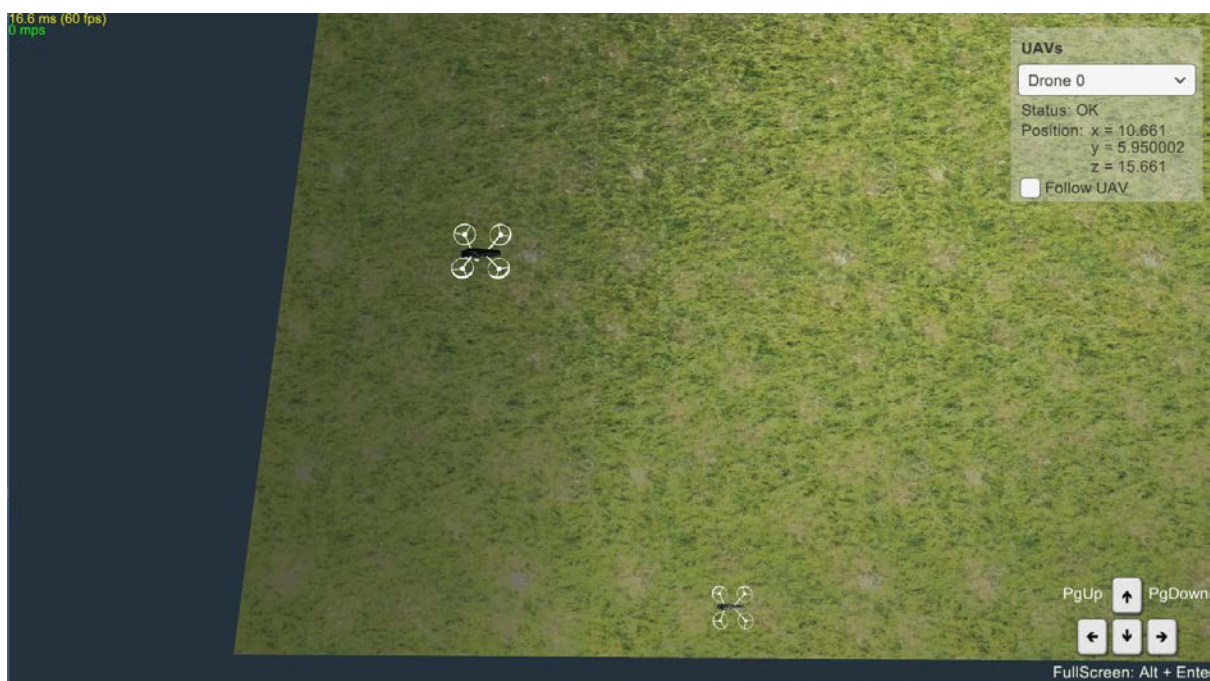
Samotné grafické rozhranie a vykresľovanie simulácie je implementované v nástroji Unity 3D. Keďže samotná logika simulácie prebieha v C# konzole, bolo potrebné prepojiť tieto dva komponenty. Komponenty sú prepojené TCP soketmi a vymieňajú si medzi sebou údaje. Pre tento účel sme sa rozhodli využiť JSON súbory, ktoré budú v sebe obsahovať potrebné informácie. Na začiatku existuje jeden hlavný konfiguračný JSON súbor, na základe ktorého sa nastaví počiatkové údaje. Ide konkrétne o veľkosť mapy, počet dronov a ich vlastnosti v podobe počiatkovej pozície, rozmerov a senzorov. Taktiež tento súbor obsahuje počet, veľkosť a pozície jednotlivých objektov nachádzajúcich sa na mape. Počas behu simulácie prostredie pravidelne oznamuje vizualizátoru aktuálne stavy a pozície dronov pomocou JSON súborov.



Obr. 12: Grafické rozhranie (vzdialený pohľad)



Obr. 13: Grafické rozhranie (približený pohľad)



Obr. 14: Grafické rozhranie (horný pohľad)

## 12 Testovanie

V rámci testovania uplatňujeme testovanie pomocou unit testov. Tieto testy sme sa rozhodli vykonávať pomocou NUNIT frameworku, ktorého adaptér sme implementovali do Visual Studia ako rozšírenie. Princíp písania testov je zadaný v metodike testovania.

# **Prílohy**

## **A Metodiky**

## Metodika dokumentácie

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

## Obsah

1. Úvod .....	3
2. Metodika generovania dokumentácie.....	4
3. Metodika tvorby dokumentácie.....	6
3.1. Pravidlá formy dokumentácie .....	6

## 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.

Použitie 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.

Použitie skratky je nutné vysvetliť v kapitole „Použitie skratky“. Použitie 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 vzhľad a štruktúru dokumentácie.

## Metodika písania kódu

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

## Obsah

1. Úvod .....	3
1.1. Formátovanie kódu .....	3
1.2. Štandard kódu .....	3
2. Metodika pomenovaní (naming conventions) .....	4
2.1. C# .....	4
2.2. Python .....	4
3. Metodika komentárov .....	5

## 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 píše 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
Inšancie 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, viacriadkový 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	<code>//</code>
Viacriadkové	<code>/* ..... */</code>
Nad premenné a metódy	<code>///</code>

### Python

Komentár	Notácia
Jednoriadkové	<code>#</code>
Nad premenné a metódy	<code>""" ..... """</code>



## Metodika pre komunikáciu

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

# LEGENDRONE

## Obsah

1. Úvod .....	3
2. Prehľad komunikačných kanálov.....	4
3. Komunikačné nástroje.....	5

## 1. Úvod

Účelom tejto metodiky je opísať a zdefinovať 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

Naším 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 prítomní. 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,

# LEGENDRONE

- 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

## Metodika pre code review a verziovanie

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



## Obsah

1. Úvod .....	3
1.1. Roly vývojárov .....	3
1.2. Slovník pojmov .....	3
2. Pravidlá code review .....	4
2.1. Pravidlá.....	4
2.2. Sumarizácia .....	4
3. Ako požiadať o code review .....	5
4. Od vytvorenia tasku k mergnutiu do projektu .....	6
5. Práca s branchami v TFS.....	7
5.1. Vytvorenie novej branche.....	7
5.2. Vytvorenie pull requestu .....	7

## 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 implementácie 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 zachovávajú 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 merge 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.

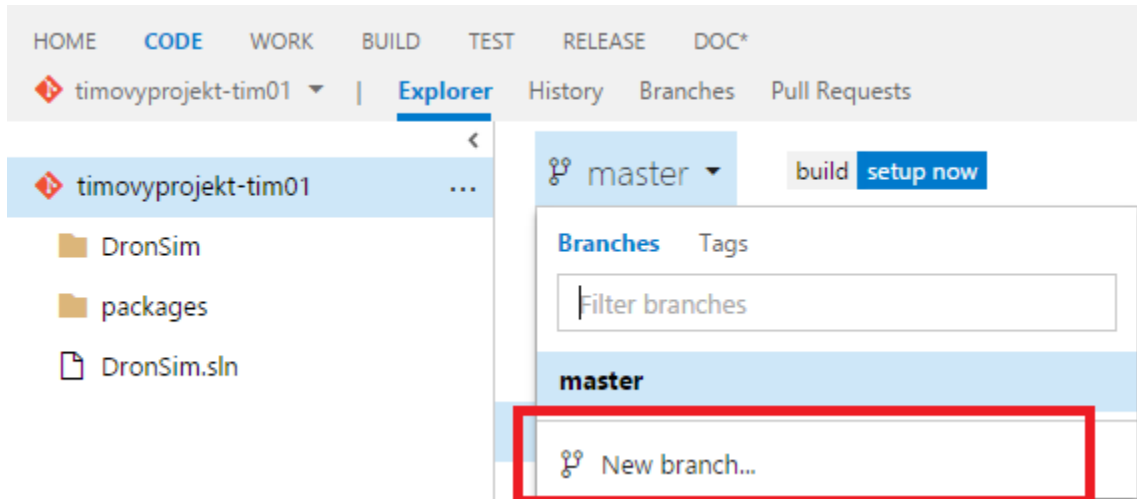
Tento postup bude vyzeráť nasledovne:

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 branchu „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 sústreďí na to, či daná implementácia spĺňa požiadavky úlohy, taktiež či je kvalita kódu dostatočná, či je kód riadne zdokumentovaný – vid’ 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 časťami 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 merge do branchu „master“

## 5.Práca s branchami v TFS

### 5.1. Vytvorenie novej branchy

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



- 2.
3. Vyplniť kolonku "Name" pre meno branchy, zvoliť, z ktorej branchy sa má nová brancha vytvoriť

✕

### Create a branch

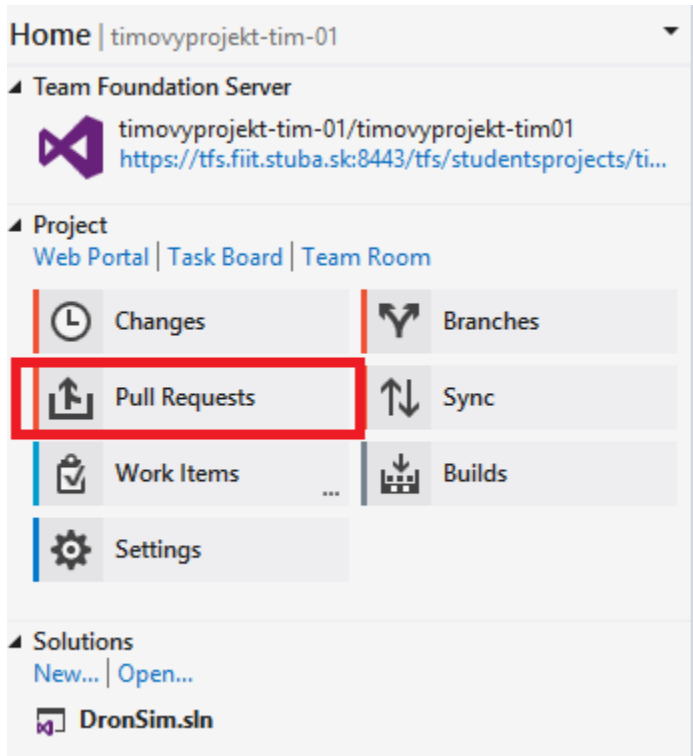
Name

Based on

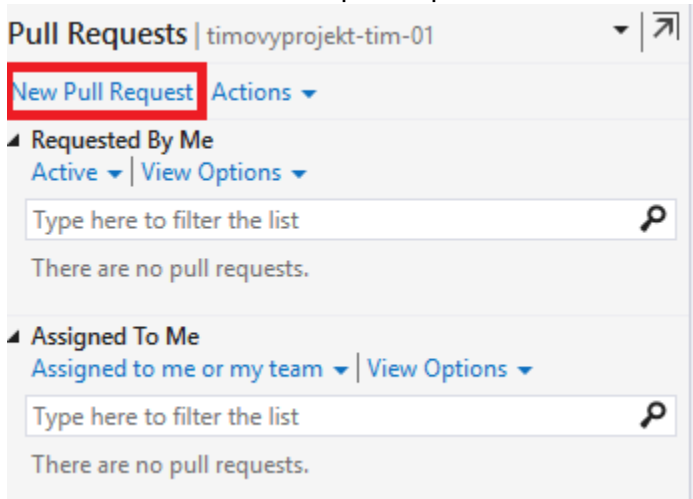
- 4.

### 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 nie sú pridané súvisiace Work itemy, pridať aj tie, následne kliknúť na New pull request

# LEGENDRONE

Review changes in  master-testbranch ▾ relative to  master ▾ ↔

Testovací pull request

**DESCRIPTION**

- pridany nový komentár

**REVIEWERS**

Search users and groups

**RELATED WORK ITEMS**

Add work items

[New pull request](#) [fewer options](#)

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



## Metodika testovania

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

## Obsah

1. Úvod .....	3
2. Súvisiace metodiky a manuály .....	4
3. Súvisiace metodiky a manuály .....	5
3.1. Testovanie kódu .....	5
3.2. Vizuálne testovanie .....	6

## 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, prekryvanie objektov, atď.

## Metodika manažmentu úloh

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

## Obsah

1. Úvod.....	3
2. Práca s úlohami v TFS .....	4
2.1. Vytvorenie novej úlohy .....	4
2.2. Zmazanie úlohy .....	5
2.3. Správa úloh .....	6
2.3.1 Presunutie úlohy do ďalšieho šprintu .....	6
2.3.2 Priradenie úlohy členovi tímu .....	6



## 1. Úvod

Účelom tejto metodiky je opísať a zdefinovať 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

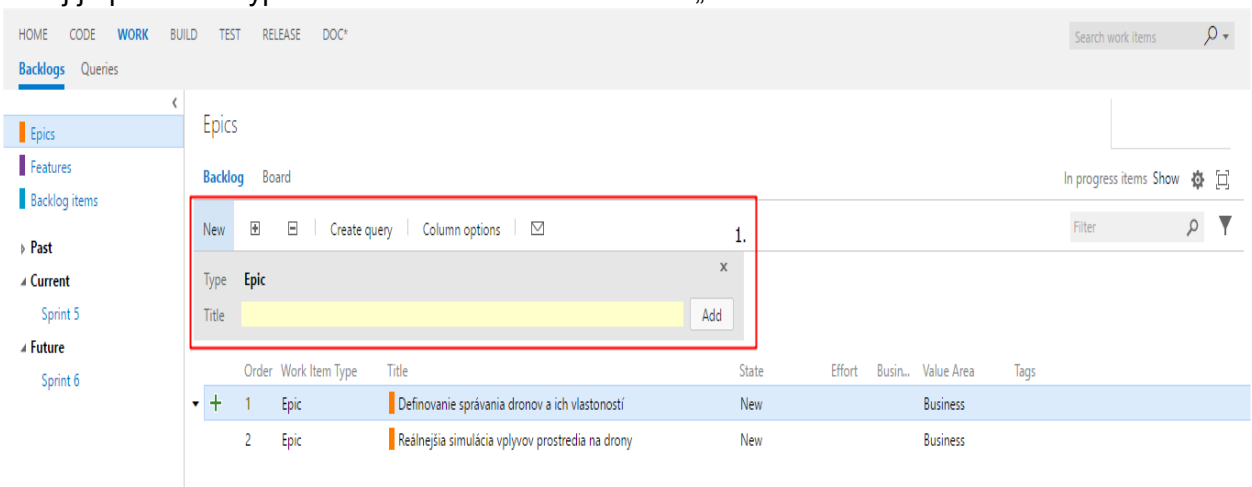
V tejto sekcii sú opísané postupy pre vytváranie, mazanie, upravovanie, popisovanie, správu úloh v TFS.

### 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“.

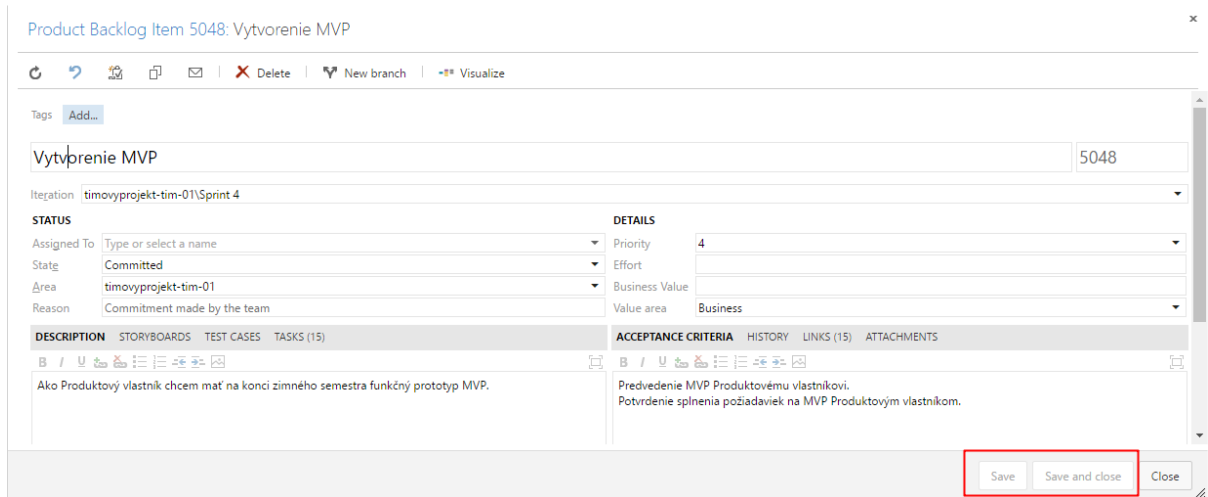


The screenshot shows the TFS Backlogs interface. On the left, there is a navigation pane with 'Epics' selected. The main area shows a 'Backlog Board' with a table of work items. A 'New' dialog box is open, allowing the user to create a new work item. The dialog box has a 'Type' field set to 'Epic' and a 'Title' field with a yellow highlight. An 'Add' button is visible next to the title field. Below the dialog box, a table lists existing work items:

Order	Work Item Type	Title	State	Effort	Busin..	Value Area	Tags
1	Epic	Definovanie správaní dronov a ich vlastností	New			Business	
2	Epic	Reálnejšia simulácia vplyvov prostredia na drony	New			Business	

3. Po vytvorení úlohy je potrebné zdefinovať 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.

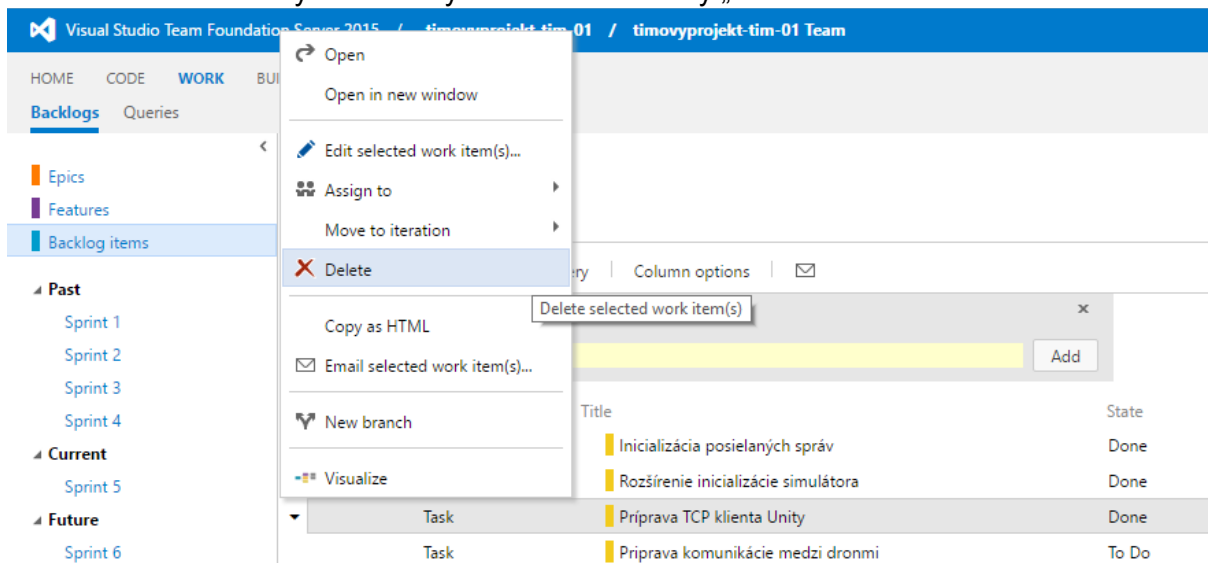
Po týchto krokoch je úloha vytvorená a učená na vykonávanie.



## 2.2. Zmazanie úlohy

Pri situácii , keď je potrebné zmazať úlohu je potrebné túto zmenu konzultovať s produktovým vlastníkom , scrum masstrom 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“.

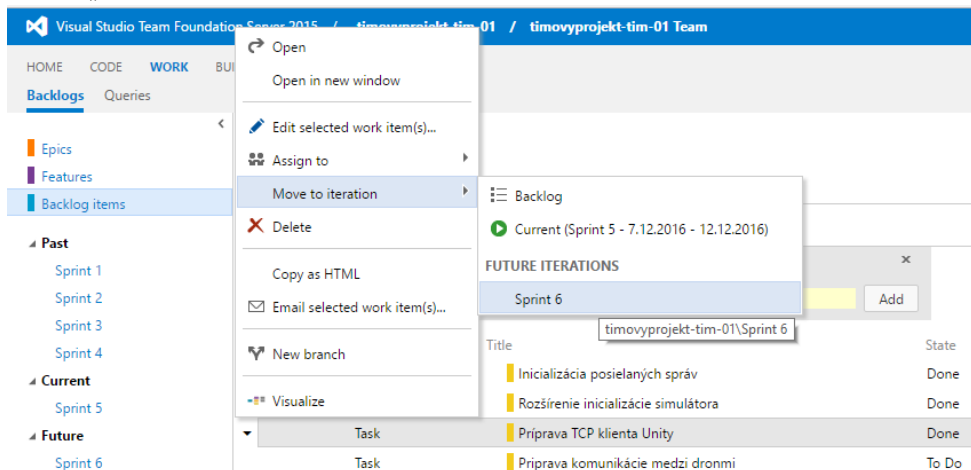


## 2.3. Správa úloh

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.

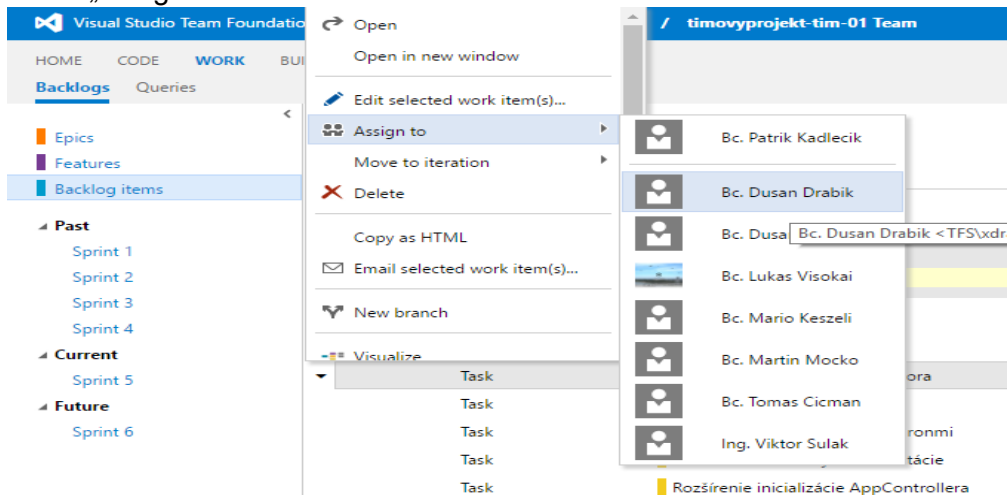
### 2.3.1 Presunutie úlohy do ďalšieho šprintu

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
4449	Product Backlog Item	Zadefinovanie správania senzorov dronov			New	Ako používateľ chcem mať možnosť zadefinovanie nového senzora pre drony.	2
4450	Product Backlog Item	Zadefinovanie správania drona			Done	Ako používateľ chcem mať možnosť pevne definovať spôsob správania drona v simulácii.	2
4779	Task		Inicializácia Enviromentu	Bc. Tomas Cicman	Done	V rámci úlohy je potrebné zanalyzovať a komentovať zdrojový kód časti Enviroment.	2
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
4782	Task		Analýza komunikácie - Python	Bc. Dusan Drabik	Done	V rámci úlohy je potrebné zanalyzovať a komentovať komunikáciu medzi časťou Enviroment a Simulátor.	2
4783	Task		Finalizácia metodík tímu	Bc. Martin Mocko	Done	V rámci úlohy je potrebná finalizácia metodík v tíme , konkrétne finalizácia dokumentov k metodike komunikácie , prehliadok kódu , správy verzií , kódových konvencií.	2
4848	Bug		Nehýbanie sa drona v UNITY - neposielanie správ do UNITY	Bc. Lukas Visokai	Done		2
4455	Product Backlog Item	Definovanie prostredia			New	Ako používateľ chcem mať možnosť definovania prostredia simulácie(definovanie mapy , objektov v simulácii, poveternostné podmienky).	2
4456	Product Backlog Item	Reálnejšie kolízie			New	Ako používateľ chcem ovplyvňovanie správania dronov v simulácii pomocou reálnejších kolízií s objektami v prostredí.	2
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

4778	Task		Príprava TCP komunikácie na localhost - UNITY modul	Bc. Dusan Hatvani	Done	V rámci úlohy je potrebné pripraviť projekt na realizáciu TCP komunikácie.	2
4458	Product Backlog Item	Tímová réžia			Done	Ako Produktový vlastník chcem mať vytvorený tímový web , vytvorený komunikačný kanál pre tím.	2
4459	Task		Metodika písania kódu	Bc. Lukas Visokai	Done	V rámci úlohy je potrebné vytvoriť dokument , ktorý popisuje metodiku písania kódu v tíme.	3
4460	Task		Vytvorenie webovej stránky	Bc. Patrik Kadlecik	Done	V rámci úlohy je potrebné vytvoriť webovú stránku tímu, ktorá je dostupná na: <a href="http://labss2.fiiit.stuba.sk/TeamProject/2016/team01is-si/">http://labss2.fiiit.stuba.sk/TeamProject/2016/team01is-si/</a>	2
4461	Task		Metodika generovania dokumentácie	Bc. Mario Keszeli	Done	V rámci úlohy je potrebné vytvoriť dokument , ktorý popisuje metodiku generovania dokumentácie.	3
4462	Task		Refaktoring zdrojového kódu Environmentu	Bc. Dusan Drabik	Done	V rámci úlohy je potrebné vykonať refaktoring zdrojového kódu z časti projektu Enviroment .	4
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: <a href="http://labss2.fiiit.stuba.sk/TeamProject/2016/team01is-si/">http://labss2.fiiit.stuba.sk/TeamProject/2016/team01is-si/</a>	4
4464	Task		Refaktoring zdrojového kódu vizualizátora Unity	Bc. Dusan Hatvani	Done	V rámci úlohy je potrebné zrefaktorovať zdrojový kód časti vizualizátora Unity.	4
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
4653	Task		Integrácia TFS a Unity	Bc. Lukas Visokai	Done	V rámci úlohy je potrebné vyhľadať možnosti integrácie zdrojového kódu Unity a TFS.	4
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 využití senzorov v dronoch v rôznych prostrediach.	4

4680	Task	Rešerš použitia dronov	Bc. Lukas Visokai	Done	V rámci úlohy je potrebné vyhľadať možnosti a konvencie používania UAV v reálnom živote. Je potrebné vytvoriť dokument zahrňujúci bežné využitie UAV v praxi.	4
4857	Task	Migrácia TFS projektu na GIT projekt		Done	V rámci úlohy je potrebné migrovať aktuálny TFS projekt na GIT projekt.	2
4903	Task	Kontinuálna integrácia	Bc. Martin Mocko	Done	V rámci úlohy je potrebné vyriešiť kontinuálnu integráciu v TFS.	4
4897	Product Backlog Item	Vytváranie špecifikácie z požiadaviek		Done	Ako Produktový vlastník chcem vedieť , či moje požiadavky na vytváraný systém boli správne pochopené.	2
4898	Task	Návrh formátu Logov	Bc. Mario Keszeli	Done	V rámci úlohy je potrebné navrhnúť formát správ generovaných systémom (Logov).	1
4899	Task	Návrh SDK pre Python scripty	Bc. Dusan Hatvani	Done	V rámci úlohy je potrebné vytvoriť návrh SDK pre Python scripty.	2
4900	Task	Návrh diagramov systému	Bc. Dusan Drabik	Done	V rámci úlohy je potrebné navrhnúť diagram komponentov , diagram tried a sekvenčné diagram systému.	4
4901	Task	Návrh formátu správ posielaných pri komunikácii medzi dronmi	Bc. Lukas Visokai	Done	V rámci úlohy je potrebné navrhnúť formát správ , ktoré sa budú posilať medzi dronmi pri ich komunikácii.	3
4902	Task	Návrh spôsobu simulácie	Bc. Lukas Visokai	Done	V rámci úlohy je potrebné navrhnúť spôsob simulácie dronov v prostredí simulátora.	2
4904	Task	USE CASE diagram	Bc. Patrik Kadlecik	Done	V rámci úlohy je potrebné vytvoriť use case diagram použitia dronov.	1
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, navrhnúť objekty v prostredí v simulátore.	2
4906	Task	Metodika testovania	Bc. Tomas Cicman	Done	V rámci je potrebné vytvoriť dokument , ktorý popisuje metodiku testovania v tíme.	1
4907	Task	Návrh formátu správ medzi Enviromentom a Unity	Bc. Dusan Hatvani	Done	V rámci úlohy je potrebné navrhnúť formát správ posielaných medzi Enviromentom a Unity.	1



4908	Task	Dokumentácia 1. kontrolného bodu	Bc. Mario Keszeli	Done	V rámci úlohy je potrebné dokončiť dokumentáciu k 1. kontrolnému bodu.	3
4985	Product Backlog Item	Komunikácia medzi dronmi		New	Ako používateľ chcem komunikáciu medzi dronmi v roji pri simulácii v prostredí.	2
5048	Product Backlog Item	Vytvorenie MVP		Committed	Ako Produktový vlastník chcem mať na konci zimného semestra funkčný prototyp MVP.	4
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 atribútmi a ich vzťahy.	2
5050	Task	Vytvorenie sekvenčných diagramov	Bc. Dusan Drabik	Done	V rámci úlohy je potrebné vytvoriť sekvenčné diagramy a identifikovať základné správy, ktoré sa budú posielat medzi časťami systému. Je potrebné vytvoriť sekvenčný diagram na inicializáciu aplikácie a obnovovanie polohy UAV.	2
5051	Task	Inicializácia Enviromentu	Bc. Tomas Cicman	Done	V rámci úlohy je potrebné implementovať základnú funkcionality Enviromentu a pripojiť časť Enviromentu s časťou ApplicationController.	2
5052	Task	Inicializácia Simulátora	Bc. Lukas Visokai	Done	V rámci je potrebné implementovať základnú funkcionality simulátora, teda aktualizácia polohy UAV.	2
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
5054	Task	Vytvorenie komunikátora	Bc. Dusan Hatvani	Done	V rámci úlohy je potrebné pripraviť potrebné triedy a atribúty na implementovanie základného komunikátora medzi simulátorom a vizualizátorom.	2
5055	Task	Inicializácia aplikácie (AppController)	Bc. Patrik Kadlecik	Done	V rámci úlohy je potrebné inicializovať spustenie aplikácie. V triede ApplicationController je potrebné rozdeliť hlavný konfiguračný skript na dva , kde v prvom bude mapa reprezentovaná JSON-om. V druhom bude pole jednotlivých UAV.	2
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 prijímanie a posielanie správ z Enviromentu k UAV a naopak.	2

5185	Task	Rozšírenie inicializácie simulátora	Bc. Lukas Visokai	Done	V rámci úlohy je potrebné rozšíriť funkcionality simulátora doplnením základných atribútov a metód na pôsobenie vetra na UAV. Prepojenie simulátora s vizualizátorom Unity .	2
5186	Task	Príprava TCP klienta Unity	Bc. Dusan Hatvani	Done	V rámci úlohy je potrebné pripraviť základnú kostru TCP klienta na strane vizualizátora.	2
5187	Task	Príprava komunikácie medzi dronmi	Bc. Dusan Drabik	Done	V rámci úlohy je potrebné pripraviť základnú štruktúru komunikácie medzi dronmi.	2
5188	Task	Dokončenie finálnej dokumentácie	Bc. Mario Keszeli	Done	Je potrebné doplniť dokumentáciu o požadované informácie a pripraviť ju na finálne odovzdanie.	2
5189	Task	Rozšírenie inicializácie AppControllera	Bc. Patrik Kadlecik	Done	V úlohe je potrebné rozšíriť funkcionality AppControllera a vytvoriť príslušné testy.	2
5190	Task	Rozšírenie inicializácie Enviromentu	Bc. Tomas Cicman	Done	V úlohe je potrebné rozšíriť funkcionality Enviromentu. Je potrebné spracovať JSON objekt reprezentujúci mapu a objekty v prostredí a tiež spracovať pole UAV objektov.	2
5191	Task	Posielanie správ	Bc. Martin Mocko	Done	V rámci úlohy je potrebné rozšíriť funkcionality posielania a prijímania správ v triede Message.	2