SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

Fakulta informatiky a informačných technológií

Ilkovičova 2, 842 16 Bratislava 4

Tímový projekt
RoboCup
Dokumentácia k riadeniu projektu

Bc. Filip Blanárik

Bc. Michal Blanárik

Bc. Štefan Horváth

Bc. Štefan Linner

Bc. Martin Markech

Bc. Roman Moravčík

Bc. Tomáš Nemeček

Tím č. 9:	Gitmen
Vedúci projektu:	Ing. Ivan Kapustík
Predmet:	Tímový projekt I
Ročník:	1
Akademický rok:	2013/2014, zimný semester
Mailový kontakt:	gitmen09@gmail.com

Obsah

1 Úvod

Tento dokument obsahuje dokumentáciu k riadeniu projektu: RoboCup , ktorý je vypracovávaný v rámci predmetu Tímový projekt.

Súčasťou dokumentu je aj ponuka v kapitole 2, ktorá bola vypracovaná pri uchádzaní sa o projekt.

Rozdelenie manažérskych úloh medzi jednotlivých členov tímu sa nachádza v kapitole číslo 3 Úlohy členov tímu.

Úlohou tohto dokumentu je vymedzenie postupov riešenia úloh v tíme pomocou zadefinovaných metodík v kapitolách 5 až 12.

Práca na projekte je rozdelená na šprinty, ktorých dĺžka je stanovená na dva týždne pričom oficiálne stretnutie sa koná raz týždenne. Na oficiálnych stretnutiach sa prezentuje priebežný postup práce a stanovujú sa ciele na nasledujúce obdobie. Výstupom každého oficiálneho stretnutia je zápisnica ktorá sa nachádza v kapitole: Príloha A Záznamy zo stretnutí.

2 Zoznam kompetencií tímu

V kapitole sa nachádzajú informácie k zručnostiam a znalostiam jednotlivých členov tímu a dokumenty tikajúce sa vytvorenej ponuky na danú tému.

2.1 Predstavenie členov tímu

Bc. Blanárik Filip

Absolvent bakalárskeho študijného programu Informatika na FIIT STU. Obhájil bakalársku prácu s názvom "Modelom riadená evolúcia pravidlového systému". Počas štúdia nadobudol skúsenosti s jazykmi C, Java, PHP/MySQL, Asembler.

Bc. Blanárik Michal

Absolvent bakalárskeho štúdia na FIIT STU. Obhájil bakalársku prácu z oblasti vyhľadávania a spracovania informácií. Ovláda jazyky C, Java, Assembler, PHP/MySQL, HTML. Počas štúdia a riešenia bakalárskej práce a iných projektov nadobudol skúsenosti s knižnicami OpenCV, GATE, Apache OpenNLP, MPI a ďalšie.

Bc. Horváth Štefan

Absolvent bakalárskeho štúdia na Fakulte Matematiky, Fyziky a Informatiky UK. Počas štúdia získal znalosti z jazykov Pascal, C++, HTML, CSS, PHP/MySQL, Oracle a Java. Má skúsenosti z vývojom automatizovaného testovacieho nástroja pre web.aplikácie na platforme .NET a tvorbou šablón v softvéri Stimulsoft. Svoje štúdium zameriava hlavne na geometriu a počítačovú grafiku v spojení C++/OpenGL/GLSL.

Bc. Linner Štefan

Absolvoval štúdium na FIIT STU obhájením bakalárskej práce z oblasti umelej inteligencie. Ovláda jazyky C/C++, Java. Má skúsenosti s vedením tímu konzultantov aplikačnej podpory platobného informačného systému ako aj s pracovaním v malom tíme na tvorbu webových stránok ako PHP/MySQL programátor a HTML/CSS/JQuery kóder. Ovláda prácu s grafickým editorom Adobe Photoshop na pokročilej úrovni.

Bc. Markech Martin

Absolvent bakalárskeho študijného programu Informatika na FIIT STU. Počas štúdia získal vedomosti z oblasti webových technológií, umelej inteligencie i paralelného programovania. Ovláda jazyky a technológie ako Asembler, C, Java, PHP, Ruby, Ruby on Rails, HTML, XSLT, CSS, MySQL. Má skúsenosti s tvorbou webových systémov a pracuje na pozícii Ruby on Rails programátora, kde má na starosti okrem rozširovania funkcionality aplikácie i správu servera. Pri práci získal skúsenosti s verziovacím nástrojom Git a podpornými nástrojmi.

Bc. Moravčík Roman

Absolvent bakalárskeho študijného programu Informatika na FIIT STU. Obhájil bakalársku prácu s názvom Brankár pre simulovaný robotický futbal, počas riešenia ktorej nadobudol znalosti z umelej inteligencie v robotike. Počas štúdia získal vedomosti z oblasti algoritmov a návrhu softvéru. Ovláda C/C++, Java, Objective C. V práci získal praktické skúsenosti s návrhom architektúry softvéru, jej implementácie ako aj s testovaním a s koordináciou ľudí na úlohach pri projekte.

Bc. Nemeček Tomáš

Absolvoval bakalárske štúdium na Fakulte informatiky a informačných technológii. Počas bakalárskeho štúdia získal vedomosti a skúsenosti z oblasti umelej inteligencie, paralelného spracovania údajov a simulácie. Ovláda jazyky Java, C a Python. Pracuje na pozícii programátora v jazyku Java, kde získal skúsenosti s nástrojmi na riadenie projektov, s verziovacími nástrojmi a rozsiahle skúsenosti s JUnit testovaním.

2.2 Ponuka pre tému Robotický futbal (RoboCup)

Téma RoboCup zaujala náš tím vďaka tomu, že patrí medzi projekty s dlhou tradíciou na Fakulte informatiky a informačných technológií. Náš tím sa chce zapojiť do riešenia tohto zaujímavého nápadu a prispieť k prezentácii našej fakulty v lige simulovaného futbalu.

Hlavnou výhodou nášho tímu je fakt, že jeden člen tímu v rámci svojej bakalárskej práce riešil zadanie v oblasti vyššieho správania brankára. Vďaka tejto skúsenosti má hlbšie vedomosti z tejto domény a pozná úzke hrdlá komunikácie medzi hráčmi a problémy pri koordinácii hráčov. Ďalší dvaja členovia tímu sa vo svojich bakalárskych prácach venovali oblasti genetických algoritmov a jeden z členov sa venoval téme zaoberajúcej sa strojovému učeniu. Tieto prístupy sú tiež uplatniteľné pri rozvíjaní hráčov RoboCupu.

Väčšina tímu taktiež absolvovala bakalárske predmety ako Umelá inteligencia a Modelovanie a simulácia vyučovaných na Fakulte informatiky a informačných technológií, kde získali skúsenosti pri tvorbe umelej inteligencie a pri tvorbe simulácií. Výhodou je taktiež, že všetci členovia tímu veľmi dobre poznajú platformu Java a pracovali na viacerých projektoch v tomto jazyku. Niektorí členovia taktiež aktívne pracujú s jazykom Ruby na viacerých projektoch.

Hlavným cieľom, ktorý chceme dosiahnuť je úprava vyššieho rozhodovania hráčov, kedy by hráči pozorovaním hracej plochy vedeli vykonávať akcie dynamickejšie podľa aktuálnej situácie na ihrisku s pomocou plánovania viacerých akcii dopredu. Z toho vyplýva analyzovanie aktuálneho systému a návrh nového flexibilnejšieho a modulárnejšieho mechanizmu správania. Týmto by sa zdrojový kód hráča sprehľadnil a zlepšila by sa jeho rozšíriteľnosť, keďže v aktuálnej verzii sú niektoré časti kódu vyššej úrovne rozhodovania

duplicitné. Ďalším cieľom je doplnenie schopností hráča o výsledky úspešných BP a DP prác, aby mohol používať najlepšie schopnosti, ktoré sa doposiaľ študentom podarilo vytvoriť.

2.3 Ponuka pre tému Distribuované počítanie na FIIT

Popri riešení rôznych úloh, či už súvisiacich so štúdiom, vypracovávaní bakalárskych prác alebo mimo štúdia, sa každý z nás stretol s problémom výpočtovej zložitosti a časovými obmedzeniami, ktoré v mnohých prípadoch zbrzďujú dopracovanie sa k požadovanému výsledku. Viacerí z nás absolvovali povinne voliteľný predmet Paralelné programovanie s entuziazmom a nie jeden z nás následne využil získané vedomostí pri riešení vlastných projektov s cieľom zlepšiť prácu s programom a dostatočne využiť potenciálny výpočtový výkon. Aj v prípade, že sa nejednalo o priamu požiadavku riešenia a dodatočné zapracovanie bolo implementované z vlastnej iniciatívy. Z tohto dôvodu má každý z nás kladný vzťah k distribuovanému počítaniu a práve táto téma sa už zo základného popisu javí ako mimoriadne zaujímavá a každému členovi nášho tímu je jasná užitočnosť integrácie technológie BOINC v rámci fakultnej siete.

Zavedenie platformy BOINC na našej fakulte by otvorilo dvere študentom, ktorí potrebujú dostupný výpočtový výkon presahujúci možnosti ich vlastných zariadení a zefektívnilo by sa využitie výpočtových prostriedkov inštalovaných na fakulte pre potreby rozsiahlych výpočtových úloh.

Oproti existujúcemu riešeniu na fakulte by bolo pomocou tejto technológie možné využiť viac dostupných prostriedkov, ktoré zahŕňajú napríklad v noci nepoužívané počítače v počítačových učebniach a ostatných pracovníkov fakulty, ako aj osobné počítače a mobilné zariadenia študentov. Výsledný výkon môže pokojne presiahnuť bežné riešenia fakultných gridov a môže fakulte ušetriť značnú časť financií. Dôkazom je existujúce riešenie na londýnskej univerzite.

Zaujímavé by bolo otestovať vytvorenú infraštruktúru na novom fenoméne v oblasti výpočtovo náročných úloh, ktorým je tzv. ťažba bitcoin-ov. Bitcoin je decentralizovaná „kryptomena“, ktorej základom je výpočet hašovacej funkcie. Výpočet tejto funkcie nie je pamäťovo náročný a teda je vhodným kandidátom na distribuované počítanie pomocou technológiou BOINC.

3 Úlohy členov tímu

Rozdelenie pozícií v tíme je uvedené v tabuľke 3.1.

Manažérska rola	Manažér
Vedúci tímu	Roman Moravčík
Manažér podpory vývoja	Martin Markech
Manažér dokumentovania	Michal Blanárik
Manažér rozvrhu	Tomáš Nemeček
Manažér monitorovania projektu	Filip Blanárik
Manažér rizík	Štefan Horváťh
Manažér kvality	Štefan Linner

Tabuľka 3.1 - Rozdelenie pozícií v tíme

4 Plán

4.1 Plán na zimný semester

V tabuľke Tabuľke 4.1 – Plán rozvrhu pre zimný semester je zobrazený pôvodný plán rozvrhu ,ktorý bol určený na prvý semester riešenia.

Číslo šprintu	Číslo týždňa	Dátum stretnutia	Plán stretnutia
0	1	25.09.2013	Vytvorenie ponuky tímu
	2	03.10.2013	Pridelenie témy, Analýza existujúceho riešenia, Konfigurácia podporných nástrojov
1	3	10.10.2013	Analýza existujúceho riešenia, Analýza zahraničných tímov, Zostavenie backlogu, Zmena správania agenta
	4	17.10.2013	
2	5	24.10.2013	Analýza zahraničných tímov, analýza diplomových prác, Odstránenie ruby z kódov
	6	31.10.2013	
3	7	7.11.2013	Návrh a implementácia novej architektúry pre strategickú, taktickú vrstvu a návrh vrstvy highskillov
	8	14.11.2013	
4	9	21.11.2013	Testovanie a dolaďovanie novej architektúry
	10	28.11.2013	
5	11	5.12.2013	Pridávanie novej funkcionality , Príprava na súťaž, Odovzdanie finálneho prototypu a dokumentácie na zimný semester
	12	12.12.2013	

Tabuľka 4.1 - Plán rozvrhu pre zimný semester

Plán na zimný semester sa oproti pôvodnému rozvrhu spomalil a museli sme upraviť rozvrh pre posledné 2 šprinty. Spomalenie nastalo kvôli slabej analýze existujúceho riešenia a následne zlému návrhu architektúry. Vysvetlením tohto spomalenia je aj potreba venovať sa zadaniam na iných predmetoch. Upravený rozvrh je zobrazený v tabuľke Tabuľke 5.2 - Upravený plán rozvrhu.

Číslo šprintu	Číslo týždňa	Dátum stretnutia	Plán stretnutia
4	9	21.11.2013	Implementácia a testovanie novej architektúry
	10	28.11.2013	
5	11	5.12.2013	Testovanie a dolaďovanie novej architektúry, Odovzdanie finálneho prototypu a dokumentácie na zimný semester
	12	12.12.2013	

Tabuľka 4.2 - Upravený plán rozvrhu

5 Manažment kvality

5.1 Metodika testovania

Osoby

Programátor – po naprogramovaní požadovanej funkcionality či už vytvorením novej alebo zmenou staršej implementácie. Programátor je každá osoba ktorá robí zásahy v zdrojových kódoch.

Hlavný programátor – osoba ktorá pozná problematiku testovanej funkcionality a vie, ako ktoré časti fungujú.

Tester – osoba poverená testovaním. Pri konkrétne zadaných úlohach ako je implementácia výpočtových funkcií, je ním programátor. Pri zložitejších prípadoch ako je úprava správania alebo pohybov, tester môže byť určený, keďže sa jedná o komplexné problémy a pre programátora je testovanie časovo náročné kvôli množstvu situácii ktoré môžu nastať a teda pri základnej kontrole funkčnosti nemusia byť všetky problémy odhalené.

Súvisiace metodiky

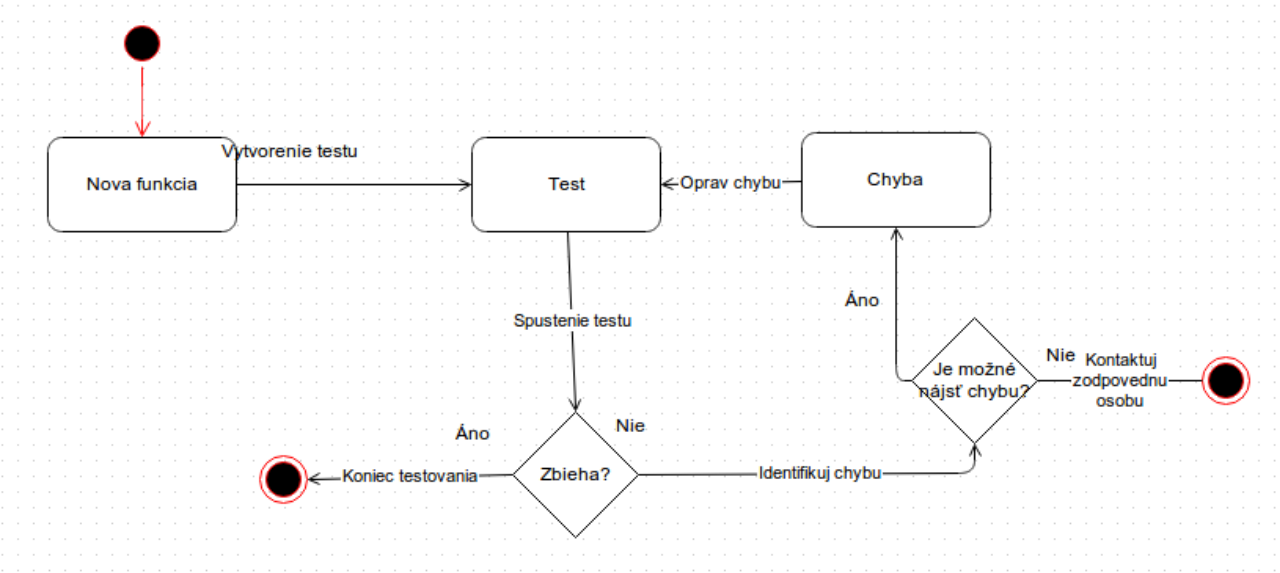
Metodika komunikácie - súvis v prípade potreby testovania alebo potreby pomoci od ďalšej osoby a to či už iného programátora alebo hlavného programátora

Metodika prehliadky zdrojových kódov – súvis s písaním automatických testov, ktoré tiež je potrebné okomentovať a napísať tak aby spĺňali štandard kódu.

Metodika manažmentu verziovania – súvisí s odovzdávaním testov a otestovaného alebo opraveného kódu do repozitára.

Testovanie

Automatické testovanie



Obrázok. 5.1 Diagram automatického testovania funkcii

Pre funkcie ktoré majú na starosť výpočty, nastavnovanie alebo spracovanie údajov, je potrebné vytvoriť automatické testy. Znáznornenie procesu je ukázané na diagrame na obrázku 5.1.

Automatický test pre nové samostatné statické metódy

1. Vytvorenie novej metódy v príslušnej triede. Napr. prevody jednotiek v triede Utils.
2. Vytvorenie testovacej triedy ktorá pozostáva z názvu triedy, v ktorej bola metóda vytvorená, a slova test. V ukážkovom prípade ak sa vytvorila metóda v triede Utils, testovacia trieda sa bude volať UtilsTest. Táto testovacia trieda sa pridáva do adresára Tests.
3. V testovacej triede sa vytvorí metóda - public void implementovanáMetódaTest. Nad ňu je potrebné napísať @Test ako označenie že sa jedná o test.
4. V testovacej metóde sa zavolá testovaná metóda, ktorej výsledok sa uloží do dočasnej premennej.

Následnej sa použije overenie triedou Assert s metódou assertEquals, ktorá ma vstupné dva parametre a to očakávaná hodnota a skutočná hodnota.

Napríklad Assert.assertEquals(true, jePredomnou(mojaPozícia, súperovaPozícia)) je test, či pre zadané hodnoty bolo správne vypočítané, že súper predomnou.

5. Je potrebné napísať dva testovacie prípady. Jeden, keď sa očakáva pozitívny výsledok, a druhý negatívny výsledok.
6. Testovaciu triedu je potrebné spustiť. Spôsob spustenia záleží od vývojového prostredia. V NetBeans stačí kliknúť pravým tlačidlom myši na testovaciu triedu a zvoliť test. Spustí sa test, ktorého výsledok je možné vidieť v záložke spodnej lišty.
7. Testovaciu triedu je potrebné pridať triedy AllTest v podobte TriedaTest.class. Týmto je zabezpečené že sa nový test spustí pri spustení všetkých testov.

Automatický test pre nové členské metódy triedy

1. Vytvorenie novej metódy v príslušnej triede. Napr. určenie a vrátenie pozície na základe údajov v triede.
2. Vytvorenie a pridanie testovacej triedy a metódy, ak neexistujú na základe predchádzajúcej časti o pridávaní testov pre samostatné metódy.
3. Keďže sa jedná o metódu ktorá používa údaje triedy, je potrebné v testovacej metóde vytvoriť inštanciu tejto triedy. Následne treba nastaviť potrebné premenné pred zavolaním testovanej metódy a to najprv pre pozitívny scenár a následne pre negatívny scenár. Pre overenie výsledku sa používa trieda Asssert so svojimi metódami.
4. Spustenie testov

Automatický test pre zmenené metódy

1. Zmena implementácie metódy.
2. Spustenie testu.
3. Ak test neprešiel, je potrebné skontrolovať novú implementáciu danej funkcie a opraviť chybu a následne spustiť test.
4. Ak test neprechádza a nie je možné nájsť chybu, je potrebné kontaktovať ďaľšieho programátora so žiadosťou o pomoc pri hľadaní chyby alebo hlavného programátora s informovaním o problémoch v novej implementácii.
5. Ak sa menili vstupné parametre metódy alebo návratová hodnota, je potrebné upraviť aj príslušnú testovaciu metódu v danej testovacej triede a následne spustiť test. Ak neprešiel, postup je ako v bode 4.

Manuálne testovanie

Niektoré časti funkcionality projektu nie je možné otestovať automaticky a to konkrétne správanie, rozhodovanie a pohyby hráča. Požadované správanie je dané implementáciou správania, ktoré bolo určené hlavným programátorom.

Testovanie nového pohybu

1. Vytvorenie pohybu ako xml súboru.
2. Vytvorenie plánu a schopnosti pre spustenie testovaného pohybu.
3. Spustenie simulačného prostredia, hráča a hry.
4. Pozorovanie hráča a jeho vykonávania pohybu. Ak je pohyb nestabilný alebo neprebieha ako má, je potrebné pozorovaním určiť, ktoré časti pohybu sú chybné a upraviť ich.
5. Opakovanie bodov 4. a 5. kým pohyb neprebieha ako je vyžadované.

Testovanie nových schopností a plánov.

1. Vytvorenie schopnosti alebo plánu.
2. Spustenie simulačného prostredia a hráča.
3. Vytvorenie podmienok na ihrisku, tak, aby bola splnená prvá situácia alebo podmienka v pláne alebo schopnosti.
4. Spustenie hry a teda simulácie.

5. Overenie, napríklad výpismi v konzoli a vizuálne na simulácii, či hráč vykonal potrebné akcie ktoré mal pre danú situáciu.
6. Zastavenie simulácie ak je to potrebné, napríklad hráč sa dostal do nevyriešiteľného stavu, alebo vytvorenie novej situácie.
7. Prejdenie všetkých podmienok alebo situácii ktoré sa nachádzajú v pláne.
8. Ak je s niektorou časťou problém, treba ho identifikovať a vyriešiť ho. Ak to nie je možné a je zlý napríklad návrh, je potrebné kontaktovať zodpovednú osobu tj. hlavného programátora.
9. Ak nie je možné niektorú situáciu vytvoriť, daná časť sa zakomentuje, označí sa tagom FIXME: a prejde sa na ďalšiu časť. Ak je zakomentovaná časť kritická pre správanie hráča, je potrebné problém zapísať a prejsť na bod 10.
10. Ak niektoré časti vytvárajúce správanie hráča boli zakomentované a odstavené, ich zoznam sa pošle hlavnému programátorovi.

Zmena v správaní

Postup je rovnaký ako pri vytvorení nového správania. V závislosti od implementácie treba prejsť všetky možnosti alebo iba tú jednu konkrétnu zmenenú.

Hľadanie chýb

Pri hľadaní chýb je možné používať testovacie výpisy a to konkrétne triedy AgentInfo pomocou metódy loguj("text") ktorá sa stará o vypisovanie správ do konzoly. Najlepšie použitie pre overenie správania správami podľa situácii napríklad: Som na zemi, Idem k lopte, Hľadám loptu a pod.
Pre výpočtové funkcie, ale je možné použiť aj pre akýkoľvek prípad, je najlepšie použiť breakpoint-y v kóde a spustiť projekt v debug móde. Keď sa kód dostane na miesto, ktoré je označené breakpointom, zastaví sa vykonávanie na danom mieste a je možné prejsť po krokoch ďalej kódom a pozerať a kontrolovať obsah premenných.

Anotácie pre metódy automatického testovania

Tieto anotácie je možné použiť podľa potreby pri písaní automatických testov.
Anotácia @Test je povinná vždy.

@Test
public void method() - identifikácie metódy ako testovacej metódy.

@Test (expected = Exception.class) – zlyhá ak nie je vyhodená výnimka daného typu

@Test(timeout=100) – zlyhá ak čas vykonávania je dlhší ako 100 milisekúnd

@Before
public void method() - metóda je zavolaná pre každým testom. Použitie je pre prípravu testovacieho prostredia napríklad inicializácia, načítanie vstupu a pod.

@After
public void method() - metóda je zavolaná po každom teste. Použitie je pre zrušenie zmien alebo uvoľnenie veľkých objektov z pamäte.

Zdroje:

http://www.vogella.com/articles/JUnit/article.html#unittesting_junit

6 Manažment rizík

Pri rôznych projektoch je vhodné analyzovať možné riziká, ktoré sa pri vývoji môžu objaviť a negatívne ovplyvniť výstup projektu. Cieľom manažmentu rizík je minimalizovať možnosť zhmotnenia rizika, ak je to možné, alebo minimalizovať účinok jeho skutočného zhmotnenia. Vďaka analýze rizík dokážeme zvýšiť šance projektu na jeho úspešné zavŕšenie.

S ohľadom na charakter nášho projektu, sme identifikovali hlavné generické a špecifické riziká, zobrazené v tabuľke (*tabuľka 6.1*). Obsahuje pravdepodobnosť nastátia (výskytu) udalosti aj rozsah škôd v stupnici hodnôt nízka, stredná a vysoká. Konkrétne riziká sú v nasledovných častiach priblížené bližšie.

Skupina	Riziko	Pravdepodobnosť výskytu	Miera dopadu
Generické	Nerealistické rozvrhy	Stredná	Vysoká
	Nedostatok požadovaných znalostí a zručností skúseností vývojového tímu	Nízka	Stredná
Špecifické	Nedostatočné porozumenie zavedených princípov z minulosti	Stredná	Stredná
	Konflikty medzi spolupracujúcimi tímami	Stredná	Nízka

Tabuľka 6.1 Prehľad rizík

6.1 Generické riziká

Nerealistické rozvrhy

Základným problémom väčšiny projektov sú nerealistické odhady a plány. Tento problém je umocnený v prípade príchodu nových členov tímu alebo v prípade úplne nového zloženia tímu, nehovoriac o prípadoch, kedy členovia tímu pracujú paralelne na viacerých projektoch.

Vyhnutie tomuto riziku spočíva hlavne v analýze všetkých prebiehajúcich projektov členov tímu v dostatočnom predstihu, nie neskôr ako pred začatím nového šprintu. Kritické je v tomto prípade začať s prácou na tomto projekte hneď pri začatí šprintu, čím riešiteľ zníži riziko nesprávneho odhadnutia času vypracovávanej úlohy, poprípade bude ešte dostatok času na upravenie odhadovaného času, poprípade prerozdelenie časti práce na ďalších členov tímu.

Nedostatok požadovaných znalostí a zručností vývojového tímu

Bežným rizikom je taktiež nedostatok požadovaných znalostí a zručností vývojového tímu. Zameranie členov tímu môže byť rovnaké, no pravdepodobne nie každý má úplne rovnaké skúsenosti a znalosti z danej oblasti. Pri spolupráci skúsenejších a menej skúsených členov tímu na tímovej úlohe je pre jej úspešné dokončenie kľúčové, aby každý člen tímu bol na

rovnakej úrovni porozumenia riešeného problému ako jeho kolegovia. Povinnosťou každého člena tímu, ktorý si všimne, že rozumie problému lepšie, je teda uprednostniť oboznámenie kolegov s daným problémom ešte pred jeho riešením.

6.2 Špecifické riziká

Nedostatočné porozumenie zavedených princípov z minulosti

Špecifickou črtou nášho projektu oproti iným tímovým projektom je funkčný stav projektu. Väčšina tímov začína vytvárať produkt, viac menej od nuly, no náš tím nadväzuje na projekte RoboCup 3D, vyvíjaného na našej fakulte už niekoľko rokov. Do projektu bolo už investované množstvo človekodní, čoho výsledkom je rozsiahly produkt. Veľké riziko (respektíve jeho dopad) teda spočíva v nedostatočnom oboznámení nových riešiteľov s každým zákutím projektu. Výsledkom môže byť dokonca znefunkčnenie existujúcej funkcionality alebo istých častí zdrojového kódu.

Predísť tejto situácií sa samozrejme dát zvýšením dôrazu na preverenie dostatočného pochopenia projektu každého člena tímu, no plytvanie času na takéto účely nie je efektívne, a tím tak stráca na zdrojoch. Výskyt tohto rizika však nie je nikdy možné úplne odstrániť, no je možné zvýšiť jeho detekciu a znížiť jeho dopad pre budúcich riešiteľov tvorbou dostatočného množstva unit testov. Samozrejmé je zabezpečiť dokumentovanie projektu, ktoré musí spĺňať požadovanú kvalitu.

Konflikty medzi spolupracujúcimi tímami

Ak spolupracuje na jednom projekte viac tímov, ako je to v našom prípade, vzniká tu riziko konfliktu medzi spolupracujúcimi tímami. Konflikt môže byť napr. vytvorenie nekompatibilných komponentov, ktoré sa nedajú nijak jednoducho integrovať. Po integrácii takýchto komponentov vzniká chaos v kóde a v konečnom dôsledku je výstup dosť neefektívny pretože musí existovať mediátor, ktorý zabezpečuje komunikáciu. Toto riziko vzniká v prípade zlého porozumenia a naivného prístupu k problému. Riešením je intenzívna komunikácia medzi tímami a dobrá dokumentácia, v ktorej si vedia všetci členovia tímu nájsť informácie, ktoré im prípadne unikli počas stretnutí.

Iným druhom konfliktu môže byť práca na rovnakých alebo podobných úlohách. Toto je jasný príklad nedobrej organizácie, komunikácie a nejasného rozdelenia úloh pre tímy. Riešiť sa to dá rovnako komunikáciou, no hlavne dobrým plánovaním úloh. K zníženiu rizík konfliktov by všeobecne mohlo pomôcť viacstupňový manažment, kde každý tím by mal svojho vedúceho a najsilnejšia komunikácia by šla medzi vedúcimi tímov.

Vypadnutie člena tímu na šprint stretnutí

Hoci by sa členovia tímu mali snažiť stretávať sa pravidelne a v kompletnej zostave, existuje tu riziko že jeden alebo viac členov tímu sa stretnutia nezúčastnia. Dôvodov existuje niekoľko, no väčšinou má na to člen tímu racionálny dôvod. V takomto prípade je úlohou tímu zabezpečiť informácie pre chýbajúceho člena, aby nevznikalo riziko neporozumenia problematiky. Na stretnutí treba preberať veci, za ktoré nie je zodpovedná chýbajúca osoba, treba čo najviac maximalizovať produktivitu tímu na stretnutí. Ak chýba scrum master, každý člen tímu by mal mať prehľad čo sa robilo a čo bolo naplánované, teba tím si zvolí iného scrum mastera podľa internej dohody.

Nedokončenie stanovených úloh

Často sa stáva, že tím nedokončí stanovené úlohy na obdobie šprintu. Dôvodov existuje veľké množstvo, no najčastejšie býva nedostatok času, chýbajúce skúsenosti, zlé plánovanie a časový odhaď a zlý odhad výkonnosti tímu. Ak nastane takáto situácia, je treba určiť, aká veľká časť úloh ostala nedokončená a koľko ešte chýba k dokončeniu. Následne treba prehodnotiť priority a zvážiť dokončenie úloh v období nasledujúceho šprintu. Na elimináciu tohto rizika je dobré sledovať efektivitu tímu pri riešení úloh a viesť štatistiky vyriešených a nevyriešených úloh. Z týchto metrických údajov je možné zhodnotiť výkonnosť tímu lepšie tak odhadnúť kvantum úloh na ďalší šprint.

7 Manažment rozvrhu (plánovanie)

7.1 Spôsob plánovania

Tím používal ako metodiku vývoja techniku SCRUM. Táto metodika patrí medzi agilné metodiky prebiehajúce s inkrementálno-iteratívnym vývojom. Pri tomto druhu vývoja sa úlohy prideľujú a riešia počas iterácií. Jedna iterácia sa nazýva šprint a trvá pevne stanovenú dobu, ktorú sme si v rámci nášho tímu stanovili na dva týždne. Dôležitou úlohou v metodike SCRUM je pozícia vlastníka produktu (product owner). V našom tíme úlohu vlastníka zastával pedagogický vedúci tímu. Ďalšou dôležitou pozíciou je vedúci stretnutia(scrum master). Táto pozícia kolovala medzi všetkými členmi tímu, tak aby každý šprint bol vedúcim

7.2 Podporné nástroje

JIRA

Ako podporný nástroj pre plánovanie a evidenciu úloh sme vybrali nástroj **Atlassian JIRA**. Tento software patrí do rodiny nástrojov na podporu vývoja v tíme od spoločnosti Atlassian. Jira je nástroj na evidenciu projektov, úloh, ludí, zdrojov a chýb. Ďalšou vlastnosťou tohto nástroja je automatické generovanie manažérskych diagramov ako **Burndown chart.** Tento nástroj sme zvolili z nasledujúcich dôvodov:

- Prehľadné používateľské rozhranie
- Dostupný z siete Internet – webová aplikácia
- Veľké množstvo funkcií
- Integrovateľnosť s množstvom nástrojov na podporu vývoja od spoločnosti Atlassian
- Skúsenosti členov tímu

Hlavným využitím v tíme je možnosť sledovania stavu úloh počas riešenia projektu. Ďalšou výhodou je podpora agilného vývoja metodikou SCRUM a delenia úloh do jednotlivých šprintov a vytvorenie produktovej nástenky. Manažér plánovania sa stará o vytvorenie a uzavretie úloh. Každý člen tímu aktualizuje stav jemu pridelenej úlohy a zapisuje strávený čas na úlohe a stručne charakterizuje výsledok riešenia úlohy.

7.3 Metodika plánovania

Úvod

Táto metodika sa zaoberá manažmentom plánovania úloh v rámci šprintu. Konkrétne zápisom úlohy do aktuálneho nového šprintu.

Túto metodiku využívajú vedúci stretnutí a manažér plánovania pri vytváraní plánu nového šprintu.

Použité pojmy

Issue – Znamená v preklade problém. V kontexte evidencie úloh je issue chápaný ako zadanie, ktoré má dodatočne definovaný typ.
SCRUM – Metodika riadenia projektu
Šprint – Jeden cyklus iterácie v projekte riadenom metodikou SCRUM.
Nástenka produktu – Zoznam navrhovanej funkcionality na výsledný systém, ktorá bola schválená Vlastníkom produktu. (**product backlog**)
Šprintová nástenka - Zoznam navrhovanej funkcionality, ktorá sa má v rámci jedného šprintu implementovať a po ukončení šprintu odovzdať zákazníkovi. (s**print backlog**)
Vlastník produktu – Člen tímu zastupujúci zákazníka. (**product owner**)
Vedúci stretnutia – člen tímu zodpovedný za vedenie šprintu. Členovia tímu sa striedajú na pozícii vedúceho stretnutia pre každý nový šprint. (**scrum master**)
Manažér plánovania - Člen tímu zodpovedný za plnenie celkového plánu a správca nástroja Jira.
Tím – Skupina ľudí pracujúcich na rovnakom projekte pomocou metodiky SCRUM
Riešiteľ – Riešiteľ úlohy. (**assignee**)
Tester – Je osoba, testujúca výstup jednotlivých úloh.
Jira – Nástroj na podporu projektového riadenia a evidenciu úloh vyvinutý spoločnosťou Atlassian.

Zoznam nadväzujúcich metodík

Na túto metodiku nadväzujú metodiky:

- Metodika manažmentu verziovania
- Metodika prehliadky zdrojových kódov
- Metodika dokumentácie
- Metodika testovania

Roly a zodpovednosti účastníkov

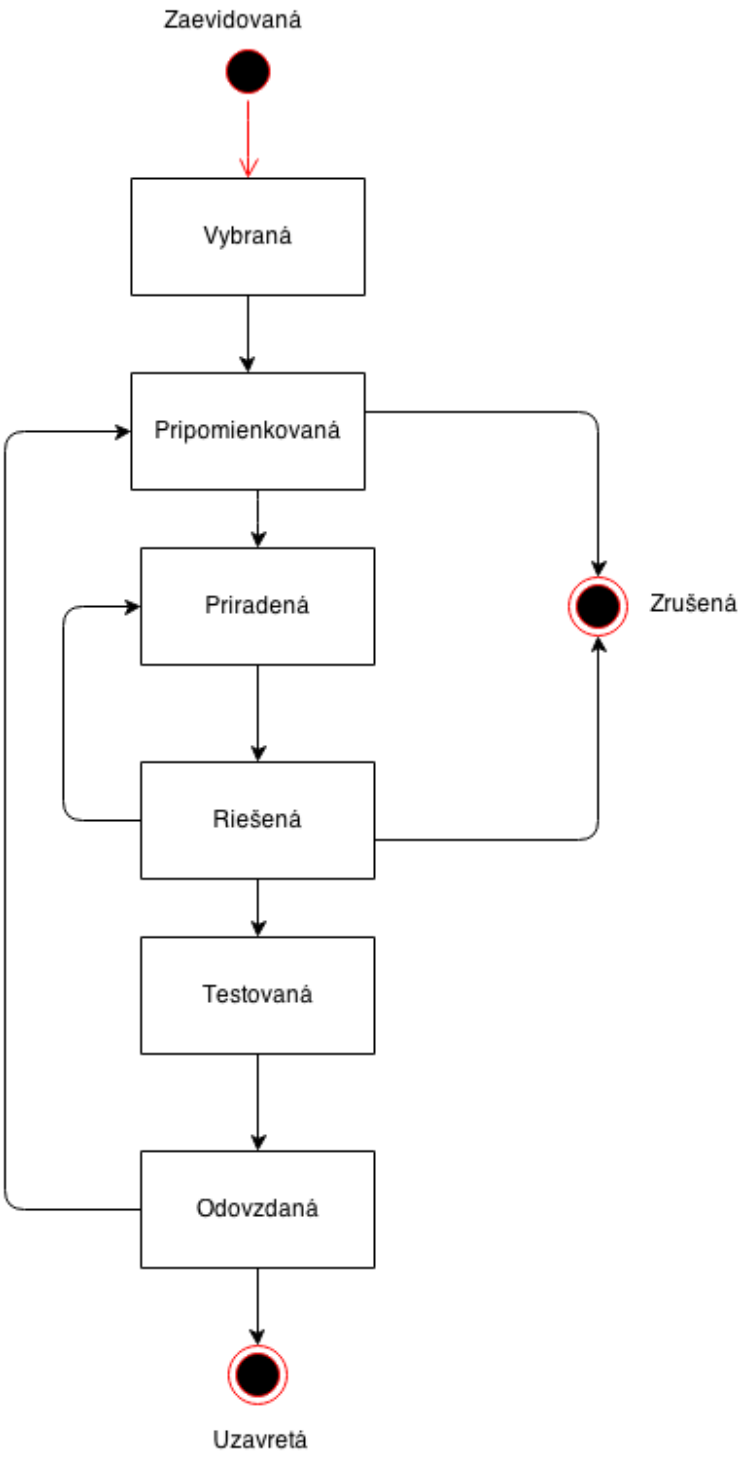
V tejto kapitole sú identifikované jednotlivé roly účastníkov zúčastňujúcich sa procesov spojených s prípravou šprintu. Jednotlivé roly účastníkov sú zobrazené v Tabuľke 7.1 – Roly účastníkov.

Rola	Zodpovednosť
Vlastník produktu	• Zástupca zákazníka • Schválenie finálneho zadania úlohy • Schválenie prevedenia úlohy
Vedúci stretnutia	• Plnenie plánu v rámci šprintu • Zodpovedný za ukončenie šprintu
Manažér plánovania	• Plnenie plánu • Vytvorenie šprintovej nástenky • Aktualizovanie šprintovej nástenky
Tím	• Každý člen si musí vybrať úlohu zo šprintu
Riešiteľ	• Riešiteľ úlohy • Odovzdanie výstupu úlohy na testovanie
Tester	• Testovanie výstupu úlohy • Potvrdenie správnosti výstupu

Tabuľka 7.1 – Roly účastníkov

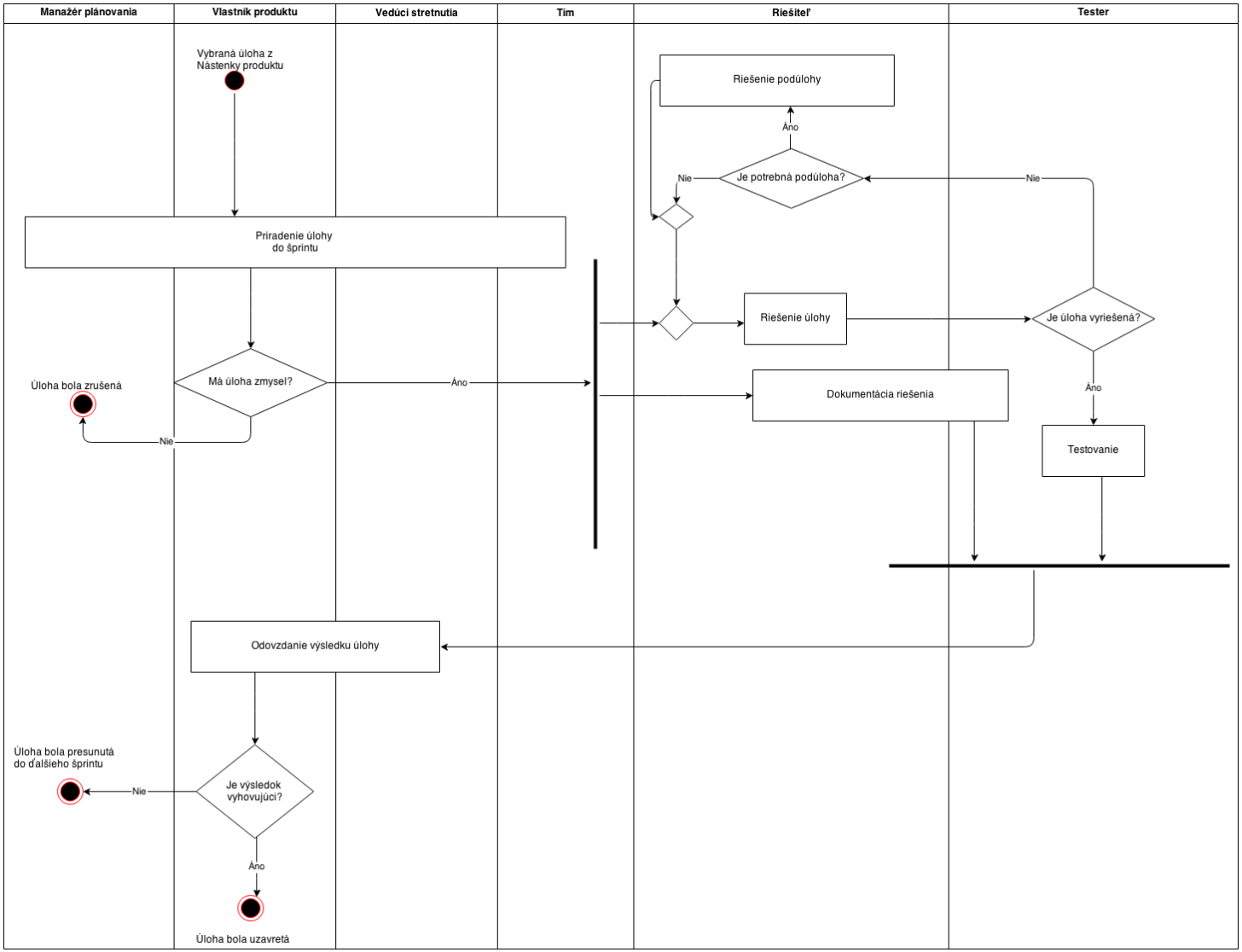
Identifikované procesy

V kapitole Identifikované procesy je znázornený Obrázku 7.1 Stavy úlohy, kde sú zobrazené jednotlivé stavy, ktoré môže úloha počas svojhu životného cyklus dosiahnúť.



Obrázok 7.1 Stavy úlohy

Na Obrázku 7.2. Identifikované procesy sú vizualizované identifikované procesy vykonávané v rámci šprintu.



Obrázok 7.2 Identifikované procesy

Priradenie úlohy do šprintu

Názov:	Pridanie úlohy do šprintu
Zodpovedný:	Vedúci stretnutia
Vykonávateľ:	Tím, Vlastník produktu, Vedúci stretnutia, Manažér plánovania
Vstup:	Vybraná úloha z nástenky produktu
Výstup:	Formálne upravená úloha

Tabuľka 7.2 priradenie úloh do šprintu

Tento proces je vykonávaný na začiatku každého šprintu, kedy vlastník produktu vyberie úlohy, ktoré chce v rámci šprintu riešiť. Potom vedúci stretnutia vedie diskusiu kedy vlastník produktu spolu s tímom diskutuje o zadaní úlohy o určení prácnosti a časovej náročnosti úlohy. Po schválení dohodnutého zadania manažér plánovania doplní tieto údaje do záznamu úlohy v nástroji Jira. Následne si každý člen tímu vyberá úlohy na ktorých chce v rámci šprintu pracovať.

Pri každej činnosti dodefinovania atribútov úlohy vlastník produktu schvaľuje dohodnuté hodnoty a v prípade ak sa mu hodnota nepozdáva, vyzve tím na zopakovanie aktivity.

Zoznam vykonávaných činností pre tento proces je v tabuľke Tabuľke 7.3. Zoznam činností - Priradenie úlohy do šprintu.

Číslo činnosti	Názov činnosti
5.1.1	Dodefinovanie zadania úlohy
5.1.2	Dodefinovanie prácnosti úlohy
5.1.3	Dodefinovanie časovej náročnosti úlohy
5.1.4	Doplnenie záznamu úlohy v nástroji
5.1.5	Pridanie úlohy do šprintovej nástenky
5.1.6	Pridanie úlohy riešiteľovi

Tabuľka. 7.3 Zoznam činností - Priradenie úlohy do šprintu

Dodefinovanie zadania úlohy

Názov:	Dodefinovanie zadania úlohy
Zodpovedný:	Vedúci stretnutia
Vykonávateľ:	Vlastník produktu, tím
Vstup:	Vybraná úloha z nástenky produktu
Výstup:	Formálne upravená úloha

Tabuľka 7.4 dodefinovanie zadania úlohy

Členovia tímu spolu s vlastníkom produktu vybrali na začiatku šprintu úlohu, ktorá sa má v rámci šprintu vykonať.

Ak zadanie nespĺňa požiadavky zákazníka, vlastník produktu túto úlohu zruší, inak je potrebné dodefinovať zadanie úlohy.

Členovia tímu s vlastníkom produktu dodefinujú rozšírenie textu zadania úlohy v slovenčine. Pri dodefinovaní zadania úlohy sa pýtajú na cieľ úloh a ako má úloha vylepšiť produkt. Diskusia k zadaniu úlohy trvá maximálne 5 minút. Z výsledku diskusie sa zapíše zadanie úlohy.

Dodefinovanie prácnosti úlohy

Názov:	Dodefinovanie prácnosti úlohy
Zodpovedný:	Vedúci stretnutia
Vykonávateľ:	Tím
Vstup:	Vybraná úloha z nástenky produktu s upraveným zadaním
Výstup:	Vybraná úloha z nástenky produktu s upraveným zadaním a prácnosťou

Tabuľka 7.5 dodefinovanie prácností úlohy

Následne členovia tímu určujú prácnosť pomocou kariet (scrum cards) a hry plánovací poker (planning poker), kedy všetci členovia tímu hlasujú za hodnoty na kartičkách. Hlasovanie prebieha ukazovaním vybranej karty každého člena tímu a následnou diskusiou o najvhodnejšej hodnote spomedzi vybraných kariet. Diskusia trvá maximálne 1 minútu v rámci jedného kola. Tento proces sa opakuje pokiaľ sa členovia tímu nezhodnú na jednej hodnote.

Dodefinovanie časovej náročnosti úlohy

Názov:	Dodefinovanie časovej náročnosti úlohy
Zodpovedný:	Vedúci stretnutia
Vykonávateľ:	Manažér plánovania
Vstup:	Vybraná úloha z nástenky produktu s upraveným zadaním a prácnosťou
Výstup:	Formálne upravená úloha

Tabuľka 7.6 dodefinovanie časovej náročnosti úlohy

Manažér plánovania po určení prácnosti úlohy zapisuje odhad časovej náročnosti úlohy v hodinách podľa určenej hodnoty prácnosti. Kedy hodnota časovej náročnosti sa rovná dvojnásobku hodnotu prácnosti úlohy.

Doplnenie záznamu úlohy v nástroji

Názov:	Doplnenie záznamu úlohy v nástroji
Zodpovedný:	Vedúci stretnutia
Vykonávateľ:	Manažér plánovania
Vstup:	Formálne upravená úloha
Výstup:	Formálne upravená úloha je uložená v nástroji Jira

Tabuľka 7.7 doplnenie záznamu úlohy v nástroji

Po doplnení všetkých atribútov úlohy manažérom plánovania a schválením tejto úpravy vlastníkom produktu sa modifikácia zadania zapíše do nástroja Jira nasledovne:

1. Otvorenie Agile nástenky.
2. Presunutie sa do nástenky produktu .
3. Výber zvolenej úlohy.
4. Zvolenie možnosti úpravy.
5. Doplnenie dohodnutej hodnoty prácnosti na stretnutí do poľa units.
6. Doplnenie dohodnutej časovej náročnosti na stretnutí do poľa estimated time vo formáte xxxh.
7. Doplnenie zadania úlohy do poľa description.
8. Doplnenie mena vedúceho stretnutia do poľa Zadávateľ.
9. Uloženie zmien.

Pridanie úlohy do šprintovej nástenky

Názov:	Pridanie úlohy do šprintovej nástenky
Zodpovedný:	Vedúci stretnutia
Vykonávateľ:	Manažér plánovania
Vstup:	Formálne upravená úloha v nástroji Jira
Výstup:	Úloha pridaná do šprintovej nástenky

Tabuľka 7.8 pridanie úlohy do šprintovej nástenky

Po dodefinovaní atribútov úlohy je potrebné priradenie takto upravenej úlohy do šprintovej nástenky. Tento krok vykonáva manažér plánovania počas stretnutia v nástroji Jira podľa nasledujúceho postupu:

1. Otvorenie Agile nástenky.
2. Výber šprintu
   2.1. Ak nie je vytvorený šprint, vedúci stretnutia vytvorí nový šprint pomocou nástroja.
   2.2. Ak je vytvorený šprint, vedúci stretnutia nevykoná žiadnu akciu.
3. Vedúci stretnutia vyberie zvolenú úlohu a metódou „drag-and-drop“ presunie vybranú úlohu z nástenky produktu do šprintovej nástenky.

V tomto bode je potrebné bližšie definovanie postupu vytvorenia šprintu v nástroji Jira.

1. Otvorenie Agile nástenky.
2. Presunutie sa do nástenky produktu .
3. Zvolenie možnosti Create Sprint.
4. Doplnenie názvu šprintu tvoreného slovom Šprint a poradovým číslom šprintu.
5. Doplnenie dátumu začatia a ukončenia šprintu.
6. Začatie nového šprintu.

Priradenie úlohy riešiteľovi

Názov:	Priradenie úlohy riešiteľovi
Zodpovedný:	Vedúci stretnutia
Vykonávateľ:	Manažér plánovania, Tím, Vedúci stretnutia
Vstup:	Úloha zo šprintovej nástenky
Výstup:	Úloha bola priradená riešiteľovi

Tabuľka 7.9 priradenie úlohy riešiteľovi

Po pridaní úlohy do šprintovej nástenky si túto úlohu môže vybrať jeden člen tímu.

Tento proces prebieha vo forme diskusie a následnej dohody medzi členmi tímu. Diskusia je riadená vedúcim stretnutia, ktorý vykonáva funkciu mediátora a snaží sa o najvhodnejšie a najférovejšie pridelenie úlohy medzi jednotlivých členov.

Po pridelení úlohy manažér plánovania v nástroji Jira priradí vybranú úlohu svojmu riešiteľovi nasledovne:

1. Otvorenie Agile nástenky.
2. Presunutie sa do šprintovej nástenky.
3. Vybratie konkrétnej úlohy zo šprintovej nástenky.
4. Vyplnenie mena riešiteľa do poľa Assignee.
5. Potvrdenie a uloženie zmien v zadaní.

8 Manažment podpory vývoja a integrácie

8.1 Metodika manažmentu verziovania

Úvod

Táto metodika určuje pravidlá a postupy pre členov tímu č.9 – Gitmen. Slúži na zlepšenie organizácie práce s Gitom, určuje vhodné postupy a konvencie, podľa ktorých treba pri práci s Gitom v rámci tímu postupovať. Podrobne popisuje postupy pri práci s vetvami, pomenovávanie názvov komitov, či postupy pri synchronizovaní so vzdialenými vetvami.

Pojmy

- **Git** – systém na správu verzií. Metodika predpokladá používanie verzie 1.8.4
- **repozitár** – úložisko súborov s históriou verzií predstavujúce náš projekt
- **komit** – súhrn vykonaných zmien označených spoločným názvom zmeny
- **vzdialený repozitár** – repozitár nahraný na serveri, s ktorým sa synchronizujú jednotliví vývojári
- **lokálny repozitár** – repozitár u každého vývojára, ktorý je prístupný iba pre seba
- **vetva** – línia vývoja s vlastnou históriou zmien
- **šprint** – etapa vývoja pri používaní metodológie Scrum
- **konflikt** – konflikt nastane ak viacero vývojárov modifikuje rovnakú časť súboru
- **squash** – zlúčenie x posledných komitov do jedného pre lepšiu prehľadnosť zmien
- **Jira** – nástroj na plánovanie a organizáciu úloh

Roly a zodpovednosti

Správca repozitára

- má za úlohu dohľad nad repozitárom,
- v prípade odoslania zmeny, ktorá pokazí zostavenie (anglicky *"build"*) projektu, má za úlohu odstránenie daného komitu z repozitára
- dohľad nad funkčnosťou externých služieb – Bitbucket

Projektový manažér

- má za úlohu značenie vetiev po jednotlivých šprintoch

Vývojár

- tvorba a odosielanie zmien
- riešenie konfliktov pri spájaní vetiev

Zoznam nadväzujúcich metodík a dokumentov

Metodika manažmentu verzií úzko súvisí s týmito metodikami:

- Manažment úloh v rámci šprintu
- Metodika prehliadky zdrojových kódov

Procesy manažmentu verziovania

Manažment verziovania obsahuje nasledovné procesy:

- Proces implementácie úlohy
- Proces odoslania zmien do vzdialeného repozitára
- Zlúčenie komitov
- Proces značenia v jednotlivých vetvách
- Proces spájania vetiev
- Proces vybratia zmien do druhej vetvy
- Proces riešenia kolízií pri spájaní vetiev

Proces implementácie úlohy

- Ak vývojár nie je nastavený na vetve master, nastaví sa do nej pomocou príkazu
  > git checkout master
- Stiahne si aktuálne zmeny zo vzdialeného repozitára pomocou
  > git pull origin master
- Ak vývojár plánuje minimálnu zmenu, zväčša opravu problému vrámci jednoho súboru, nemusí vytvárať novú vetvu, môže zostať na hlavnej master vetve.
- Ak vývojár plánuje väčšiu zmenu (viac ako jeden súbor), vytvorí si novú lokálnu vetvu. Do jej názvu uvedie číslo úlohy z Jiry, tj napríklad ROBOCUPTP-33.
- V tejto lokálnej vetve implementuje želanú funkcionalitu
- Pridá všetky vykonané zmeny pomocou
  > git add .
- Zmeny uloží pomocou
  > git commit -m "ROBOCUPTP-33 popis úlohy"
- Názov komitu vždy začína názvom projektu a čísla úlohy podľa nástroja Jira. Popis v komite musí byť pre celý tím v jednotnom jazyku, preto je potrebné vybrať si medzi slovenčinou alebo angličtinou a nemiešať jednotlivé jazyky. Preferovaný jazyk je anglický.

Proces odoslania zmien do vzdialeného repozitára

- Vývojár sa prepne do hlavnej vetvy príkazom
  > git checkout master
- Pred nahratím zmien do vzdialeného repozitára vývojár najskôr získa do svojej lokálnej vetvy zmeny zo vzdialeného repozitára a zlúči ich. Vykoná to príkazom
  > git pull origin master
- Nahrá zmeny do vzdialeného repozitára pomocou príkazu
  > git push origin master
- Ak mal vývojár vytvorenú lokálnu vetvu a prácu na danej úlohe dokončil, môže ju zmazať príkazom
  > git branch -D nazov_vetvy

- Ak vývojár túto jeho lokálnu vetvu nahral do vzdialeného repozitára, môže ju zmazať príkazom
  > git push origin :nazov_vetvy
  Najskôr sa ale musí uistiť, že nikto nezačal vyvíjať na tejto vetve.

Zlúčenie komitov

- Vývojár je nastavený vo vetve, v ktorej chce posledné komity zlúčiť do jedného
- Vykoná príkaz
  > git rebase HEAD~2
  kde číslo označuje počet komitov, ktoré chce zlúčiť
- V otvorenom textovom editore na prvom riadku ponechá slovo *pick*, na ďalších riadkoch prepíše slovo *pick* za slovo *squash*
- Uloží súbor a v ďalšom okne otvoreného editora napíše sumárny názov komitu pre upravované komity.
- Zlučovanie komitov používať iba na lokálnych vetvách. Použitie nad vzdialenou vetvou nie je povolené, nakoľko sa tým prepisuje história a niektorí vývojári by mohli mať medzi tým stiahnutú verziu s pôvodnou históriou.

Proces značenia v jednotlivých vetvách

- Značia sa iba zmeny v hlavnej vetve, zmeny v ostatných vetvách sa neznačia
- Úlohu značenia vetiev má na starosti produktový manažér po skončení šprintu
- V deň skončenia šprintu produktový manažér zlúči všetky vetvy do hlavnej a označí zmeny príkazom
  > git tag -a v1.2.3 -m "popis verzie 1.2.3"
- Pri číslovaní sa dodržiava sémantické značenie v tvare *x.y.z*. *X* reprezentuje hlavné číslo verzie projektu, ktoré sa mení iba pri veľkých zmenách. *Y* reprezentuje menšie zmeny, ktoré sú vrámci hlavného čísla spätne kompatibilné. *Z* reprezentuje iba drobné zmeny, typicky sú to verzie s opravami chýb.
- Následne produktový manažér nahrá označenia do vzdialeného repozitára príkazom
  > git push origin --tags

Proces spájania vetiev

- Keď je práca na lokálnej vetve s číslom úlohy dokončená, vývojár ju môže spojiť do hlavnej vetvy.
- Nastaví sa do hlavnej vetvy príkazom
  > git checkout master
- Spojí vetvu do hlavnej príkazom
  > git merge master
- Vyhnite sa spájaniu rôznych vetiev (mimo master) medzi sebou. Vždy je lepšie spájať vetvy do hlavnej vetvy, čím sa vývojári s touto vetvou synchronizujú a nenastane pri spájaní viacerých vetiev množstvo konfliktov.

Proces vybratia zmien do druhej vetvy

- Vývojár sa nastaví do vetvy, z ktorej chce preniesť zmenu (komit) do inej vetvy

príkazom
> git checkout nazov_vetvy

- Zobrazí si zoznam komitov príkazom
  > git log
- V prvom riadku každého odstavca komitu je za slovom "*commit"* zobrazený identifikátor daného komitu. Vývojár skopíruje aspoň prvých 6 číslic (nemusí kopírovať celý identifikátor.
- Prepne sa do hlavnej vetvy príkazom
  > git checkout master
- Zadá príkaz
  > git cherry-pick identifikator

  kde slovo *identifikator* reprezentuje skopírovaný identifikátor pre komit, ktorý chce preniesť do hlavnej vetvy
- Proces vyberania zmien nie je nutné aplikovať iba pri vyberaní zmien do hlavnej vetvy, môže byť použitý i pri vyberaní zmien medzi vetvami.

Proces riešenia kolízií pri spájaní vetiev

- Ak sa počas procesu spájania vetiev nepodarilo spojiť vetvy, Git zobrazí správu s názvami súborov, ktoré sa nepodarilo spojiť automaticky a treba ich spojiť manuálne.
- Vývojár otvorí tieto súbory v textovom editore, pre program gedit je príkaz
  > gedit cesta_k_suboru
- Začiatok a koniec problematickej oblasti, ktorú sa nepodarilo spojiť je vyznačený šípkami spolu s názvom vetvy, ku ktorej daná oblasť kódu patrí.
- Vývojár vyberie verziu, ktorú chce zlúčiť a odstráni druhú časť verzie označujúcu verziu z inej vetvy
- Skontroluje, či korektne zmazal pomocné šípky zavedené Gitom pre označenia rozsahu jednotlivých častí kódu
- Zatvorí textový editor a v konzoli pridá zmeny a uloží zmeny
  > git add .
  > git commit -m "merge robocup-33"
- V názve komitu sa pri manuálnom spájaní uvedie popis správy, v ktorom bude slovo merge, za ktorým bude nasledovať názov vetvy, ktorá sa spájala.

Obnova zmazanej vetvy

- Vývojár napíše
  > git reflog
- Vyberie a skopíruje číslo posledného komitu v zmazanej vetve
- Príkazom
  > git checkout -b nazov_zmazanej_vetvy sha1_identifikator
  obnoví zmazanú vetvu

Bibliografia

CHACON, S. *Pro Git* [online]. Apress, 2009. 288 s. ISBN 978-1430218333. Dostupné na internete <http://git-scm.com/book>

NVIE, A successful Git branching model [online]. Dostupné na internete <http://nvie.com/posts/a-successful-git-branching-model/ >

9 Manažment monitorovania projektu

9.1 Metodika prehliadky zdrojových kódov

Vymedzenie obsahu

Účelom metodiky je stanoviť postup akým je potrebné vykonávať prehliadky zdrojových kódov, aby bol výsledok prehliadky jednotný, dostupný všetkým zainteresovaným a nápomocný pri oprave prípadných nedostatkov zistených prehliadkou.

Prehliadka zdrojových kódov je spôsob ako zabezpečiť kvalitu výsledného produktu odhalením a následným odstránením chýb v čo najskoršej fáze vývoja pomocou systematickej kontroly zdrojových kódov.

Ciele prehliadky

- Overiť, či prehliadaný zdrojový kód spĺňa všetky definované požiadavky
- Overiť, či prehliadaný zdrojový kód spĺňa všetky atribúty kvality
- Overiť, či prehliadaný zdrojový kód spĺňa štábnu kultúru písania zdrojového kódu a jeho komentárov
- Identifikovať všetky odchýlky od požiadaviek
- Odhaliť možné vylepšenia prehliadaných zdrojových kódov

Použité softvérové nástroje

Na prehliadku je použitý **Eclipse** plug-in: **PerConIK** [1] v kooperácií s nástrojom na manažment projektu: **Jira** [3].

Zoznam nadväzujúcich metodík a dokumentov

Súvisiace metodiky

Metodika testovania
Metodika plánovania: Manažment úloh v rámci šprintu
Metodika písania zdrojových kódov
Metodika manažmentu verziovania

Dokumenty

PerConIK: Používateľská príručka
Dokumentácia k inžinierskemu dielu

Vymedzenie pojmov a skratiek

Šprint – Časovo ohraničený úsek, počas ktorého sa vytvára „hotový“, použiteľný, a potenciálne nasaditeľný prírastok produktu

Scrum – Rámec pre vývoj komplexných produktov a pre ich udržiavanie
Zdrojový kód – Postupnosť príkazov zrozumiteľných človeku napísaných v programovacom jazyku.
Jira – Softvérový nástroj od spoločnosti Atlassian podporujúci a uľahčujúci proces riadenia projektu, a požiadaviek. [3]
PerConIK – výskumný projekt zameraný na podporu vývoja softvérového produktu. Umožňuje dopĺňanie preddefinovaných alebo vlastných značiek do zdrojových kódov. [1,2]
Git – Distribuovaný systém riadenia revízií [4]
Plug-in – Zásuvný modul alebo prídavný modul je počítačový program, ktorý rozširuje funkcie iného programu alebo ho dopĺňa.

Príprava prehliadky

Úlohou v príprave prehliadky je identifikácia nového alebo zmeneného zdrojového kódu, ktorý ešte nebol podrobený prehliadke.

- Výber zdrojového kódu, ktorý je potrebné podrobiť prehliadke

Vstupy:
- Zdrojové kódy dostupné v Git repozitári
- Zoznam úloh v nástroji Jira

Podmienky:
- Nové zdrojové kódy v Git repozitári, ktoré sú skompilovatelné a spustiteľné.
- Dokumentácia k novým zdrojovým kódom, ktorú je možné nájsť v ***Dokumentácii k inžinierskemu dielu*.**

Výstupy
- Zdrojový kód, ktorý sa bude v ďalšom kroku prehliadať
- Vytvorená podúloha v nástroji Jira

Výber zdrojového kódu

V tomto kroku sa vyberie úloha ktorej výsledok sa podrobí prehliadke a preberú sa zdrojové kódy obsahujúce výsledok prehliadanej úlohy.

1. Vybrať úlohu alebo podúlohu zo zoznamu úloh pre aktuálny šprint z nástroja Jira , ktorej výstupom je zdrojový kód a hodnota jej stavu je „vyriešená“.
   1.1. Výber je závislý od priority úloh. Pokiaľ sú vyriešené viaceré úlohy, vždy sa na prehliadku určí tá, ktorá ma najvyššiu prioritu.
2. Vytvoriť podúlohu k úlohe určenej v kroku 1 s názvom: Prehliadka: ***názov úlohy určenej v kroku 1***. Spôsob akým sa vytvárajú úlohy v Jire je obsiahnutý v metodike plánovania. Vytvorená úloha bude mať stav „začatá“
3. Prebrať zdrojové kódy z Git repozitára.

Vykonanie prehliadky

Úlohou tohto kroku je prehliadnuť zdrojové kódy a s využitím nástroja PerConIK vytvoriť komentáre v zdrojovom kóde.

- Prehliadka zdrojového kódu
- Zaznamenanie nezrovnalostí

Vstupy

- Zdrojový kód, ktorý bol určený v **kapitole Výber zdrojového kódu**
- PerConIK: používateľská príručka [2]

Výstupy

- Okomentovaný zdrojový kód

Prehliadka zdrojového kódu

Na základe dokumentácie a analýzy prehliadaného kódu určiť, mieru splnenia požiadaviek a ohodnotiť zdrojový kód pomocou značiek ***ToDo***, ***Bug***, ***Smell***, ***CodeReview*** a ***Recommendation***, ktorých významy a atribúty sú definované v dokumente: ***PerConIK: používateľská príručka*** v kapitole Používanie značkovačov v IDE (str 3,4).

Príklad značky:

```
//Bug #Task(ID úlohy z Jira) #Solver(Riešiteľ ktorý ma nedostatok odstrániť)
#Priority(Závažnosť nedostatku) | prihlasovacie meno do Git repozitára 2013-11-
7T22:37:51.3290000Z
....//zdrojový kód\\....
//@< | prihlasovacie meno do Git repozitára 2013-11-17T22:37:51.3290000Z
```

1. Okomentovať kód pomocou preddefinovaných značiek v plug-in PerConIK
   1.1. Odhaliť odchýlky od štábnej kultúry písania zdrojového kódu následkom ktorých je znížená prehľadnosť zdrojového kódu. (značka ***CodeReview***)
      1.1.1. Ako podporný dokument slúži ***Metodika písania zdrojových kódov***.[5]
   1.2. Odhaliť chyby v zdrojovom kóde, ktoré majú priamy vplyv na funkcionalitu (značka ***Bug***, ***Smell***).
      1.2.1. V tomto kroku sa využíva ***Dokumentácia k inžinierskemu dielu***
   1.3. Odhaliť nedokončené časti zdrojového kódu. (značka ***ToDo***)
   1.4. Poukázať na možné vylepšenia zdrojového kódu, ktoré nemusia priamo súvisieť so splnením požiadaviek. (značka ***Recommendation***)
2. Doplniť ku každej z použitých značiek za jej atribúty opis odhaleného nedostatku. Tento opis musí obsahovať presné miesto výskytu nedostatku a vplyv na zdrojový kód.

Zaznamenanie nezrovnalostí

Pokiaľ sa vyskytnú nedostatky, ktoré nie je možné popísať s využitím značiek z kapitoly 1.5.1, budú tieto nedostatky popísané pomocou vlastných značiek.

1. Navrhnúť vlastnú značku, ktorá čo najlepšie popisuje identifikovaný nedostatok podľa návodu v dokumente: ***PerConIK: používateľská príručka*** v kapitole ***Vytváranie značiek*** (str. 5).
2. Po vytvorení vlastnej značky pokračuje proces v **krokom 2 v kapitole Prehliadka zdrojového kódu**.

Vytvorenie záznamu o vykonanej prehliadke

Záznam o vykonanej prehliadke slúži na informovanie autora zdrojového kódu o jeho nedostatkoch a potrebe odstránenia týchto nedostatkov z kódu.

- Vloženie okomentovaného zdrojového kódu do Git repozitára
- Zmena stavu podúlohy Prehliadka: ***názov úlohy určenej v kapitole Výber zdrojového kódu***
- Zmena stavu úlohy určenej v kroku 1.4.1 v závislosti od výsledku prehliadky

Vstupy

- Okomentovaný zdrojový kód

Výstupy

- Zmena stavu úlohy: ***Prehliadka: úloha určená v kapitole Výber zdrojového kódu*** na stav „vyriešená“.
- Okomentovaný zdrojový kód v Git repozitári.
- Zmena stavu úlohy: ***úloha určená v kapitole Výber zdrojového kódu*** na stav „zatvorená“ v prípade ak prehliadka neodhalila žiadne nedostatky.

1. Vložiť okomentovaný zdrojový kód do Git repozitára. Spôsob vkladania nových zdrojových kódov je opísaný v dokumente: ***Metodika manažmentu verziovania***.
2. Zmeniť stav úlohy ***Prehliadka: úloha určená v kapitole Výber zdrojového kódu*** v nástroji Jira na stav „vyriešená“.
3. Do pola „Popis práce“ v nástroji Jira napísať výsledok prehliadky.
   3.1. Výsledok prehliadky musí obsahovať názvy všetkých použitých značiek, ktoré boli použité pri prehliadke kódu.
   3.2. Ku každej značke musí byť priradený názov súboru zdrojového kódu v ktorom bola použitá.
   3.3. Ku každej značke musí byť popis, ktorý sa zhoduje s popisom v zdrojovom kóde.
4. Ak pri prehliadke nebol odhalený žiaden nedostatok.
   4.1. Zmeniť stav úlohy ***úloha určená v kapitole Výber zdrojového kódu*** na stav „zatvorená“.

Overenie opravy zdrojového kódu

Keďže cieľom prehliadky nie je odstránenie chýb, ale ich odhalenie a upozornenie na ich existenciu autora zdrojového kódu. Po vykonaní prehliadky s odhalenými nedostatkami prehliadku opätovne vykonať po odstránení odhalených nedostatkov.

Vstupy

- Opravený zdrojový kód, ktorý bol určený v **kapitole *Výber zdrojového kódu*** v Git repozitári.

Podmienky

- ***úloha určená v kapitole Výber zdrojového kódu*** nemá stav „zatvorená“

Výstupy

- Skontrolovaný zdrojový kód

1. Vykonať **kapitoly Vykonanie prehliadky a Vytvorenie záznamu o vykonanej prehliadke**
2. Pokiaľ kroky z **kapitoly Vykonanie prehliadky a Vytvorenie záznamu o vykonanej prehliadke** neodhalili žiaden nedostatok, prehliadka zdrojových kódov končí.
3. Pokiaľ boli odhalené ďalšie nedostatky opakujú sa kroky z **kapitoly Overenie opravy zdrojového kódu** dokiaľ nebudú všetky nedostatky odstránené.

Externé zdroje

[1] http://perconik.fiit.stuba.sk/
[2] http://sdrv.ms/1dRt8zi
[3] http://jira.fiit.stuba.sk/
[4] http://git-scm.com/
[5] http://labss2.fiit.stuba.sk/TeamProject/2013/team09is- si/downloads/Metodika%20pisania%20zdrojovych%20kodov.docx

10 Manažment komunikácie a ľudských zdrojov

10.1 Metodika komunikácie

Úvod

Cieľom tejto metodiky je definovať a usmerniť rôzny typy komunikácie členov tímového projektu. Na tieto účely je využívaný Facebook, konkrétne jeho služba Facebook Skupiny (ďalej len „skupina“), ktorá bola s týmto cieľom vytvorená a využívaná ešte pred začiatkom aktuálneho semestra. Metodika opisuje procesy používania ponúkaných funkcií skupiny za účelom efektívnej, konzistentnej a štruktúrovanej komunikácie.

Skupina umožňuje vytvárať príspevky, komentovať príspevky, sledovať príspevky, vytvárať hlasovania (otázky), rozosielať hromadné správy členom skupiny, skupinový chat, zdieľať obrázky alebo ľubovoľné súbory, s možnosťou vkladať revidované verzie súborov so zachovaním predchádzajúcich verzií a v neposlednom rade umožňuje notifikovanie členov či už prostredníctvom emailu, *push* notifikácií v mobile alebo priamo v internetovom prehliadači. Väčšina podstatných služieb je integrovaná priamo, alebo dostupná pomocou aplikácií, pre všetky rozšírené mobilné zariadenia. V rámci celej skupiny a teda v rámci celej komunikácie je možné vyhľadávať fulltextovo.

Použité pojmy

- **komunikácia** – všeobecná komunikácia medzi členmi tímu, zahŕňajúca rôzne oznámenia, pokyny, výzvy, upozornenia, požiadavky a pod.
- **príspevok** – (angl. *post, wall post*) – individuálny prvok facebook skupiny vytvorený členom skupiny, obsahujúci informáciu v podobe textu, obrázku alebo odkazu na internetovú stránku
- **komentár** – reakcia člena skupiny na ľubovoľný príspevok (zobrazené pod daným príspevkom)
- **JIRA** - nástroj na evidenciu a monitorovanie úloh na tímovom projekte

Roly

Rola	Popis
Člen tímu	Názov role hovorí sám za seba. Podstatné je, že z pohľadu komunikácie sa tím nerozlišuje na vedúceho, programátora alebo manažéra komunikácie. Všetci členovia sú rovnocenný (s výnimkou ďalších uvedených rolí v tejto metodike)
Zodpovedná osoba	Člen tímu, zodpovedný vykonanie konkrétnej úlohy
Iniciátor	Ľubovoľný člen tímu, ktorý potrebuje komunikovať s ľubovoľným členom, resp. členmi tímu

Tabuľka10.1 Roly

Všeobecné pokyny

Táto kapitola obsahuje pokyny, aplikovateľné na viaceré procesy metodiky, popísané v nasledujúcej kapitole:

- **Povinnosť označovať konkrétne mená** – V prípade každej komunikácie (popisovanej v tejto metodike), kedy člen skupiny spomína konkrétneho člena tímu, je povinné spomínanú osobu označiť v texte pomocou zadania znaku „@". Po zadaní tohto znaku, je potrebné napísať meno člena skupiny a stlačiť klávesu *enter*.

- **Povinnosť „lajkovať"" všetko čo je pre mňa určené** – Člen skupiny je povinný kliknúť na tlačidlo *Like*, umiestnené pod každým príspevkom (komentárom) v prípade, že je zadaný príspevok určený pre neho (t.j. aj keď je určený pre celý tím alebo podmnožinu členov tímu). „Lajknutím" člen dáva najavo, že oznam berie na vedomie, a súhlasí s ním. V prípade, že nesúhlasí, pridá k príspevku komentár s vyjadrením nesúhlasu.

- **Vytvorenie príspevku** – Niektoré z popisovaných procesov v tejto metodike obsahujú vytváranie príspevku. Príspevok sa vytvára v hlavnom okne skupiny (*obrázok 10.1*) napísaním požadovaného textu do textového poľa a stlačením tlačidla *Post*.

Obrázok 10.1 Vytvorenie príspevku

Procesy

V uvedenej tabuľke (*tabuľka 10.2*) je zoznam šiestich typov komunikácie. Prvé tri typy (č. 1 – 3) sa týkajú všeobecného, bližšie nešpecifikovaného druhu komunikácie, a druhé tri typy (č. 4 – 6) obsahujú špecifické typy komunikácií, týkajúce sa práce na tímovom projekte.

Prípady použitia konkrétneho procesu sú bližšie popísané priamo v procesoch spolu s podmienkami a výnimkami použitia. Procesy č. 2 a 3 sú takmer totožné, s výnimkou odlišnej priority komunikácie. V prípade, že daná informácia vyžaduje reakciu každého člena tímu do 12 hodín, zvolí iniciátor proces s vysokou prioritou (č. 3).

Č.	Názov	Kapitola
1	Komunikácia na člena alebo časť tímu	3.1
2	Komunikácia na všetkých členov tímu s nízkou prioritou	3.2
3	Komunikácia na všetkých členov tímu s vysokou prioritou	3.3
4	Oboznámenie o práci na projekte	3.4
5	Rozdeľovanie úloh na projekte	3.5
6	Spolupráca na prioritných úlohách	3.6

Tabuľka 10.2 Procesy komunikácie

Komunikácia na člena alebo časť tímu

Proces predpisuje prípad potreby naviazať ľubovoľnú, bližšie nešpecifikovanú komunikáciu na konkrétneho člena tímu, alebo na niektorých členov tímu. V prípade, že sa jedná o viacerých členov tímu, je na uvážení iniciátora správne odhadnúť, či je informácia naozaj až taká špecifická, a nie je vhodnejšie už priamo smerovať komunikáciu na celý tím, keďže informovanosť celého tímu je dôležitejšia, ako vznik prípadov, kedy sa menšia časť tímu dozvie niečo navyše.

Iniciátor komunikácie vytvorí príspevok (angl. *post*) a pomocou zadania znaku „@" na ľubovoľnom mieste v texte označí príslušného člena. Po zadaní tohto znaku, je potrebné napísať meno člena skupiny a stlačiť klávesu *enter*. Označený členovia sú následne notifikovaný o spomenutí ich mena vo vytvorenom príspevku.

Komunikácia na všetkých členov tímu s nízkou prioritou

Tento proces predpisuje prípad potreby naviazať komunikáciu na všetkých členov tímu, kedy sa jedná o informáciu s nízkou prioritou. Pokyny k určeniu priority sú uvedené v úvode tejto kapitoly.

Iniciátor komunikácie vytvorí príspevok (angl. *post*) a na začiatok príspevku zadá značku v presne uvedenom tvare „[TEAM]". Všetci členovia skupiny sú následne notifikovaný o pridaní príspevku v tímovej skupine.

Komunikácia na všetkých členov tímu s vysokou prioritou

Tento proces predpisuje prípad potreby naviazať komunikáciu na všetkých členov tímu, kedy sa jedná informáciu s vysokou prioritou. Pokyny k určeniu priority sú uvedené v úvode tejto kapitoly.

Iniciátor komunikácie vytvorí príspevok (angl. *post*) a na začiatok príspevku zadá značku v presne uvedenom tvare „[! TEAM !]". Všetci členovia skupiny sú následne notifikovaný o pridaní príspevku v tímovej skupine.

Oboznámenie o práci na projekte

V prípade kolaborácii na konkrétnej úlohe viacerých členov tímu, je člen, ktorý začína pracovať na tejto úlohe ako prvý, povinný vytvoriť príspevok v tvare „[WORK][ROBOCUP-TPxx] <popis_práce>", kde ROBOCUP-TPxx predstavuje identifikátor úlohy v systéme JIRA a <popis_práce> obsahuje informáciu o konkrétnej činnosti, na ktorej iniciátor tejto komunikácie ide pracovať. Ak sa nejedná o úlohu, ktorá je vytvorená aj v systéme JIRA, identifikátor úlohy sa nezadáva.

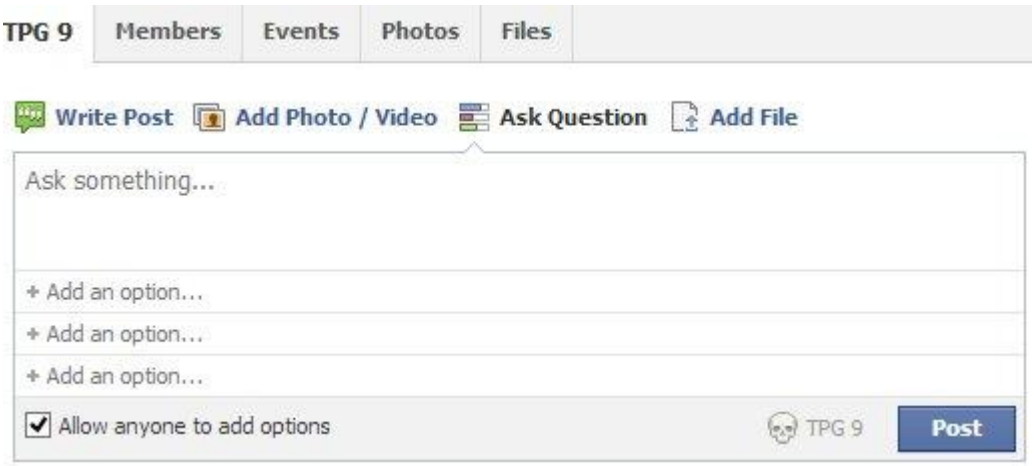
Ďalší členovia tímu sú povinní pred začatím akejkoľvek práce na kolaboratívnej úlohe tímového projektu skontrolovať, či sa nenachádza v skupine príspevok s identifikátorom danej úlohy. Ak nie, postupujú podľa pokynov v predchádzajúcom odstavci, ak áno, sú povinní pridať komentár k tomuto príspevku v tvare „[WORK] <popis_práce>", kde popis práce značí konkrétnu činnosť.

Po ukončení činnosti sa odporúča pridať komentár v tvare „[DONE] <popis_práce>". Nie je to však povinnosť. Rovnako popis práce nie je nutné uvádzať.

Tento proces je aplikovateľný aj na prípad, kedy sa nejedná o kolaboráciu na jednej úlohe. V prípade, že úloha člena tímu je závislá aj od iných úloh ostatných členov tímu, je tento člen povinný oznámiť začiatok a koniec svojej práce spomínaným postupom. **Rovnako sa odporúča tento postup použiť aj v prípade, ak úloha nie je vyslovene závislá od iných úloh ostatných členov tímu, ale vyžaduje z iných príčin oboznámenie ostatných o jej priebehu.**

Rozdeľovanie úloh na projekte

Tento proces pokrýva situáciu, kedy má tím zadanú kolaboratívnu úlohu, kde majú členovia tímu voľnosť pri zvolení si konkrétnej podúlohy, na ktorej budú pracovať. V tomto prípade je zodpovedná osoba za úlohu povinná vytvoriť v skupine otázku kliknutím na položku „Položiť otázku“ (angl. *Ask Question*) a ak je známe rozdelenie podúloh, je povinná ich zadať ako možnosť odpovede (*obrázok 10.2*). Popis otázky musí byť v tvare „[ROBOCUP-TPxx] <popis_úlohy>”. Otázku môže zadať namiesto zodpovednej osoby aj ľubovoľný člen tímu.



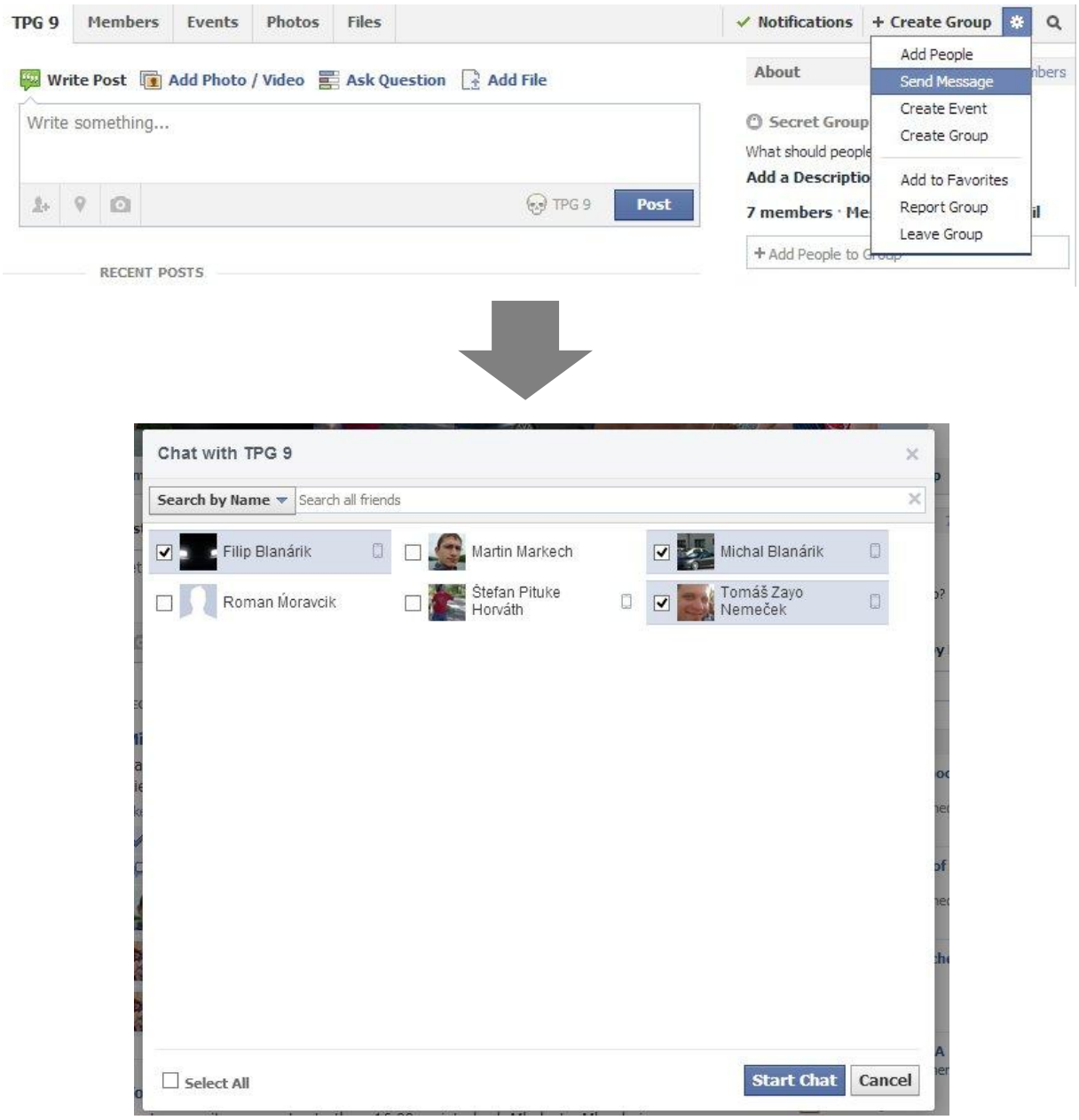
Obrázok 10.2 Vytvorenie otázky

Ostatní členovia tímu, ktorí sa zúčastňujú na kolaborácii, sú povinní označiť aspoň jednu z možností podúloh a tým sa zaväzujú túto podúlohu vykonať. V prípade, že sa požadovaná podúloha nenachádza v možnostiach, člen tímu túto podúlohu vytvorí a hneď aj označí.

Spolupráca na prioritných úlohách

V prípade, že tím (alebo časť tímu) spolupracuje na prioritnej úlohe, ktorá vyžaduje väčšiu mieru interakcie členov, vytváranie doteraz spomínaných príspevkov nie je vhodné. V takomto prípade je povinnosť použiť skupinový chat, respektíve skupinovú správu. Založenie skupinového chatu automaticky otvorí samostatné „chatovacie“ okno, avšak systém ukladá komunikáciu aj do klasických správ, označených ako konverzácia.

Iniciátor konverzácie založí skupinový chat v danej skupine kliknutím na symbol  a zvolením možnosti „Poslať správu (angl. *Send message*). Následne označí požadovaných členov tímu, alebo vyberie všetkých označením položky „Označ všetko“ (angl. *Select all*). Postup je znázornený na *obrázok 10.3*.



Obrázok 10.3 Vytvorenie skupinovej správy

Ak niektorí z členov zatvorí „chatovacie“ okno, opätovne ho otvorí vyhľadaním danej konverzácie v správach, kliknutím na symbol ⚙ a zvolením možnosti „Otvoriť chat“ (angl. *Open in chat*).

11 Manažment tvorby dokumentácie

11.1 Metodika dokumentovania

Vymedzenie obsahu

Účelom metodiky je stanoviť jednotný štýl tvorby jednotlivých dokumentov, zápisníc zo stretnutí tímu a celkovej dokumentácie vyvíjaného softvérového produktu, aby bolo možné minimalizovať nutný počet opravných zásahov do týchto dokumentov.

Tento dokument je určený pre členov vývojového tímu, ktorí píšu dokumenty a zápisnice a zmyslom metodiky je povinnosť každého člena tímu riadiť sa a dodržiavať pravidlá, ktoré sú v nej uvedené.

Zoznam nadväzujúcich metodík a dokumentov

Súvisiace Metodiky
Metodika plánovania

Dokumenty
Šablóna zápisnice
Šablóna dokumentácie k inžinierskemu dielu

Vymedzenie pojmov, vymedzenie skratiek

Scrum – spôsob riadenia projektu
Šprint – jeden časovo obmedzený cyklus iterácií v rámci projektu riadeného metodikou SCRUM
Scrum Master – Nie je to manažér tímu, ale člen tímu zodpovedný za riadenie rokovaní, ktoré prebiehajú medzi tímom a vedúcim projektu (Product Owner).
Scrum poker – metóda odhadu náročnosti úloh používaná v rámci Scrum. Každý člen tímu tajne zvolí číslo z Fibonacciho postupnosti, ktoré podľa jeho názoru najlepšie vystihuje zložitosť danej úlohy. Následne každý člen tímu ukáže zvolenú hodnotu a prezentuje svoj názor. Toto sa opakuje v iteráciách kým celý tím nedospeje k jednotnému číslu.

Použité softvérové nástroje

Na tvorbu finálnych dokumentov sa používa Microsoft Office Word 2007 alebo novší. Pri priebežných verziách je možné použiť nástroj Google Docs. Šablóny sú vytvorené pomocou Microsoft Office Word 2007. Na zdieľanie dokumentov sa používa Google Drive.

Zápisnica zo stretnutí

Zápisnica je dokument, ktorý sa vytvára na základe poznámok zo stretnutia tímu s vedúcim tímového projektu a slúži na uchovanie záznamu o stretnutí, rozšírenie informácií aj medzi členov tímu, ktorí sa stretnutia z nejakých dôvodov nezúčastnili a v neskoršom štádiu projektu aj ako zdroj na vyhľadanie poznatkov a informácií, ktoré sa neuchovali v pamäti jednotlivých členov vývojového tímu.

Pre tvorbu zápisnice je k dispozícii samostatná šablóna. Každá zápisnica z oficiálneho stretnutia tímu musí byť vytvorená a zverejnená na webovom sídle tímu najneskôr do 24h od ukončenia stretnutia. Zápisnicu zverejňuje vždy zapisovateľ, ktorý danú zápisnicu vytvoril.

Vzorový príklad zápisnice je uvedený v kapitole Príloha – Vzorové príklady dokumentov.

Obsah zápisnice

Každá zápisnica musí obsahovať nasledujúce údaje v bodoch podľa uvedeného poradia:

1) Názov dokumentu

Hlavička dokumentu

2) Dátum: dátum, v ktorom sa stretnutie uskutočnilo vo formáte deň.mesiac.rok.
3) Miestnosť: názov miestnosti v rámci budovy fakulty, kde sa konalo stretnutie.
4) Prítomní: zoznam mien členov tímu, ktorí sa zúčastnili daného stretnutia a meno pedagóga, ktorý je v úlohe vedúceho tímového projektu (Procut Owner).
5) Stretnutie viedol: uvedené meno Scrum Master-a, ktorý viedol stretnutie.
6) Zápis: uvedené meno člena tímu, ktorý bol zvolený na predchádzajúcom stretnutí za zapisovateľa a vykonal zápis z tohto stretnutia.
7) Zvolený nasledujúci zapisovateľ: uvedené meno nasledujúceho zapisovateľa, ktorý bol zvolený tímom v rámci tohto stretnutia.

Jadro dokumentu

8) Téma stretnutia – zhrnutie, čo bolo predmetom daného stretnutia.

9) Vyhodnotenie úloh z predchádzajúceho stretnutia / šprintu – kapitola obsahuje údaje k jednotlivým úlohám zapísané v tabuľke, ktoré vyplynuli z predchádzajúcich stretnutí a sú naplánované v rámci aktuálneho šprintu.

   Tabuľka obsahuje tieto stĺpce:

   - **Číslo úlohy** – identifikačné číslo úlohy, ktoré je zhodné s číslom uvedeným v systéme manažovania projektu (JIRA).
   - **Zhrnutie úlohy** – názov úlohy, ktorý stručne vystihuje jej podstatu.

- **Riešiteľ** – meno člena tímu, ktorý bol určený na vyriešenie danej úlohy, alebo v prípade, že úlohu rieši viac členov tímu alebo celý tím, hodnota v stĺpci pre danú úlohu bude *Celý tím*.
- **Zodpovedný** – meno člena tímu, ktorý je zodpovedný za dohľad nad vyriešením danej úlohy. Kontroluje či sa postupuje pri riešení danej úlohy, a či je možné dodržať termín dokončenia. Riešiteľ aj zodpovedný môže byť ten istý člen tímu.
- **Stav úlohy** - hodnota, ktorá hovorí o tom, či je daná úloha vyriešená alebo nie. V stĺpci môžu byť tieto hodnoty:
  - *Splnené* – ak je práca na úlohe ukončená a je vytvorený príslušný dokument, ktorý popisuje čo sa v rámci danej úlohy vykonalo.
  - *Nesplnené* – ak je daná úloha v štádiu rozpracovania alebo ešte na nej nebola zahájená práca.

Táto časť dokumentu sa musí nachádzať vo všetkých zápisniciach zo stretnutí okrem prvej zápisnice v rámci projektu, kedy ešte nie sú pred stretnutím riešené žiadne úlohy.

10) Opis stretnutia – najdôležitejšia časť zápisnice. Obsahuje záznam o všetkých bodoch, ktoré sa preberali v rámci stretnutia. Jednotlivé body musia byť opísané do takej miery, aby bolo možné po ich prečítaní jednoznačne určiť kto sa k danej téme vyjadroval, čo bolo obsahom bodu a aký je záver diskusie k danému bodu.

11) Úlohy do nasledujúceho stretnutia / konca šprintu – kapitola obsahuje údaje k jednotlivým úlohám zapísané v tabuľke, ktoré vyplynuli zo stretnutia a sú naplánované v rámci aktuálneho šprintu.

Tabuľka obsahuje tieto stĺpce:

- **Číslo úlohy** – rovnako ako v bode 9.
- **Zhrnutie úlohy** – rovnako ako v bode 9.
- **Riešiteľ** – rovnako ako v bode 9.
- **Zodpovedný** – rovnako ako v bode 9.
- **Body** – číselný odhad náročnosti danej úlohy, ktorý sa používa v rámci metodiky Scrum. Hodnota je určená v rámci stretnutia pri Scrum pokri.
- **Čas –** odhad časovej náročnosti danej úlohy, môže, ale nemusí sa zhodovať s hodnotou uvedenou v stĺpci Body.

Táto časť dokumentu sa musí nachádzať v zápisniciach zo stretnutí, z ktorých vyplynuli nové úlohy na vyriešenie v rámci aktuálneho šprintu. Najčastejšie sú to stretnutia, na ktorých sa zahajuje nový šprint, ale v niektorých prípadoch je možné, že aj v priebehu šprintov sa vyskytnú nové úlohy.

12) Popis k úlohám – ak naplánované úlohy nie sú dostatočne opísané v rámci kapitoly *10 Opis stretnutia*, tak sa k nim v tejto kapitole uvedú doplňujúce informácie.

13) Poznámky – obyčajne prázdna kapitola, ale v prípade, že sa vyskytla nejaká skutočnosť, ktorá priamo nespadá do žiadnej z uvedených kapitol a je dôležité túto informáciu rozšíriť medzi všetkých členov tímu alebo uchovať o nej záznam, táto skutočnosť sa sem uvedie.

Formátovanie zápisnice

- Názov – písmo Arial; veľkosť písma 18 pt; Bold; zarovnanie podľa okraja; dolné orámovanie (jednoduché, šírka čiary 0,5 pt); medzera pred 12 pt; medzera za 12 pt; riadkovanie 1,5
- Dátum, Miestnosť, Prítomní, Stretnutie viedol, Zápis, Zvolený nasledujúci zapisovateľ – písmo Times New Roman; veľkosť písma 12 pt; Bold; zarovnanie podľa okraja; medzera za 10 pt; riadkovanie 1,15
- Téma stretnutia, Vyhodnotenie úloh z predchádzajúceho stretnutia / šprintu, Opis stretnutia, Úlohy do nasledujúceho stretnutia / konca šprintu, Popis k úlohám, Poznámky – písmo Times New Roman; veľkosť písma 14 pt; Bold; zarovnanie podľa okraja; medzera pred 18 pt; medzera za 6 pt; riadkovanie 1,5
- Text v rámci kapitol – písmo Times New Roman; veľkosť písma 12 pt; zarovnanie podľa okraja; riadkovanie 1,15

Celková dokumentácia vyvíjaného softvérového produktu

Dokumentácia sa vytvára priebežne počas celého trvania projektu. Dokumentácia za príslušný šprint sa zverejňuje vždy najneskôr týždeň od konca šprintu a v kontrolných bodoch sa odovzdáva vždy celková dokumentácia inžinierskeho diela aj riadenia projektu obsahujúca všetky doteraz vytvorené šprintové dokumentácie vrátane dokumentácie práve skončeného šprintu.

Typy dokumentov

Dokumentácia k inžinierskemu dielu – celková dokumentácia, ktorá pozostáva z dokumentácií k jednotlivým šprintom.
Dokumentácia k riadeniu projektu – obsahuje zoznam kompetencií tímu, zápisnice zo stretnutí, metodiky, preberacie protokoly a všetky skutočnosti spojené s riadením tímového projektu
Dokument k jednotlivej úlohe – zachytáva analýzu, návrh a implementáciu danej úlohy. Spojením viacerých dokumentov k úlohám v rámci jedného šprintu vznikne dokumentácia k šprintu.

Obsah dokumentácie

1) Titulná strana – musí ju obsahovať každý dokument a je tvorená týmito bodmi:
    - Názov univerzity a fakulty
    - Adresa fakulty
    - Názov predmetu
    - Téma projektu
    - Názov dokumentu / názov úlohy v rámci šprintu
    - Mená autorov dokumentu / členov tímu
    - Číslo a názov tímu
    - Meno vedúceho projektu
    - Ročník
    - Akademický rok a semester
    - Mailový kontakt na tím
2) Obsah – kapitola je uvedená len v dokumentácií k inžinierskemu dielu a v dokumentácií k riadeniu projektu. Automaticky generovaný obsah pomocou nástroja v Microsoft Word.
3) Samotný text dokumentu – závisí od typu dokumentu, obsahuje všetky relevantné údaje k predmetu dokumentu.

Formátovanie dokumentácie

- Titulná strana – formátovanie podľa ukážky v kapitole *Príloha B – Vzorové príklady dokumentov.*
- Obsah – obsah dokumentu, v ktorom sú uvedené nadpisy úrovne 1, 2 a 3
- Nadpis 1 – písmo Arial; veľkosť písma 18 pt; Bold; zarovnanie podľa okraja; dolné orámovanie (jednoduché, šírka čiary 0,5 pt); medzera pred 12 pt; medzera za 12 pt;

riadkovanie 1,5; viacúrovňové číslovanie (úroveň 1) , štýl číslovania 1, 2, 3, ...; zlom strany pred odsekom

- Nadpis 2 – písmo Arial; veľkosť písma 14 pt; Bold; zarovnanie podľa okraja; medzera pred 18 pt; medzera za 6 pt; riadkovanie 1,5; viacúrovňové číslovanie (úroveň 2) , štýl číslovania 1, 2, 3, ...;
- Nadpis 3 – písmo Arial; veľkosť písma 14 pt; Bold; zarovnanie podľa okraja; medzera pred 9 pt; medzera za 3 pt; riadkovanie 1,5
- Podnadpis – písmo Times New Roman; veľkosť písma 12 pt; Bold; podčiarknutie; zarovnanie podľa okraja; medzera za 6 pt; riadkovanie 1,15
- Text v rámci kapitol – písmo Times New Roman; veľkosť písma 12 pt; zarovnanie podľa okraja; riadkovanie 1,15
- Zvýraznené časti v rámci textu – formátovanie textu rovnaké ako Text v rámci kapitol, s použitím podčiarknutia, kurzívy alebo bold-u. V prípade použitia zvýraznenia je potrebné dodržať rovnaké zvýrazňovanie v rámci jednej kapitoly.

12 Ďalšie metodiky

12.1 Metodika práce so softvérovým návrhom

Úvod

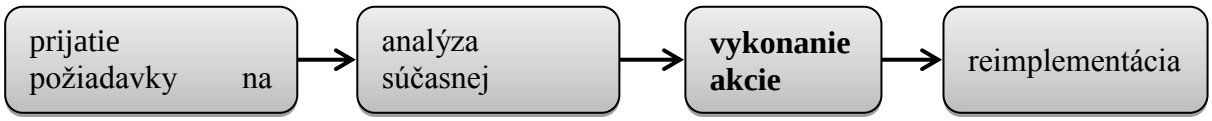
Metodika práce so softvérovým návrhom hovorí o spôsoboch, ktoré treba voliť pri práci s architektúrou softvéru zapísanou v diagrame tried pomocou UML. Definuje symboly a pravidlá definovania entít a popisuje, akým spôsobom je treba pridávať, odoberať, meniť a presúvať artefakty v softvérovom návrhu tak, aby sa čo najviac eliminovali riziká vzhľadom na funkčnosť softvéru.

Dôvod vzniku metodiky

Táto metodika vznikla z dôvodu potrieb tímového projektu. Tím dostal za úlohu zvýšiť modularitu softvéru, refaktorovať kód a doplniť softvér o nové vlastnosti. Na dosiahnutie týchto cieľov je potrebné prerobiť veľkú časť návrhu, pričom celý vývoj prebieha agilnou metódou SCRUM a vývoja sa zúčastňujú 2 tímy, ktoré musia úzko spolupracovať. Pri zmene softvérového modelu tak vzniká riziko poškodenia konzistencie architektúry z dôvodu neporozumenia notácií a nesprávnej úpravy softvérového návrhu. Toto riziko je možné znížiť dodržaním postupov opísaných v tejto metodike.

Ohraničenie metodiky

Metodiku môžeme formálne ohraničiť nasledujúcim diagramom, pričom akciou rozumieme vykonanie zmeny v diagrame tried:



Obrázok 12.1 formálne ohraničenie metodiky

Vstup: vyšpecifikovaná požiadavka, v prípade existujúcej architektúry hotová analýza

Výstup: nový alebo upravený diagram tried
Využitie tejto metodiky začína tam, kde je nutné urobiť zásah do architektúry. Spravidla tomu predchádza prijatie požiadavky na zmenu a v prípade existujúcej architektúry aj jej analýza. Následne je vykonaná akcií. Po ukončení je nutné zmeny reimplementovať a otestovať.

Pre koho je metodika určená

Metodika je určená pre člena tímového projektu, ktorý bude upravovať architektúru.

Pojmy

- Návrh – budeme rozumieť ako diagram tried popisujúci architektúru

- Artefakt – ľubovoľný objekt alebo proces, v našej terminológii objekt, komponent, trieda, štruktúra, algoritmus
- Komponent – unifikovaná časť systému, ktorá obsahuje ucelenú funkcionalitu
- UML (Unified Modeling Language) – štandardizovaný modelovací jazyk na záznam, vizualizáciu a dokumentáciu artefaktov prevažne softvérových systémov

Súvisiace metodiky

Úprava architektúry je riziková a je vhodné zvážiť použitie nasledujúcich metodík:

- Požiadavky na zmenu
- Evidencia úloh
- Údržba softvéru
- Písanie zdrojového kódu

Metodika pre prácu s architektúrou

Modifikácia architektúry úzko súvisí zo zdrojovým kódom softvéru. Keď sa zmení architektúra, zmena sa ihneď prejaví na zdrojovom kóde, ktorý treba reimplementovať podľa nových vzťahov. Často robíme zmeny v rozsiahlom návrhu, kde je vykonaná akcia pre element vo viacerých vzťahoch k iným elementom. Pri takomto elemente je nutné analyzovať riziká poškodenia funkčnosti vzhľadom na všetky elementy vo vzťahu a prípadne aplikovať viac metodík.

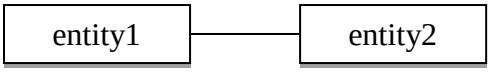
Pri práci s architektúrou identifikujeme tieto akcie:

- Tvorba novej architektúry
- Pridávanie nových elementov
- Úprava existujúcich elementov
- Odstránenie elementov

Architekrúru budeme vytvárať a modifikovať v softvéri Software Ideas Modeller, pričom budeme dodržiavať konvencie UML. Použitím konvencií sa vytvárajú na prvý pohľad typovo odlíšiteľné relácie medzi artefaktami.

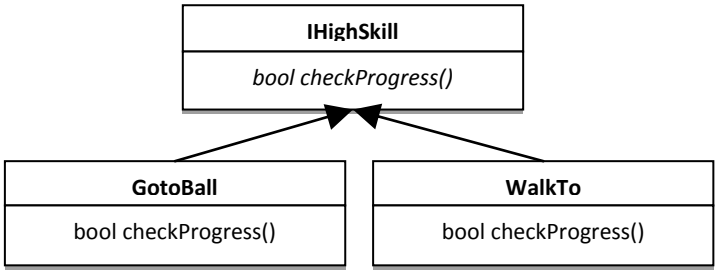
Budú použité tieto UML notácie:

- **Asociácia** – ak budeme definovať nejaký generický vzťah medzi entitami



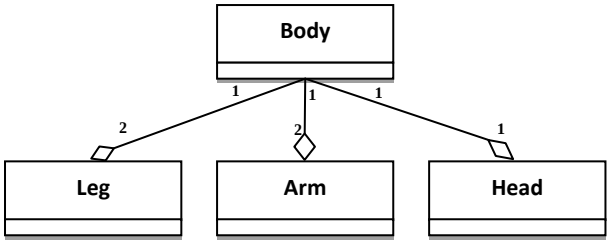
Obrázok 12.2 asociácia

- **Generalizácia** – ak budeme reprezentovať vzťah dedenia vlastností entít



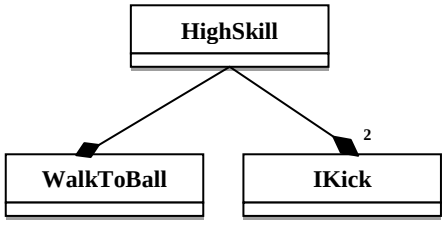
Obrázok 12.3 generalizácia

- **Kompozícia** – ak budeme reprezentovať vzťah, keď jedna entita je tvorená inými, bez ktorých by neposkytovala správne svoju funkcionalitu



Obrázok 12.4 kompozícia

- **Agregácia** – ak budeme reprezentovať vzťah, kde jedna entita referencuje (vyžaduje prístup) k inej



Obrázok 12.5 agregácia

Na označenie viditeľnosti atribútov triedy budú použité:

- „+“ pre verejný atribút (public)
- „#“ pre chránený atribút (protected)
- „-“ pre súkromný atribút (private)

Vykonanie akcie v prípade artefaktu v asociačnom vzťahu

Vykonanie akcie pre tento element návrhu je príliš všeobecné, čo vyplýva zo samotnej definície asociácie. Metodika na to nevie zadefinovať presný postup, ale vie navrhnúť spôsob analýzy rizík pri manipulácii. Treba mať na mysli, že element v asociácii je v nejakom vzťahu k inému elementu. Zmena tohto elementu ovplyvní aj všetky elementy v relácii, tým pádom

sa treba vždy zamyslieť nad rizikom, ktoré môže tohto manipuláciou vzniknúť. Až po identifikácii rizík a dôkladnom zvážení následkov je možné s týmito elementmi operovať. Treba prihliadnuť aj na možnosť, kde asociácia nie je ohraničená a manipulácia môže ovplyvniť aj elementy na inej architektonickej úrovni.

Tvorba nového návrhu

Je to takmer bezriziková akcia. Ak tvoríme nový návrh softvéru, jednoducho definujeme nové vzťahy medzi artefaktami s dodržaním predchádzajúcich notácií. Ak máme vytvorenú architektúru, takto navrhnutý softvér môžeme posunúť na implementáciu.

Pridávanie nových elementov

Pridanie elementu do generalizácie

1. Pridáme element do návrhu a vytvoríme generalizáciu s požadovanými artefaktami.
2. Vytvoríme požiadavku na pridanie artefaktu do kódu

Ak pridávame element generický, pridávame ho v zmysle nadtriedy v programovom kóde. Riziko z hľadiska poškodenia funkcionality je nulové nakoľko nový artefakt zatiaľ nie je nikde použitý.
Ak pridávame element ako generalizovaný, pridávame ho v zmysle podtriedy, ktorá dedí vlastnosti nadtriedy. Riziko z hľadiska poškodenia funkcionality je nulové, jedná sa o pridanie novej vlastnosti, ktorá dodržiava isté pravidlá ktoré v súčasnom návrhu existujú a sú zaintegrované.

Pridanie elementu do kompozície

1. Pridáme element do návrhu a vytvoríme kompozíciu s požadovanými artefaktmi.
2. Identifikujeme rizikové časti návrhu, ktoré táto akcia ovplyvní.
3. Vytvoríme požiadavku na pridanie artefaktu do kódu a integráciu do elementov v relácii

Pridanie elementu do kompozície realizujeme pravdepodobne za účelom vylepšenia pôvodného artefaktu. Táto akcia vyžaduje zaintegrovanie, pričom treba identifikovať riziká poškodenia funkcionality voči elementom v relácii tohto pôvodného elementu nakoľko sa jedná o zásah do existujúcej implementácie.

Pridanie elementu do agregácie

1. Pridáme element do návrhu a vytvoríme agregáciu s požadovanými artefaktmi.
2. Identifikujeme rizikové časti návrhu, ktoré táto akcia ovplyvní.
3. Vytvoríme požiadavku na pridanie artefaktu do kódu a integráciu do elementov v relácii

Pridanie elementu do agregácie znamená využitie nového artefaktu. To vyžaduje zaintegrovanie podobne ako pri kompozícii vrátane rizík.

Úprava existujúcich elementov

1. Upravíme element v návrhu
2. Ak upravujeme generický element, identifikujeme potrebu úpravy generalizovaného elementu a referencií, ak je potrebné, vykonáme úpravu
3. Vytvoríme požiadavku na reimplementáciu

Úprava elementu v generalizácii

Ak upravujeme generalizovaný element , riziko poškodenia funkcionality je nulové.
Ak upravujeme generický element, je potrebné analyzovať, prípadne vykonať zmeny na generalizovaných elementoch. Tiež je dôležité identifikovať riziká na všetkých miestach softvéru, kde je tento element referencovaný.
Ak sa mení rozhranie generického elementu, je nutné upraviť všetky generalizované elementy a tiež všetky referencie.

Úprava elementu v kompozícii

1. Upravíme element v návrhu
2. Ak upravujeme kompozičný element, zistiť potrebu úpravy na koreni
3. Vytvoríme požiadavku na reimplementáciu

Ak upravujeme koreň kompozície, z hľadiska kompozície to nie je riziková akcia. Ak upravujeme kompozitový element, pravdepodobne vzniká riziko poškodenia funkčnosti na koreni. Toto riziko treba identifikovať a prípadne upraviť použitie na koreni.

Úprava elementu v agregácií

1. Upravíme element v návrhu
2. Ak upravujeme agregovaný element, je nutné identifikovať riziká na všetkých referencovaných miestach a navrhnúť spôsob úpravy
3. Vytvoríme požiadavku na reimpementáciu

Ak upravujeme koreň agregácie, z hľadiska agregácie akcia nepredstavuje žiadne riziko. Ak upravujeme agregovaný element, predstavuje to rozsiahlejší problém, a to na všetkých miestach, kde upravovaný element referencujeme. Na týchto miestach treba vykonať analýzu rizík poškodenia funkčnosti a navrhnúť úpravy. Je pravdepodobné, že táto akcia bude vyžadovať aj zmenu návrhu ak napr. odstránime časť rozhrania.

Odstránenie elementov

Odstránenie elementu z generalizácie

1. Odstránime element z návrhu.
2. Vytvoríme požiadavku na zmazanie artefaktu z kódu
3. V prípade odstránenia generalizovaného elementu požadujeme tiež nahradenie všetkých referencií generického artefaktu na posledný ostávajúci generalizovaný.

Ak odstraňujeme generalizovaný element, vykonáme túto akciu z nulovým rizikom v prípade dobre vytvoreného návrhu a implementácie. Avšak prakticky sa často stáva, že implementácia používajúca generický element tohto generalizovaného elementu ma v sebe napevno spravený „case“, kde rozhoduje cez všetky generalizované objekty. Toto riziko však odhalí kompilátor, teda nie je nutné nevyhnutne evidovať riziko do požiadavky na zmenu kódu.
Generický element budeme odstraňovať len v prípade, ak má pod sebou práve jeden generalizovaný element.

Odstránenie elementu z kompozície

1. Odstránime element z návrhu.
2. Identifikujeme riziká spôsobené vykonaním tejto akcie, ktoré zahrnieme do požiadavky.
3. Vytvoríme požiadavku na zmazanie artefaktu z kódu a požadujeme reimplementáciu závislých súčastí

Ak odstraňujeme koreň kompozície, z hľadiska kompozitových elementov je to bezpečná akcia. Ak odstraňujeme kompozitový element, je treba ho vymazať z implementácie a identifikovať riziká, ktoré môžu nastať odstránením tejto funkcionality.

Odstránenie elementu z agregácie

1. Odstránime element z návrhu
2. Identifikujeme závislosti a v prípade potreby vykonáme zmenu dotyčného elementu
3. Vytvoríme požiadavku na zmazanie elementu z agregácie, kde uvedieme aj zoznam nutných zmien na iných elementoch, ktoré treba reimplementovať.

Ak odstraňujeme koreň agregácie, na agregovaný element to nemá žiaden vplyv, čiže bezriziková akcia. Ak však odstraňujeme agregovaný element, môže to mať rozsiahlejšie následky. V tomto prípade treba identifikovať riziká aj na ďalších architektonických stupňoch. Napr. ak je takýto element ako parameter, znamená to prešpecifikovanie daného artefaktu v návrhu, čo má za následok vykonanie akcie zmeny.

12.2 Metodika písania zdrojových kódov

Úvod

Táto kapitola slúži ako smernica pre správne písanie zdrojových kódov v projekte RoboCup.

Je veľmi dôležité, aby boli dodržiavané isté pravidlá a dohody slúžiace na zosúladenie a sprehľadnenie zdrojových kódov v projekte. Vďaka týmto pravidlám je výsledný produkt práce ľahko pochopiteľný aj pre nezainteresovaných programátorov a pomáha k rýchlejšiemu a lepšiemu zorientovaniu sa v kóde.

Pri návrhu týchto smerníc sme vychádzali zo základných pravidiel používaných pri vývoji aplikácií v jazyku JAVA vytvorených spoločnosťou SunSystems. Následne sme tieto základné pravidlá obohatili pravidlami, ktoré boli doplnené predchádzajúcimi riešiteľmi projektu ROBOCUP.

JavaDoc komentáre

Javadoc je generovaná dokumentácia zo zdrojových kódov pre aplikačné programové rozhranie (API) vo formáte HTML. Tento formát je využívaný na prepojenie jednotlivých dokumentov tvoriacich JavaDoc.

JavaDoc komentáre pre triedy

Nad každou triedou je potrebné vytvoriť JavaDoc komentár v minimálnej forme:

- Názov súboru triedy
- Popis triedy (cca. 3 riadky)
- Anotácia Titul pre názov projektu
- Anotácia Autor vo formáte loginu v AIS
- Anotácia Autor s názvom tímu

Príklad:

```
/**
 * Settings.java
 *
 * Encapsulatesglobalsettingsthatalterthebehaviourofthecodethroughout
 * theentireproject. Canchange default settingsforthe game. Usuallyis
 * meant to be set in ./scripts/config/settings.rb.
 *
 * @Title      Jim
 * @author     xnemecek
 * @author     gitmen
 */
publicfinalclassSettings { ...
```

JavaDoc komentáre pre metódy

Nad každou metódou je potrebné vytvoriť JavaDoc komentár v minimálnej forme:

- Popis metódy (cca 3-5 riadkov)
- Anotácia Autor vo formáte loginu v AIS
- Anotácia Autor s názvom tímu
- Parametre metódy (ak sú): @paramnázovParametra – popis parametra
- Návratová hodnota metódy: @returnnázovNávratovejHodnoty – popis
- Výnimky, ktoré môže metóda vyhadzovať (ak sú): @throwsNázovVýnimky - popis

Príklad:

```
/**
 * Checksemployees and findoutwhois on vacation.
*
 *  @authorxnemecek
 *  @authorgitmen
*
 * @paramemployeesToCheck–employeesneeded to bechecked.
 * @paramemployeesOnVacation–employeeswho are on vacation.
 * @returnresultEmployees – employees on vacationfilteredoutofemployeesToCheck.
 * @throwsIllegalArgumentException – ifany parameter isnull.
 */
privatestaticArrayListgetCheckedEmployeesOnVacation(ArrayListemployeesToCheck,
ArrayListemployeesOnVacation) throwsIllegalArgumentException { ...
```

Komentáre pre premenné

Nad premennú je potrebné uviesť stručnú definíciu na jednoznačné určenie významu a cieľového použitia.

```
/**
 * Nameof  agent's team
 */
publicstaticString teamName;
```

Všeobecné komentáre

Poznámky v kóde

Ostatné komentáre, ktoré sa neviažu na definíciu premennej, metódy, triedy a pod. môžu byť v základnom tvare// *something*pre jednoriadkové komentáre alebo/* *something* */pre viac riadkové komentáre.

Komentáre by sa mali používať iba v nutných prípadoch, kedy kód nie je možné pre jeho komplexnosť zjednodušiť a je nutné na jeho ľahšie pochopenie členenie pomocou komentárov.

Zakomentovaný kód

Zakomentovaný kód je možné použiť pre krátkodobú osobnú potrebuv lokálnom repozitári. Pred ukladaním zdrojových súborov do repozitára je nutné, aby bol zakomentovaný kód odstránený, keďže zakomentovaný kód nemá žiadnu funkčnú hodnotu a teda je zbytočný.

TODO komentár

V prípade, že si autor kódu v určitej časti myslí, že je vhodné neskôr doplniť svoj kód, ale z vážneho dôvodu to nemôže spraviť hneď, tak je vhodné použiť TODO komentár. K týmto komentárom je potrebné sa pravidelne vracať a dať si záležať, aby bola funkcionalita, na ktorú poukazujú doplnená a komentár následne odstránený. Výhodou TODO komentárov je možnosť ľahkého vyhľadania pomocou vývojového prostredia.

Obsah TODO komentára bude pozostávať z mena autora (tvar loginu v AIS), názvu tímu a popisu :

```
/* TODOAddspecialmove #2 afterdiscussionwithcolleagues.
 * @authorxnemecek, gitmen
 */
```

TODO komentár neberie do úvahy značky začínajúce zavináčom, preto môžeme použiť vyššie uvedenú skrátenú formu zápisu autora a tímu do TODO komentára.

Zmeny v kóde

Niektoré tímy si v kóde komentármi označujú časti kódu, ktoré zmenili. Táto praktika je zbytočná, keďže na podobné účely slúži JavaDoc a dokumentácia k inžinierskemu dielu. V rámci trvania projektu sa dá na tento účel taktiež použiť verziovací systém v prepojení s vývojovým prostredím.

Štruktúra triedy

V tejto kapitole popíšeme hierarchiu zápisu premenných a metód:

1. Konštanty - v poradí public, protected, private

Príklad:

```
publicstaticfinalString CONST ="konstatna";
protectedstaticfinalint FINAL_SCORE =0;
privatestaticfinallong MAX_COUNT =15;
```

2. Statické premenné - v poradí public, protected, private

Príklad:

```
publicstaticString STATIC ="statik";
protectedstaticint numOfGoals =0;
publicstaticlong numOfPlayers =15;
```

3. Atribúty inštancií - je dôležité, aby tieto premenné mali modifikátor prístupu private a pristupovalo sa k nim prostredníctvom getterov a setterov.
4. Konštruktory
5. Metódy s modifikátorom public- používať iba pre metódy volané mimo ich balík
6. Metódy s modifikátorom protected - používať iba pre metódy volané v rámci balíka
7. Metódy s modifikátorom private - Umiestňovať pod metódy ku ktorým prislúchajú v poradí, v akom ich metóda využíva.
8.

Konvencia kódu

Je vhodné používať klasickú java konvenciu, keďže základy tejto konvencie sú všeobecne známe. V používaných vývojových nástrojoch je táto konvencia štandardne nastavená.

Príklad:

```
publicclassClassA {

        publicintnumber = 1;
        privateStringletters[] = newString[]{"A", "B"};

        publicvoiddefineCharacter(Stringcharacter, int index) {
                if (character == null) {
                        character = "a";
                } elseif (character.length() == 0) {
                        character = "empty";
                } else {
                        index++;
                }
        }
}
```

Názvy premenných a metód

Pri názvoch je vhodné používať výrazy, ktoré vystihujú význam použitia.
Ďalej je vhodné používať metódu CamelCase:

- Lower-CamelCase pre názvy metód a premenných, kedy je prvé písmeno malé.

  Príklad:

  ```
  publicabstractvoiddefineCharacter(Stringcharacter, int index)
  ```

- Upper-CamelCase pre názvy tried a rozhraní

Príklad:

```
publicclassAgentAtributes{
```

Malé metódy a triedy

Metódy a triedy by mali byť krátke, aby sme sa vyhli zbytočnému zahlteniu rozsiahlou funkcionalitou a strate prvotného využitia.
Metódy je vhodné písať tak, aby vždy vykonávali iba jednu atomickú operáciu.

Príklad:

```
publicintgetNearestPlaymate(Worldworld) {
        List<Agent>players = getSimulation().getWorld().getPlayers();
        AgentnearestAgent = players.get(0);
        for (Agent a : players) {
                if (a.getTeam.getName.equals(MYTEAMNAME)
                                && (a.getDistance() <me.getDistance())) {
                        nearestAgent = a;
                }
        }
        returnnearestAgent;
}
```

Zdrojové kódy

Zdrojové kódy je vhodné písať v angličtine a bez diakritiky. Vďaka tomuto sú zdrojové kódy jednoducho pochopiteľné aj zahraničným kolaborantom na projektoch.

Ďalšou dôležitou časťou je kódovanie súborov. Vhodným kódovaním pre rôzne operačné systémy je kódovanie UTF-8.

Zdrojové kódy v jazyku Ruby

Vzhľadom nato ,že kódy v jazyku Ruby tvorili veľkú časť projektu, rozhodli sme sa venovať kapitolu aj tomuto jazyku. Smernice aj v tomto prípade majú pomôcť pri efektívnejšej práci v projekte.

Ako zdroj pre tieto smernice sme vybrali dokument na stránke GitHub, ktorý sa nachádza na https://github.com/bbatsov/ruby-style-guide.

A Záznamy zo stretnutí

V tejto prílohe sa nachádzajú zápisnice z oficiálnych stretnutí tímu s vedúcim projektu.

A-1. Zápis z 1. Stretnutia tímu č. 9

Dátum: 2.10.2013

Miestnosť: Jobsové softvérové štúdio (lab 1.31) (FIIT-STU)

Prítomní:

Pedagóg :	Ing. Ivan Kapustík
Členovia tímu:	Bc. Filip Blanárik, Bc. Michal Blanárik, Bc. Štefan Horváth Bc. Štefan Linner, Bc. Martin Markech, Bc. Roman Moravčík Bc. Tomáš Nemeček

Stretnutie viedol: Ing. Ivan Kapustík

Zápis: Bc. Martin Markech

Zvolený nasledovný zapisovateľ: Bc. Filip Blanárik

Téma stretnutia

Oboznámenie sa s RoboCupom, organizačné pokyny

Opis stretnutia

1. Úvodom stretnutia sme sa oboznámili s povinnosťami, ktoré nás čakajú. Dozvedeli sme sa náležitosti dokumentácie, boli sme upozornení že webová stránka by mala byť statická a nemala by obsahovať dynamické časti. Boli sme informovaný o forme zápisníc, že to nemajú byť iba metazápisy, ale aby v zápisniciach bolo jasne napísané, k akým záverom sme sa dopracovali. Ďalšie stretnutie už budeme viesť my, každý šprint bude mať na starosti iný človek, každý šprint sa bude meniť aby každý z členov dostal priestor vyskúšať si túto úlohu. Vedúci nás upozornili na dôrazné dodržiavanie termínov.

2. V ďalšej časti stretnutia nasledovalo oboznámenie s architektúrou RoboCup. Projekt pozostáva z viacerých častí. Server simuluje hráčov, reálnu fyziku, pohyby hráčov. Hráči sú agenti (samostatná časť), v ktorých je implementovaná rozhodovacia logika. Komunikujú so serverom prostredníctvom TCP/IP protokolu. Server posiela naspäť agentom stav okolitého sveta a stav agentovho tela. Simulovaný je robot NAO. Ďalšou časťou je monitor, ktorý sa pripája na server a renderuje hru. Server teda len počíta. Ďalej existuje ešte podporné prostredie. To umožňuje riadiť simuláciu – posunutie hráča, lopty. Využíva sa na testovanie a ladenie hráča. Vyhodnocuje pozície predmetov, často je napojené na hráča.

   Server simuluje hru po krokoch. 1 krok predstavuje 20ms.Server príjme pakety,

realizuje, odsimuluje, transportuje pakety pre hráčov. Pakety sa prenesú hráčom, prepočítajú hodnoty a hráči sa rozhodnú čo urobia a odošlú pakety. Táto činnosť predstavuje 1 cyklus, ktorý sa musí stihnúť do 20ms. Preto pri simulácii je dôležité sieťové oneskorenie – aby bolo čo najnižšie.

Boli sme upozornení, že komunikácia medzi agentami je zakázaná a nesmieme vniesť takúto implementáciu do projektu. Povolené je iba centralizované riešenie (prenesené cez server), pretože i pri reálnom futbale ak jednotlivý hráč zakričí nejaký pokrik, tento pokrik počujú i ostatní hráči súperiaceho tímu. Aby sa však predišlo tomu, že pri tejto komunikácii prenesieme celý vnútorný stav hráča, komunikácia je obmedzená len na určitý počet znakov (popr. Bitov) – napr 10, čo zodpovedá i reálnej hre.

Ďalej sme sa oboznámili so spôsobom, ako hráč dostáva informácie o jeho polohe. Polohu dostáva v polárnych súradniciach, teda kdekoľvek je tak hráč má súradnice [0,0]. Hráč dostane i informácie o bránke a objektoch ktoré vidí, ale iba v jeho zornom uhle, tj 120 stupňov. Musí si dopočítať kde sa vzhľadom na polohu ihriska nachádza a akým smerom je natočené ihrisko, keď chce kopnúť loptu. Robot môže hlavou otáčať o 60 stupňov a teda môže vidieť i čo sa deje mimo tohto obvyklého zorného uhla.

3. Informácie o RoboCupe sú spracované na wiki od minuloročného tímu, avšak keďže ten server nebeží, musíme si rozbehať wiki sami. Dostali sme k nej zdrojové kódy ako i zdrojové kódy minuloročného tímu, v ktorých máme pokračovať.

4. V ďalšej časti stretnutia nám pedagogický vedúci popísal vnútornú štruktúru hráča. Táto štruktúra pozostáva z komunikácie, modelu sveta, rozhodovanie/riadenie, výberu akcií. Každý kĺb robota má dané ohraničenia, na ktoré treba myslieť.

5. V práci by sme sa mohli venovať časti rozhodovania. Existuje viacero spôsobov pohybu hráča, ako pohyb so stabilizáciou, či bez stabilizácie. Nestabilizovaný pohyb je rýchlejší, momentálne má tento projekt najrýchlejšiu chôdzu na svete, asi 3,5m/s. V práci by bolo dobré, keby sa nám podarilo vylepšiť rozhodovanie kedy aký pohyb použiť a implementovať parametrizovaný pohyb. Totiž ak hráč beží k lopte rovno, musí sa pri nej otočiť, aby kopol loptu správnym smerom. Táto akcia pozostáva teda z dvoch pohybov, rýchlej chôdze a natočenia. Síce hráč beží rýchlo, no stráca veľa času pri natáčaní na správny smer. Ideálne by bolo, keby hráč bežal trochu obkľukou (tj parametricky) a teda aby k lopte prišiel natočený na správny smer – smer ktorým chce kopnúť loptu. Vtedy by nemusel robiť druhý pohyb - natočenie, čím ušetrí veľa času.

6. Schopnosti hráča sa delia na lowskill a highskill. Lowskill sú spísané v xml súboroch, highskill sú kombináciou lowskillov. Našou úlohou by bolo vylepšiť tieto highskill, urobiť ich systematickejšie, aby sa hráč vedel lepšie učiť. Aby bolo možné vložiť nové schopnosti ako cez plugin a teda umožniť lepšiu rozšíriteľnosť. Tiež sme boli

informovaní, že by predagogickí vedúci radi vynechali z projektu časti napísané v jazyku Ruby a teda aby sme ich prepísali do Javy.

7. Ku koncu stretnutia sme si zosumarizovali úlohy, ktoré máme urobiť. Mali by sme sa hlavne posnažiť rozbehať projekt, pozrieť zahraničné tímy, súťaže a o týždeň v ďalšom šprinte budeme robiť analýzu čo používajú a ako sa ich roboti rozhodujú. Mali by sme pozerať hráčov 2013, pozerať najnovšie články na IEEE, odkazy nájdeme na stránke Ing. Kapustíka. Do budúceho týždňa si máme rozbehať wiki minuloročného tímu, keďže ich server s wiki nie je dostupný. Na wiki stránke pozrieť materiály, v ktorých sa dozvieme ako projekt funguje. Súčasťou našej práce bude i zapisovať si chyby a tak udržiavať wiki aktuálnu. Každý z členov tímu si má rozbehať hráčov a server na vlastnom PC. Máme si určiť scrummastera do ďalšieho stretnutia na ďalší šprint, dohodnúť sa s druhým tímom akou formou chceme viesť stretnutie, keďže stretnutie budeme viesť spoločne s druhým tímom. Doniesť vytlačenú zápisnicu s vyznačením čo sme spravili za týždeň. Máme si tiež rozbehať podporné prostriedky pre manažment projektu. Vhodné bude ak si scrummaster naštuduje o scrume už niečo v predstihu, aby sa v pondelok na prednáške mohol pýtať pokročilejšie veci. Tiež si máme založiť mailing list a poslať pedagogickému vedúcemu tímu tento kontakt. Založiť si šablóny dokumentácie – zápisov, šablóny dokumentácie. Skúsiť premyslieť, ktoré časti projektu by koho z nás viac zaujímali, kto z tímu by sa ktorej oblasti chcel venovať – ako napríklad strojové učenie, architektúra, modularita systému. Scrum bude prebiehať v dvoch týždňoch, úlohy sa budú považovať za splnené ak budú otestované, zdokumentované a funkčné. Priradenie úlohy neznamená, že celú úlohu musí vykonať človek, ktorý ju má na starosti, je však za ňu zodpovedný. Na výsledné hodnotenie na konci semestra bude treba body zo Scrumu, treba mať teda podiel na vypracovaní úloh.

Úlohy do ďalšieho stretnutia

Číslo úlohy	Zhrnutie úlohy	Riešiteľ	Zodpovedný
ROBOCUPTP-1	Vytvoriť zápisnicu zo sedenia	Martin Markech	Martin Markech
ROBOCUPTP-2	Rozbehať wiki	Martin Markech	Martin Markech
ROBOCUPTP-3	Tvorba plagátu a loga tímu	Štefan Linner	Štefan Linner
ROBOCUPTP-4	Tvorba Web stránky	Štefan Horváth	Štefan Horváth
ROBOCUPTP-5	Príprava šablón pre dokumentáciu a zápisnice	Michal Blanárik	Michal Blanárik
ROBOCUPTP-6	Sfunkčnit Jiru, prepojiť s Bitbucket	Martin Markech	Martin Markech
ROBOCUPTP-7	Rozbehanie simulačného prostredia robotického futbalu	Celý tím	Roman Moravčík
ROBOCUPTP-8	Rozbehať server	Martin Markech	Martin Markech
ROBOCUPTP-9	Vytlačiť zápisnicu zo	Martin Markech	Martin Markech

	sedenia		
ROBOCUPTP-10	Premyslieť, ktoré časti projektu by koho z nás viac zaujímali	Celý tím	Roman Moravčík
ROBOCUPTP-11	Určiť role v tíme	Roman Moravčík	Martin Markech
ROBOCUPTP-12	Určiť scrummastera na 1. šprint	Roman Moravčík	Martin Markech
ROBOCUPTP-13	Jabber	Martin Markech	Tomáš Nemeček

A-2. Zápis z 2. Stretnutia tímu č. 9

Dátum: 9.10.2013
Miestnosť: Jobsové softvérové štúdio (lab 1.31) (FIIT-STU)
Prítomní:

Pedagóg : Ing. Ivan Kapustík
Členovia tímu: Bc. Filip Blanárik, Bc. Michal Blanárik, Bc. Štefan Horváth, Bc. Martin Markech, Bc. Roman Moravčík, Bc. Tomáš Nemeček

Stretnutie viedol: Bc. Martin Markech
Zápis: Bc. Filip Blanárik
Zvolený nasledovný zapisovateľ: Bc. Roman Moravčík

Téma stretnutia

Zahájenie prvého šprintu, naplánovanie úloh na prvý šprint a odhadovanie časovej náročnosti jednotlivých úloh.

Vyhodnotenie úloh z predchádzajúceho stretnutia

Číslo úlohy	Zhrnutie úlohy	Riešiteľ	Zodpovedný	Stav úlohy
ROBOCUPTP-1	Vytvoriť zápisnicu zo sedenia	Martin Markech	Martin Markech	Splnené
ROBOCUPTP-2	Rozbehať wiki	Martin Markech	Martin Markech	Splnené
ROBOCUPTP-3	Tvorba plagátu a loga tímu	Štefan Linner	Štefan Linner	Splnené
ROBOCUPTP-4	Tvorba Web stránky	Štefan Horváth	Štefan Horváth	Splnené
ROBOCUPTP-5	Príprava šablón pre dokumentáciu a zápisnice	Michal Blanárik	Michal Blanárik	Splnené
ROBOCUPTP-6	Sfunkčniť Jiru, prepojiť s Bitbucket	Martin Markech	Martin Markech	Splnené
ROBOCUPTP-7	Rozbehanie simulačného prostredia robotického futbalu	Celý tím	Roman Moravčík	Čiastočne splnené

ROBOCUPTP-8	Rozbehať server	Martin Markech	Martin Markech	Čiastočne splnené
ROBOCUPTP-9	Vytlačiť zápisnicu zo sedenia	Martin Markech	Martin Markech	Splnené
ROBOCUPTP-10	Premyslieť, ktoré časti projektu by koho z nás viac zaujímali	Celý tím	Roman Moravčík	Čiastočne splnené
ROBOCUPTP-11	Určiť role v tíme	Roman Moravčík	Martin Markech	Splnené
ROBOCUPTP-12	Určiť scrummastera na 1. šprint	Roman Moravčík	Martin Markech	Splnené
ROBOCUPTP-13	Jabber	Martin Markech	Tomáš Nemeček	Nesplnené

Opis stretnutia

1. Na úvod stretnutia scrummaster Bc. Martin Markech zhodnotil mieru splnenia jednotlivých úloh z predošlého obdobia. Úlohy, ktoré boli naplánované sme splnili, okrem úloh ROBOCUPTP-7 a ROBOCUPTP-8.

   Úloha ROBOCUPTP-7 nebola úspešne dokončená z dôvodu problémov, ktoré sa pri jej riešení vyskytli a ich riešenie si vyžaduje viac času.
   - Bc. Martin Markech mal problém s blokovanim exec v ruby, ktorý bol pravdepodobne spôsobený ruby version managerom
   - Bc. Filip Blanárik mal problém so spustením viacerých agentov Jim súčasne
   - Bc. Michal Blanárik mal problém so spustením viacerých agentov Jim súčasne
   - Bc. Tomáš Nemeček mal problém s aspektom jimu v eclipse
   - Bc. Štefan Horváth mal problém s ruby skriptom, ktorý spôsoboval ukončenie procesu agenta Jima.
   - Bc. Roman Moravčík mal problém so spustením agenta z testovacieho frameworku.
   - Bc. Štefan Linner mal problém so spustením agenta z testovacieho frameworku.

   Ing. Marián Lekavý, PhD. poskytol tímom najnovšiu verziu zdrojového kódu hráča, ktorý bol použitý aj pri súťaži RoboCup.
   Úloha ROBOCUPTP-8 nebola úspešne dokončená, pretože server určený pre tímové projekty nebol sprístupnený v dostatočnom predstihu. Ako náhrada bol použitý dočasný server http://robocup.matho.sk/.
   Výsledkom splnenia úlohy ROBOCUPTP-6 je použitie školskej Jiri na adrese: http://jira.fiit.stuba.sk a Bitbucketu na adrese https://bitbucket.org/robocup_tp09, ktorý slúži na vizualizáciu verziovania s Gitom.
   Role v tíme boli určené nasledovne:
   - Vedúci tímu: Roman Moravčík
   - Manažér podpory vývoja: Martin Markech
   - Manažér dokumentovania: Michal Blanárik
   - Manažér rozvrhu: Tomáš Nemeček
   - Manažér monitorovania projektu: Filip Blanárik

- Manažér rizík: Štefan Horváth
- Manažér kvality: Štefan Linner

2. Ďalší priebeh stretnutia sa venoval úlohám pre prvý šprint. Vedúci tímového projektu nám pridelili úlohy, ktoré je nevyhnutné splniť do konca šprintu, ktorého dĺžka bola stanovená na dva týždne. Keďže úloha ROBOCUPTP-7 nebola úspešne splnená v predchádzajúcom období, jej splnenie v čo najskoršom termíne ma vysokú prioritu a jej súčasťou má byť dokument. Tento dokument má popisovať problémy a ich riešenie, ktoré sa vyskytli pri inštalácií a spúšťaní simulácie robotického hráča futbalu. Riešenia týchto problémov rozšíria inštalačný návod na wiki.

   Ďalej sa máme dôsledne oboznámiť so zdrojovými kódmi hráča, identifikovať kľúčové triedy a namodelovať diagramy zachytávajúce súčasný stav, ktorý dosiahli predošlí riešitelia. Výsledkom tejto úlohy by mal byť dokument s opisom súčasnej architektúry hráča. Je vhodné, aby sme si analýzu zdrojových kódov medzi sebou rozdelili. V rámci overenia tejto úlohy máme vykonať jednoduchú modifikáciu správania alebo pohybu hráča. Tiež má každý z nás napísať k vhodnej časti unit test.

   V ďalšom šprinte nám budú poskytnuté výsledky diplomových prác, ktoré zanalyzujeme a vhodným spôsobom zahrnieme do súčasného riešenia.

   Tiež sa máme oboznámiť s hráčmi vytvorenými zahraničnými tímami. Na konci šprintu by mal mať každý z nás vybratý tím, ktorý si preštuduje.

   Na základe analýzy zdrojových kódov a spoločného stretnutia (brainstroming technika) máme vytvoriť backlog úloh a vložiť ich do Jiri. Finálnu verziu zoznam úloh máme poslať Ing. Ivanovi Kapustíkovi do poslednej nedele 17:00 pred koncom šprintu.

   Boli sme upozornení, že je potrebné, aby boli zápisnice zo stretnutia vytvorené do konca týždňa, v ktorom sa konalo stretnutie a na každé stretnutie ma scrummaster priniesť vytlačený zoznam úloh s percentami ich splnenia a burndown chartom.

3. V nasledujúcej časti stretnutia sme sa venovali odhadovaniu časovej náročnosti jednotlivých naplánovaných úloh s použitím techniky planning poker.

Úlohy na prvý šprint

Číslo úlohy	Zhrnutie úlohy	Riešiteľ	Zodpovedný	Body	Čas
ROBOCUPTP-7 ROBOCUPTP-14	Rozbehanie simulačného prostredia robotického futbalu + dokument	Celý tím	Roman Moravčík	5 b	42 hod
ROBOCUPTP-15	Konfigurácia tímového servera, pokračovanie úlohy ROBOCUPTP-8	Martin Markech	Martin Markech	1 b	6 hod
ROBOCUPTP-13	Jabber	Martin Markech	Tomáš Nemeček	0 b	0,5 hod

ROBOCUPTP-17	Vytvoriť zápisnicu zo sedenia	Filip Blanárik	Michal Blanárik	1 b	2 hod
ROBOCUPTP-18	Zostavenie backlogu, pokračovanie úlohy: ROBOCUPTP-10	Celý tím	Martin Markech	3 b	42 hod
ROBOCUPTP-19	Analýza zdrojových kódov	Celý tím	Michal Blanárik	13 b	42 hod
ROBOCUPTP-20	Analýza zahraničných tímov	Celý tím	Štefan Linner	1 b	13 hod
ROBOCUPTP-21	Zmeniť správanie hráča	Celý tím	Filip Blanárik	8 b	14 hod
ROBOCUPTP-22	Vytvoriť unit testy	Celý tím	Tomáš Nemeček	5 b	7 hod
ROBOCUPTP-23	Spustenie stránky na tímovom serveri	Štefan Horváth	Štefan Horváth	0,5 b	4 hod
ROBOCUPTP-24	Analyzovať dostupné nástroje na review kódu	Celý tím	Martin Markech	1 b	7 hod
ROBOCUPTP-25	Coding guidelines	Ešte neurčený	Tomáš Nemeček	0,5 b	2 hod

A-3. Zápis z 3. Stretnutia tímu č. 9

Dátum: 16.10.2013

Miestnosť: Jobsové softvérové štúdio (lab 1.31) (FIIT-STU)

Prítomní:

Pedagóg : Ing. Ivan Kapustík
Členovia tímu: Bc. Filip Blanárik, Bc. Michal Blanárik, Bc. Štefan Horváth Bc. Štefan Linner, Bc. Martin Markech, Bc. Roman Moravčík Bc. Tomáš Nemeček

Stretnutie viedol: Ing. Ivan Kapustík

Zápis: Bc. Roman Moravčík

Zvolený nasledovný zapisovateľ: Bc. Michal Blanárik

Téma stretnutia

Vyhodnotenie aktuálnych úloh v šprinte.

Vyhodnotenie úloh z predchádzajúceho stretnutia

Číslo úlohy	Zhrnutie úlohy	Riešiteľ	Zodpovedný	Stav úlohy
ROBOCUPTP-7	Rozbehanie simulačného prostredia robotického futbalu	Celý tím	Roman Moravčík	Splnené

ROBOCUPTP-8	Rozbehať server	Martin Markech	Martin Markech	Splnené
ROBOCUPTP-10	Premyslieť, ktoré časti projektu by koho z nás viac zaujímali	Celý tím	Roman Moravčík	Čiastočne splnené
ROBOCUPTP-13	Jabber	Martin Markech	Tomáš Nemeček	Nesplnené
ROBOCUPTP-15	Pokračovanie ROBOCUP-TP8	Martin Markech	Martin Markech	Splnené
ROBOCUPTP-15	Vytvoriť zápisnicu zo sedenia	Filip Blanárik	Michal Blanárik	Splnené
ROBOCUPTP-18	Zostavenie backlogu, pokračovanie úlohy: ROBOCUPTP-10	Celý tím	Martin Markech	Čiastočne splnené
ROBOCUPTP-19	Analýza zdrojových kódov	Celý tím	Michal Blanárik	Nesplnené
ROBOCUPTP-20	Analýza zahraničných tímov	Celý tím	Štefan Linner	Nesplnené
ROBOCUPTP-21	Zmeniť správanie hráča	Celý tím	Filip Blanárik	Nesplnené
ROBOCUPTP-22	Vytvoriť unit testy	Celý tím	Tomáš Nemeček	Nesplnené
ROBOCUPTP-23	Spustenie stránky na tímovom serveri	Štefan Horváth	Štefan Horváth	Splnené
ROBOCUPTP-24	Analyzovať dostupné nástroje na review kódu	Celý tím	Martin Markech	Nesplnené
ROBOCUPTP-25	Coding guidelines	Tomáš Nemeček	Tomáš Nemeček	Splnené

Opis stretnutia

1. Na úvod stretnutia scrum master Bc. Martin Markech zhodnotil mieru splnenia jednotlivých úloh z predošlého obdobia. Úlohy, ktoré neboli splnené, sa presunuli do ďalšieho týždňa šprintu.
2. Bc. Tomáš Nemeček poukázal na nedostatočnú výpovednú hodnotu burndown grafu a navrhol doplnenie aj o cumulative flow diagram, ktorý lepšie znázorňuje pomer vytvorených úloh k splneným.
3. Vedúci Ing. Ivan Kapustík odporučil k nástrojom na review kódu analyzovať fakultný projekt PerConIK.

4. Na podnet Bc. Štefana Linnera sa prekonzultovala možnosť zúčastniť sa turnaja v Nemecku. Podanie prihlášky bude závisieť od dosiahnutých výsledkov do termínu podania.
5. Vedúci projektu nám pridelil na nasledujúce stretnutie analyzovať anotátor a rozdeliť highskilly na skupiny. Tiež nám priblížil, ako by nová architektúra mala fungovať, konkrétne výber highskillov.

Úlohy do ďalšieho stretnutia

Číslo úlohy	Zhrnutie úlohy	Riešiteľ	Zodpovedný	Body	Čas
ROBOCUPTP-23	Vytvoriť zápisnicu z 3. stretnutia - subtask ROBOCUPTP-16	Roman Moravčík	Roman Moravčík	1	2h
ROBOCUPTP-27	Analýza anotátora a anotácii - subtask ROBOCUPTP-19	Roman Moravčík	Roman Moravčík	2	2h
ROBOCUPTP-28	Rozdelenie high skillov - subtask ROBOCUPTP-19	Tomáš Nemeček	Roman Moravčík	2	2h

A-4. Zápis z 4. Stretnutia tímu č. 9

Dátum: 23.10.2013
Miestnosť: Jobsové softvérové štúdio (lab 1.31) (FIIT-STU)
Prítomní:

Pedagóg : Ing. Ivan Kapustík
Členovia tímu: Bc. Filip Blanárik, Bc. Michal Blanárik, Bc. Štefan Horváth, Bc. Štefan Linner, Bc. Martin Markech, Bc. Roman Moravčík, Bc. Tomáš Nemeček

Stretnutie viedol: Bc. Roman Moravčík
Zápis: Bc. Michal Blanárik
Zvolený nasledovný zapisovateľ: Bc. Tomáš Nemeček

Téma stretnutia

Vyhodnotenie prvého šprintu, zahájenie druhého šprintu, naplánovanie úloh na druhý šprint a odhadovanie časovej náročnosti jednotlivých úloh. Vytvorenie spoločného backlogu tímov 4 a 9 a zadefinovanie priorít jednotlivých úloh.

Vyhodnotenie úloh z predchádzajúceho stretnutia

Číslo úlohy	Zhrnutie úlohy	Riešiteľ	Zodpovedný	Stav úlohy
ROBOCUPTP-7 ROBOCUPTP-14	Rozbehanie simulačného prostredia robotického futbalu + dokument	Celý tím	Roman Moravčík	Splnené
ROBOCUPTP-15	Konfigurácia tímového servera, pokračovanie úlohy ROBOCUPTP-8	Martin Markech	Martin Markech	Splnené
ROBOCUPTP-13	Jabber	Martin Markech	Tomáš Nemeček	Splnené
ROBOCUPTP-18	Zostavenie backlogu, pokračovanie úlohy: ROBOCUPTP-10	Celý tím	Martin Markech	Splnené
ROBOCUPTP-19	Analýza zdrojových kódov	Celý tím	Michal Blanárik	Splnené
ROBOCUPTP-20	Analýza zahraničných tímov	Celý tím	Štefan Linner	Splnené
ROBOCUPTP-21	Zmeniť správanie hráča	Celý tím	Filip Blanárik	Splnené
ROBOCUPTP-22	Vytvoriť unit testy	Celý tím	Tomáš Nemeček	Splnené
ROBOCUPTP-23	Spustenie stránky na tímovom serveri	Štefan Horváth	Štefan Horváth	Splnené
ROBOCUPTP-24	Analyzovať dostupné nástroje na reviewkódu	Celý tím	Martin Markech	Splnené
ROBOCUPTP-25	Coding guidelines	Tomáš Nemeček	Tomáš Nemeček	Splnené
ROBOCUPTP-26 ROBOCUPTP-16	Vytvorenie zápisnice k 3. stretnutiu	Roman Moravcik	Roman Moravcik	Splnené
ROBOCUPTP-27 ROBOCUPTP-19	Analýza anotátora a anotácii	Roman Moravcik	Roman Moravcik	Splnené
ROBOCUPTP-28 ROBOCUPTP-19	Rozdelenie highskillov	TomasNemecek	Roman Moravcik	Nesplnené

Opis stretnutia

4. Na úvod stretnutia vyjadril vedúci projektu Ing. Ivan Kapustík kritiku k zápisnicu z 3. stretnutia. Nedostatky sa týkali neskorého zverejnenia zápisnice, a tiež nedostatok informácií, ktoré sú v zápisnici uvedené, vďaka čomu pripomína skôr meta zápis. V nasledujúcom priebehu projektu je potrebné zlepšiť kvalitu zápisníc.
5. Boli predvedené cummulative flow diagramu a burndown chartu z prvého šprintu, ku ktorým je potrebné okrem samotného diagramu pridať aj výpisy úloh pretože bez týchto dodatočných údajov nie je možné v plnej miere interpretovať informácie, ktoré tieto diagramy majú poskytovať.
6. V ďalšom priebehu stretnutia prebehlo vyhodnotenie úloh, ktoré boli naplánované na prvý šprint.
   - Rozbehanie simulačného prostredia sa podarilo všetkým členom tímu. Návody inštaláciu sú uchované v príslušnom dokumente. Je potrebné podľa nich aktualizovať sekciu *Návody a inštalácie* na wiki.

- Bc. Martin Markech spojazdnil tímový server, na ktorom sa nachádza stránka tímu a tiež Jabber ako nástroj na komunikáciu v rámci tímu.
- V rámci prvého šprintu bola vytvorený dokument, v ktorom je zachytená analýza súčasného stavu projektu robotického futbalu. Okrem opisov funkcionality a úloh jednotlivých tried boli vytvorené aj diagramy tried v rámci jednotlivých balíkov. Z časti programu, ktorý je písaný v Ruby bola vygenerovaná automatická dokumentácia, ktorá má podobnú štruktúru ako JavaDoc. Spomenuté dokumenty sme predviedli a bolo nám povedané, že odkaz na vytvorené dokumenty máme umiestniť na stránku tímu.
- Z dôvodu nestabilného pripojenia sme sa zdržali pri predvádzaní dokumentov, a preto by bolo vhodné mať na nasledujúce stretnutie pripravenú lokálnu kópiu súborov, ktoré chceme predviesť vedúcemu projektu.
- Tím momentálne používa službu SkyDrive na zdieľanie súborov. Z dôvodu pretrvávajúcich problémov niektorých členov tímu sa v druhom šprinte začne používať GoogleDrive.
- K úlohe zmeny správania hráča sa vyjadril Bc. Martin Markech, ktorý upravil plánovač tak, aby robot vykonával dookola dva lowskilly, a to konkrétne kývanie rukami hore a dole. Bc. Štefan Horváth upravil správanie hráča tak, že ak sa robot nachádza pri lopte spadne na chrbát. Bc. Michal Blanárik upravil plánovač na vykonávanie len jedného pohybu a to čupnutie a predpaženie jednej z rúk. Ostatní členovia tímu tiež úspešne realizovali úpravu správania sa hráča a výsledky zaznamenali v príslušnom dokumente.

  Pri vytváraní pohybu v aplikácií Editor pohybov sa nepodarila nájsť funkcia exportu pohybu do XML. Táto funkcia by mala byť v aplikácií už implementovaná, a preto treba v nasledujúcom priebehu projektu overiť či je chyba v GUI alebo sme len nenašli túto funkciu v rámci používateľského rozhrania.
- Nasledujúca úloha bolo vytvorenie unit testov. Každý člen tímu vytvoril unit test pre jednu metódu a výsledky svojej práce zdokumentoval. Momentálne sa ešte všetky tieto testy nenachádzajú v hlavnej vývojovej vetve projektu, a preto ich je tam potrebné doplniť. Bc. Tomáš Nemeček vytvoril unit test pre metódu triedy *Tacticalinfo* vďaka čomu zistil, že táto metóda sa nespráva podľa očakávaní. Bc. Štefan Linner vytvoril unit test pre metódu, ktorá zisťuje či sa agent nachádza na pozícií vo formácií. V tomto prípade sa metóda správa podľa očakávaní.
- V rámci prvého šprintu prebehla analýza dostupných nástrojov na codereview. Bol vybratý nástroj PerConIK, ktorý je vyvíjaný na škole a je dostupný zadarmo, kým iné riešenia sú často krát spoplatnené. V prípade problémov je možné sa obrátiť na Karola Rástočného.
- Bc. Tomáš Nemeček vytvoril metodiku na základe metodiky, ktorá bola používaná v rámci projektu 3D robotický futbal v minulých rokoch.
- Bc. Roman Moravčík sa vyjadril k súčasnému stavu využívania anotácií a anotátora v projekte. Automatický anotátor funguje na základe spriemerovania dát z viacnásobných spustení testovacieho prípadu. Takto vytvorené údaje môžu

byť užitočné pri rozhodovaní, ale momentálna implementácia ich využíva vo veľmi obmedzenej až nulovej miere a anotácia ani neexistuje pre všetky pohyby.
- Úloha rozdelenia highskillov nebola v rámci prvého šprintu splnená. Jeden z dôvodov jej nesplnenia je neskoršie pridanie do backlogu šprintu.
- Bc. Michal Blanárik vyhľadal dostupné informácie k zahraničným tímom, ktoré sa v minulých rokoch zúčastnili medzinárodnej súťaže RoboCup3D. Odkaz na tieto informácie by sa mal nachádzať na stránke tímu.

7. Celkovo bolo na prvý šprint naplánovaných 184h práce a tím vykázal 130h odrobených. Niektoré úlohy ako je napríklad rozbehanie simulačného prostredia sa stihli s veľkou časovou rezervou. Na druhej strane vytvorenie unit testu bolo časovo podhodnotené na začiatku šprintu.
8. Ing. Ivan Kapustík poskytol tímu diplomové práce z minulých rokov, ktoré sa týkali témy 3D robotického futbalu, a výsledky týchto prác by bolo vhodné zakomponovať do projektu.
9. Do budúceho stretnutia je potrebné vytvoriť jednotnú dokumentáciu projektu, v ktorej sa budú nachádzať všetky doteraz vytvorené dokumenty v jednotnej forme.
10. V ďalší priebeh stretnutia bol realizovaný spoločne s tímom 4. Na základe backlogov oboch tímov boli určené tieto úlohy:
    - 9.1 (critical) Verzia servera – upraviť premenne závislé od verzie servera
    - 9.2 (critical) HÍbková analýza zahraničných tímov
    - 9.3 (critical) Zjednodušiť časti v RUBY
    - 4.2 (critical) Odstrániť RUBY
    - 9.4.1 (minor) Spojazdniť automatický anotátor + dorobiť chýbajúce anotácie
    - 9.4.2 (major) Vytvoriť architektúru Highskills
    - 9.4.3 (major) Návrh strategickej vrstvy
    - (major) Implementácia úloh 9.4.x – realizácia úlohy ako 3 user stories
    - 9.5.1,2 (major) Odstránenie nepoužívaného kódu
    - 9.5.3 (minor) Presun funkcií do RoboCupLibrary
    - 9.5.4 (trivial) Premenovanie LowSkill-ov
    - 9.6 (critical) Analýza diplomových prác a doplniť do úloh 9.4.2, 9.4.3
    - 9.6.2 (minor) Integrácia diplomových prác – realizácia úlohy pomocou niekoľkých user stories
    - 9.7.1 (minor) Test framework – doplniť UI
    - 9.7.2 (minor) Upraviť test framework podľa zmien v architektúre hráča
    - 9.8 (minor) Zmena taktiky na základe výkrikov
    - 4.1 (minor) Vylepšenie stability hráča po kopnutí do lopty – treba brať do úvahy aj o aký kop sa jedná, pretože v prípade ak kopne hráč loptu do veľkej diaľky môže obetovať kvôli väčšej vzdialenosti aj stabilitu. Tiež je k dispozícii viac rôznych typov kopov, pre ktoré je potrebné riešiť stabilitu vždy samostatne.
    - Parametrický kop do lopty – vzdialenosť a uhol
    - 4.3 (blocker) Rozbehanie GIT-u – spoločný git pre tímy 4 a 9
    - (critical) Automatický Build systém – aspoň raz za deň spustí build projektu, vykoná unit testy a oznamy výsledok.

- 4.4 (critical) Aktualizovať návod na inštaláciu
- 4.5 (trivial) Editor pohybov -> XML
- 4.6 (minor) Príchod k lopte – vylepšiť – dostatočne rýchly a na presnú pozíciu
- 4.9.1 (minor) Nastavenie sa k lopte
- 4.9.2 (minor) Rozhodnutie čo s loptou
- 4.12 (trivial) vytvorenie rozhodovania pri strete hráča s hráčom
- 4.13 (minor) Hľadanie lopty – kam sa pozerať, otáčanie hlavy nezávisle na pohyboch.
- 4.14 (minor) Zlepšiť driblovanie
- 4.15 (trivial) Automatické reštartovanie hry

11. Ing Marian Lekavý nám vysvetlil dva možné prístupy k architektúre rozhodovania. Rozhodovanie prebieha na nasledujúcich vrstvách:
    - strategická (útok po pravom krídle)
    - taktická (bež k lopte)
    - highskill (choď na pozíciu)
    - pohyby (otočenie, beh, chôdza,...)

    Tieto vrstvy môžu byť prepojené pomocou rozhraní, alebo vyššie vrstvy môžu v sebe uchovávať zoznam objektov nižšej vrstvy rozhodovania. Oba prístupy sú použiteľné pre projekt no finálna architektúra rozhodovania môže vyzerať inak.
12. Počas práce na projekte by mala existovať vývojová vetva programu, ktorá bude v každej fáze projektu spustiteľná.

Úlohy na druhý šprint

Číslo úlohy	Zhrnutie úlohy	Riešiteľ	Zodpovedný	Body	Čas
ROBOCUPTP-29	9.3.1 Rozdeliť ruby na balíky a vytvoriť jeden ukážkový príklad transformácie do javy	Martin Markech	Martin Markech	13b	13 hod
ROBOCUPTP-30	9.3.1.1 transformovať high skilly z ruby do javy	Celý tím	Roman Moravčík	40 b	40 hod
ROBOCUPTP-31	9.6.1,2 Analýza diplomových prác		Štefan Linner	13b	13 hod
ROBOCUPTP-32	9.2 Hĺbková analýza zahraničných tímov	Celý tím	Štefan Horvath	7 b	7 hod
ROBOCUPTP-33	4.3 GIT spoločný pre tímy 4 a 9	Martin Markech	Martin Markech	1 b	1 hod
ROBOCUPTP-34	CI	Tomáš Nemeček	Tomáš Nemeček	20b	20 hod
ROBOCUPTP-35	Dokumentácia k prvému šprintu	Filip Blanárik	Michal Blanárik	3b	3hod
ROBOCUPTP-36	Zápisnica k prvému šprintu	Michal Blanárik	Michal Blanárik	1 b	14 hod

A-5. Zápis z 5. Stretnutia tímu č. 9

Dátum: 30.10.2013
Miestnosť: Jobsové softvérové štúdio (lab 1.31) (FIIT-STU)
Prítomní:

Pedagóg : Ing. Ivan Kapustík
Členovia tímu: Bc. Filip Blanárik, Bc. Michal Blanárik, Bc. Štefan Horváth, Bc. Martin Markech, Bc. Roman Moravčík, Bc. Tomáš Nemeček

Stretnutie viedol: Bc. Roman Moravčík
Zápis: Bc. Tomáš Nemeček
Zvolený nasledovný zapisovateľ: Bc. Štefan Horváth

Téma stretnutia

Zhodnotenie stavu plnenia plánu na druhý šprint. Rozprava k jednotlivým úlohám.

Vyhodnotenie úloh z predchádzajúceho stretnutia

Číslo úlohy	Zhrnutie úlohy	Riešiteľ	Zodpovedný	Stav úlohy
ROBOCUPTP-29	9.3.1 Rozdeliť ruby na balíky a vytvoriť jeden ukážkový príklad transformácie do javy	Martin Markech	Martin Markech	Nesplnené
ROBOCUPTP-30	9.3.1.1 transformovať high skilly z ruby do javy	Celý tím	Roman Moravčík	Nesplnené
ROBOCUPTP-31	9.6.1,2 Analýza diplomových prác		Štefan Linner	Nesplnené
ROBOCUPTP-32	9.2 Hĺbková analýza zahraničných tímov	Celý tím	Štefan Horvath	Nesplnené
ROBOCUPTP-33	4.3 GIT spoločný pre tímy 4 a 9	Martin Markech	Martin Markech	Splnené
ROBOCUPTP-34	CI	Tomáš Nemeček	Tomáš Nemeček	Nesplnené
ROBOCUPTP-35	Dokumentácia k prvému šprintu	Filip Blanárik	Michal Blanárik	Splnené
ROBOCUPTP-36	Zápisnica k prvému šprintu	Michal Blanárik	Michal Blanárik	Splnené
ROBOCUPTP-67	Nastavenie zdielania backlogu medzi timami	Martin Markech	Martin Markech	Splnené

Opis stretnutia

13. Na úvod stretnutia ing. Ivan Kapustík zhodnotil zlepšenie kvality zápisnice za posledné stretnutie. Zápisnica spĺňala všetky stanovené náležitosti.
14. Bc. Filip Blanárik sa spýtal ing. Ivana Kapustíka na reprezentáciu user stories v Jire. Jeho otázka bola mierená na nutnosť rozloženia user stories na podúlohy analýzy, implementácie a testovania. Ing. Ivan Kapustík vysvetlil, že vzhľadom na ohnutie SCRUM procesu pri výučbe nie je nutné explicitné delenie user stories na podúlohy, ale je vhodné naznačiť jednotlivé časti plnenia v opise story. Na záver šprintu je nutné

doplniť zhrnutie výsledného produktu vytvoreného pri danej story do dokumentácie riadenia projektu na lepšie pochopenie.

15. Následne scrummaster Bc. Roman Moravčík zhodnotil mieru splnenia jednotlivých úloh z predošlého šprintu.
    - Úloha ROBOCUPTP-35 bola úspešne splnená a výsledný dokument za šprint bol zavesený na webové sídlo tímu.
    - Úloha ROBOCUPTP-36 bola úspešne splnená a výsledný dokument za šprint bol zavesený na webové sídlo tímu.
    - Úloha ROBOCUPTP-34 nie je splnená. Bc. Tomáš Nemeček popísal postupné skúšanie jednotlivých nástrojov na kontinuálnu integráciu. Pri riešení tejto úlohy vyskúšal nástroj Jenkins, ktorý sa mu nepodarilo zintegrovať s tímovým repozitárom. Následne na odporučenie Bc. Martina Markecha zkontaktoval ing. Karola Rástočného a požiadal o prístup k školskej verzii nástroja Atlassian Bamboo. Doposiaľ sa mu nepodarilo vytvoriť jednotlivé testy, kvôli nekompletnosti nastavení v nástroji. Taktiež prejavil obavy z nedokončenia úlohy, kvôli veľkej odkázanosti pri konfigurácii nástroja na externých kolegov.
        - Pri tejto úlohe Bc. Tomáš Nemeček predniesol návrh na jednotný systém buildovania distribúcie agenta, ktorý implementoval za pomoci nástroja ANT. Následne sa Bc. Tomáš Nemeček spýtal na odstránenie duplicitného kódu balíka build, pre ktorý vytvoril náhradu v ANTe . Ing. Ivan Kapustík prejavil záujem o tento nápad a zaujal stanovisko, že je vhodné označiť balík slúžiaci na buildovanie distribúcie agenta za vhodný na odstránenie. Následne teda ponechať obe implementácie a v neskorších štádiách projektu zvážiť odstránenie tohto balíka.
    - Úloha ROBOCUPTP-33 bola úspešne splnená a členovia tímu 4 získali prístup k repozitáru tímu 9.
    - Úloha ROBOCUPTP-32 nie je splnená. Členovia tímu si rozdelili 7 zahraničných tímov medzi sebou. Bc. Tomáš Nemeček zhodnotil jeden zo skúmaných tímov a ako záver uviedol potrebu doplnenia modulu učenia a optimalizácie pohybov agenta do projektu.
    - Úloha ROBOCUPTP-31 nie je splnená. Bc. Štefan Linner doposial zanalyzoval väčšiu časť práce ing. P. Passaka a uviedol ze ing. Passakovi sa podarilo pomocou učenia zlepšiť kop, chôdzu a vytvoriť modul učenia agenta v projekte TestFramework.

    Ing. Ivan Kapustík vyzdvihol na základe doterajších informácii dôležitosť učenia agenta na dosiahnutie najoptimálnejších pohybov. Ďalej odporučil otestovať framework agenta a navrhnúť najlepšie miesto na doplnenie učenia sa. Následne navrhol vytvoriť úlohu na doplnenie učenia agenta do backlogu.
    - Úloha ROBOCUPTP-29 nie je splnená. Bc. Martin Markech vyjadril obavy z nedostatočnej komunikácie medzi tímami a z výsledného neúspechu pri integrácii oboch častí implementovaných tímami 4 a 9. Ďalej dohodol spolu

s Bc. Roman Moravčík stretnutie s tímom 4, kde vytvoria spoločný návrh na odstránenie Ruby. Bc. Michal Blanárik navrhol vytvorenie príkladu jednoduchého plánovača na uľahčenie práce.

- Úloha ROBOCUPTP-30 nie je splnená. Bc. Roman Moravčík vytvoril dokument, v ktorom popísal body na ktoré si pri odstraňovaní Ruby dať pozor, duplicitnú funkcionalitu v ruby a systém odvodzovania tried v ruby. Ďalej zistil, že pri odvodzovaní tried z Javy existujú nepoužívané metódy. Výsledné odstraňovanie Ruby označil za jednoduchšie.

Ing. Ivan Kapustík označil komunikáciu s druhým tímom za rizikovú a vyzval manažéra rizík Bc. Štefana Horvátha na vytvorenie dokumentu pre potreby riešenia tohto rizika.

- Úloha ROBOCUPTP-67 bola splnená. Bc. Martin Markech prepojil backlogy oboch tímov do spoločného. Išlo o doplnenie druhého tímu do nášho projektu a odčlenenie úloh pomocou premennej komponent.

Bc. Štefan Linner sa na záver stretnutia spýtal ing. Ivana Kapustíka do akej hĺbky je potrebné analyzovať zahraničné tímy. Či stačí urobiť analýzu z videí jednotlivých tímov alebo aj z dokumentácií tímov. Ing. Ivan Kapustík odpovedal, že je vhodné aby sa analyzovali z oboch zdrojov, keďže z videí je možné vysledovať spôsoby ako agenty prihrávajú, akým spôsobom tvoria formácie, nakoľko držia formácie, ako spracovávajú loptu, ako hľadajú na ihrisku, ale v publikáciách sú jednotlivé spôsoby podrobnejšie popísané.
Po tejto otázke nasledovala krátka debata o zisťovaní stavu na ihrisku. Kedy Ing. Ivna Kapustík uviedol, že je vhodné analyzovať stav, kedy sa dá točiť hlavou robota, aby nedošlo k ovplyvneniu iných vykonávaných akcií.
Ďalej ing. Ivan Kapustík poznamenal, že je vhodné jednotlivé burndown charty uviesť do dokumentu riadenia projektu s zhodnotením postupu jednotlivých parametrov ako odhadovaný čas, reálne odrobený čas a iné. Na záver pripomenul, aby sa do dokumentu riadenia pridali fotky z minulého stretnutia, kedy sa vytváral spoločný backlog pre oba tímy.

Doplňujúce úlohy šprintu

Číslo úlohy	Zhrnutie úlohy	Riešiteľ	Zodpovedný	Body	Čas
ROBOCUPTP-77	Vytvoriť zápisnicu z 5. stretnutia - subtask ROBOCUPTP-16	Tomáš Nemeček	Tomáš Nemeček	1	2h

A-6. Zápis z 6. Stretnutia tímu č. 9

Dátum: 06.11.2013
Miestnosť: Jobsové softvérové štúdio (lab 1.31) (FIIT-STU)
Prítomní:

Pedagóg :	Ing. Ivan Kapustík
Členovia tímu:	Bc. Filip Blanárik, Bc. Michal Blanárik, Bc. Štefan Horváth, Bc. Martin Markech, Bc. Roman Moravčík, Bc. Tomáš Nemeček

Stretnutie viedol: Bc. Roman Moravčík
Zápis: Bc. Štefan Horváth
Zvolený nasledovný zapisovateľ: Bc. Štefan Linner

Téma stretnutia

Zhodnotenie stavu úloh z prvého šprintu, zadefinovanie úloh na ďalší šprint.

Vyhodnotenie úloh z predchádzajúceho stretnutia

Číslo úlohy	Zhrnutie úlohy	Riešiteľ	Zodpovedný	Stav úlohy
ROBOCUPTP-29	9.3.1 Rozdeliť ruby na balíky a vytvoriť jeden ukážkový príklad transformácie do javy	Martin Markech	Martin Markech	Splnené
ROBOCUPTP-30	9.3.1.1 transformovať high skilly z ruby do javy	Celý tím	Roman Moravčík	Splnené
ROBOCUPTP-31	9.6.1,2 Analýza diplomových prác		Štefan Linner	Splnené
ROBOCUPTP-32	9.2 Hĺbková analýza zahraničných tímov	Celý tím	Štefan Horvath	Splnené
ROBOCUPTP-33	4.3 GIT spoločný pre tímy 4 a 9	Martin Markech	Martin Markech	Splnené
ROBOCUPTP-34	CI	Tomáš Nemeček	Tomáš Nemeček	Splnené
ROBOCUPTP-35	Dokumentácia k prvému šprintu	Filip Blanárik	Michal Blanárik	Splnené
ROBOCUPTP-36	Zápisnica k prvému šprintu	Michal Blanárik	Michal Blanárik	Splnené
ROBOCUPTP-67	Nastavenie zdielania backlogu medzi timami	Martin Markech	Martin Markech	Splnené

Opis stretnutia

1. Na úvod tím zhodnotil stav úlohy ROBOCUPTP-32 „Hĺbková analýza zahraničných tímov“. Úloha bola splnená, výstupný dokument sa nachádza na webovej stránke tímu. Niektorí z členov tímu zreferovali výsledky svojich analýz.
    - Bc. Filip Blanárik spomenul nájdenie architektúry a taktiky tímu.
    - Bc. Michal Blanárik spomenul detekciu padnutého spoluhráča sledovaním výšky jeho hlavy.

- Na margo toho nastala krátka diskusia o sledovaní spoluhráčov, k čomu sa vyjadril Bc. Roman Moravčík, že do istej miery táto funkcionalita existuje, a to konkrétne určovanie súperov, kde sa nachádzajú, či ležia alebo hrajú.
- Ďalšia krátka diskusia bola o spôsoboch vstávania robota po jeho páde, na čo reagoval Bc. Tomáš Nemeček s informáciou, že už existujú v našom projekte 2 druhy vstávania. Následne dal návrh na vylepšenie, že pri páde by robot mohol padnúť na ruky, čo by zrýchlilo jeho vstávanie lebo by nepadol úplne celý. Ešte poukázal na prácu Hudeca, ktorý vytvoril rýchle vstávanie robota.
- Bc. Štefan Linner odporučil výsledok zahraničného tímu RoboViz, čo je vylepšený monitor. Zhodnotil vylepšenú grafiku, možnosť ovládania hráča, predpripravené pohľady a celkovo lepšie usability. K tejto téme sa vyjadril Ing. Ivan Kapustík, že celý tím si má tento monitor rozbehať, vyskúšať a aktualizovať wiki o návod na inštaláciu RoboViz.
- Na záver Ing. Ivan Kapustík poukázal na potrebu spísania zaujímavých výsledkov analýz a našich nápadov do backlogu nie len pre túto úlohu.

2. Úloha ROBOCUPTP-34 „CI“ bola splnená. Postaral sa o to Bc. Tomáš Nemeček s pomocou nástroja Bamboo. Tiež stručne spomenul fungovanie v dvoch krokoch, a to preklad kódu a následné spustenie unit testov. Spúšťa sa to pre každú vetvu zvlášť, master raz za deň, vývojové vetvy každé 4h. Ing. Ivan Kapustík sa vyjadril, že by bolo dobré opraviť testy. Bc. Tomáš Nemeček vyjadril pochybnosti ku správnosti samotného testovaného kódu.
3. Úloha ROBOCUPTP-31 „Analýza DP“ bola splnená.
    - Bc. Roman Moravčík zreferoval prácu Martina Paššáka, kde testoval pohyby, vytvoril parametrickú chôdzu, ktorej vysokú úspešnosť zhodnotil Ing. Ivan Kapustík. Tiež sa venoval parametrickému kopu do lopty, kde využil na kop špicou do diaľky a kop hranou na blízko, čo má za následok presnejší kop. V práci sa tiež venoval prihrávkam a driblovaniu, kde ale vzniká problém so sledovaním lopty. Lopta sa stratí zo zorného poľa hráča a hráč tak začne hľadať loptu.
    - Bc. Štefan Linner zreferoval prácu Petra Paššáka, v ktorej autor vylepšil pohyby hráča. Hráč vie bežať rýchlo no má problém so zastavením. Tiež sa stáva, že ešte pred kopom si jemne kopne do lopty, čo ale nevieme s istotou označiť za chybu. Preto Ing. Ivan Kapustík odporučil skontaktovať Bc. Petra Paššáka kvôli informáciám.
    - Tím zhodnotil úspešnosť testovača autora, na čo zareagoval Ing. Ivan Kapustík potrebou vykonať analýzu jeho testovača. Tiež odporučil inšpirovať sa a vytvoriť testframework podobnej kvality. Padlo tvrdenie, že súčasný TestFramework je možné spustiť bez spusteného servera. Tím dostal za úlohu toto tvrdenie overiť.
    - Ing. Ivan Kapustík navrhol konzultáciu s druhým tímom ohľadne DP.
4. Úloha ROBOCUPTP-36 „Zápisnica“ bola splnená.
5. Úloha ROBOCUPTP-35 „Dokumentácia k prvému šprintu“ bola splnená.

6. Úloha ROBOCUPTP-67 „Nastavenie zdieľania backlogu medzi tímami“ bola splnená. Bola vyriešenia spôsobom, že oba tímy sú v jire v jednom projekte ale každý má svoj vlastný storyboard.
7. Úloha ROBOCUPTP-29 „Rozdeliť ruby na balíky a vytvoriť jeden ukážkový príklad transformácie do javy“ bola splnená.
8. Úloha ROBOCUPTP-30 „transformácia high skilly z ruby do javy“ bola splnená, no existujú isté problémy. Jeden s problémov je, že v ruby existujú jazykové konvencie, ktoré neexistujú v jave a high skilly tieto konvencie používajú. Podľa možností, boli prepísané alebo vynechané tak, aby high skill fungoval. Počas prepisovania tím spozoroval celkovo zlý stav kódu high skillov. Používa sa množstvo zdieľaných premenných, často nebolo jasné čo daná časť kódu robí, prípadne je dôležitá časť kódu zakomentovaná. Veľa krát sú robené zbytočné inicializácie premenných, ktoré boli následne pri výbere lowskillu prepísane. Tiež sa tam realizujú zbytočné výpočty, ktoré sú následne zahodené. Celkovo je kód zle zdokumentovaný. Ing. Ivan Kapustík povedal tímu, aby označovali a popisovali časti kódu. Treba napísať či je funkčný prípadne nefunkčný, čo robí, prípadne že nie je predstava o tom čo to robí. Tím tiež postrehol zlé ukončovanie high skillov, čo spôsobuje volanie plánovača, ktorý znova volá daný high skill. Vzniká tak nekonečná slučka. Bc. Martin Markech informoval o transformovaní settings do javy, tiež podotkol, že sme kompletne vylúčili ruby, tj. ak by sme ho zakomentovali, aplikáciu by bolo možné bežne spustiť.

Po ukončení review úloh bola voľná debata z druhým tímom. Riešil sa git, konkrétne boli objasnené pojmy ako push, pull, merge, rebase, commit, a pod. Ing. Marian Lekavý navrhol metodiku commitovania verzií pre oba tímy. Metodika spočíva v tvorbe nového branch pre riešený task. Tam riešiteľ robí commity, po ukončení tasku urobí merge s master vetvou. Product owners chceli po tíme, aby spísali metodiku práce s gitom.
Stretnutie ďalej pokračovalo prideľovaním úloh. Product owners požadujú, aby sa veci z backlogu priradili na vedúceho s nízkou prioritou a oni tomu následne určia prioritu. Upozornili na povinnosť označiť hotovú úlohu v jire ako „resolved“ a odporučili napísať guidelines ako riešiť statusy v jire.

Nové úlohy pre nasledujúci šprint:

Číslo úlohy	Zhrnutie úlohy	Riešiteľ	Zodpovedný	Stav úlohy
ROBOCUPTP-30	9.1 Verzia servera simulacneho prostredia	Roman Moravčík	Roman Moravčík	Nesplnené
ROBOCUPTP-38	9.4.2 Vytvorit architekturu Highskills	Celý tím	Celý tím	Nesplnené
ROBOCUPTP-39	9.4.3 Navrh a Implementacia strategickej vrstvy	Celý tím	Celý tím	Nesplnené
ROBOCUPTP-93	Opraviť Unit testy	Tomáš Nemeček	Tomáš Nemeček	Nesplnené

Popis s úlohám:

- K úlohe ROBOCUPTP-38 „Vytvoriť architektúru Highskills“ treba aj implementovať nejaký sample a overiť tak, či to tak vôbec pôjde. Pri návrhu treba uvažovať aj

výsledky analýz zahraničných tímov, napr. otáčanie hlavy pre neustále zisťovanie stavu ihriska.

- Ing. Ivan Kapustík predstavil svoj nápad na architektúru. Uviedol 4 hlavné entity, a to stratégia, taktika, high skill a low skill. Ku každému uviedol opis a príklad:
    - stratégia – niečo, čo je plánované dlhodobo, napr. útok po pravom krídle, obrana. Na rozhodnutie stratégie môže byť využitá komunikácia v tíme. Treba riešiť problém, kto volí stratégiu a kedy sa mení.
    - taktika – ako dosiahnuť cieľ stratégie. Rieši aktuálnu situáciu. Môžeme to prirovnať súčasnému plánovaču. Taktika vyberá high skilly ktoré sa použijú na realizáciu.
    - high skill – pomocou série low skills vykonáva akcie robota, napr. pohyb k lopte, kop, otočenie.
    - low skill – reálny atomický pohyb robota
- Ing. Ivan Kapustík oznámil tímu, že netreba riešiť úplne podrobný návrh architektúry, zatiaľ sa treba zamerať na základy, ale zároveň myslieť aj na rozšírenia a doplnenia nových features. K tomuto odporučil napísanie metodiky na pridávanie nových funkcií do architektúry. Návrh musí byť na úrovni rozhraní, ktoré je možné implementovať. Je nutná komunikácia z druhým tímom.
- Product owners požadujú mať základ architektúry hotové na konci tohto víkendu. Oba tímy sa dohodli na stretnutí v nasledujúci deň.

A-7. Zápis z 7. Stretnutia tímu č. 9

Dátum: 13.11.2013
Miestnosť: Jobsové softvérové štúdio (lab 1.31) (FIIT-STU)
Prítomní:

Pedagóg :	Ing. Ivan Kapustík
Členovia tímu:	Bc. Filip Blanárik, Bc. Michal Blanárik, Bc. Štefan Horváth Bc. Štefan Linner, Bc. Martin Markech, Bc. Roman Moravčík Bc. Tomáš Nemeček

Stretnutie viedol: Bc. Tomáš Nemeček
Zápis: Bc. Štefan Linner
Zvolený nasledovný zapisovateľ: Bc. Štefan Linner

Téma stretnutia

Zhodnotenie stavu úloh 3. šprintu a diskusia oboch spolupracujúcich tímov k aktuálnemu návrhu architektúry.

Vyhodnotenie úloh z predchádzajúceho stretnutia

Číslo úlohy	Zhrnutie úlohy	Riešiteľ	Zodpovedný	Stav úlohy
ROBOCUPTP-30	9.1 Verzia servera simulacneho prostredia	Roman Moravčík	Roman Moravčík	Nesplnené

ROBOCUPTP-96	Vytvorenie dokumentácie k druhému šprintu	Celý tím	Michal Blanárik	Splnené
ROBOCUPTP-93	Opraviť Unit testy	Tomáš Nemeček	Tomáš Nemeček	Nesplnené
ROBOCUPTP-94	Architektúra	Celý tím	Roman Moravčík	Nesplnené

Opis stretnutia

16. Prvým bodom stretnutia bolo zhodnotenie miery splnenia úloh (aktuálneho) 3. šprintu.
    a. Úloha ROBOCUPTP-30 nebola splnená, ani otvorená. Jedná sa o priamočiaru úlohu, nízkej priority, ktorá nie je komplikovaná a bude do ukončenia šprintu splnená.
    b. Úloha ROBOCUPTP-96 bola splnená. Dokumentácia k druhému šprintu bola doplnená o analýzu zvyšnej diplomovej práce a postrehy pri reimplementácii high skillov.
    c. Úloha ROBOCUPTP-93 nebola splnená. Bc. Tomáš Nemeček analyzoval jednotlivé unit testy, z čoho vyšlo 10 testov ako chybných, z toho
        i. 3-4 testy sú závislé od premenných servera (úloha ROBOCUPTP-30),
        ii. 4 sa týkajú pozície agenta,
        iii. jeden sa týka testu na jazyk ruby, ktorý už je odstránený,
        iv. a jeden test na metódu „logovania“.
    d. Úloha ROBOCUPTP-94 nebola splnená. Po minulotýždňovom stretnutí v stredu sa obidva spolupracujúce tímy znovu stretli nasledujúci deň vo štvrtok za účelom vypracovania náčrtu návrhu architektúry, ktorý by bol následne do konca týždňa odoslaný Ing. Kapustíkovi na posúdenie. Ing. Kapustík však nasledujúci deň (piatok) poslal obom tímom jeho špecifickejšiu a mierne odlišnú víziu architektúry. Jednotlivé zmeny oproti už navrhnutej architektúre zo štvrtka nebolo kedy odkonzultovať s väčšinou členov oboch tímov. Diskusia k návrhu architektúry bola predmetom ďalšej časti tohto stretnutia.
17. Náš tím následne zhodnotil mieru splnenia úloh druhého tímu, podľa informácií, ktoré od nich máme. Úlohy zahŕňali odstránenie ruby, ktoré bolo odstránené ale ešte nebolo publikované v repozitári, a analýzu zakomentovaného kódu, z ktorej vyplynulo, že väčšina kódu je buď pomocná, t.j. použitá pre rôzne pomocné výpočty alebo kontrolné výpisy, alebo nevyužiteľná.
18. Spomenutý bol aj nasledujúci kontrolný bod (budúci týždeň v stredu), v ktorom sa odovzdáva súčasný stav dokumentácie o inžinierskom diele (po 3. šprinte) a súčasný stav dokumentácie riadenia projektu. Zo stránky predmetu sme nezistili, či musí byť dokumentácia v tomto kontrolnom bode tlačená, Pokyny tlačiť dokumentáciu sú uvedené iba pri kontrolných bodoch až na konci semestra.

Návrh architektúry:

19. V tomto bode Bc. Martin Markech začal prezentovať načrtnutú architektúru vedúcim (produktovým vlastníkom) s pomocou vytvoreného diagramu tried.

a. Prvá otázka Ing. Mariána Lekavého (vedúci druhého tímu, ďalej Marián) znela, či stratégie budú injektovateľné. Naša odpoveď znela, že nad tým sme neuvažovali.
b. Vedúci podotkli, že v návrhu máme znázornené iba dve situácie a je dobre v návrhu jednoznačne znázorniť, že ich bude podstatne viac. Bolo ozrejmené, že situácie budú vracať hodnoty TRUE alebo FALSE. Ing. Ivan Kapustík (ďalej Ivan) vymenoval niekoľko situácií (mám loptu, môžem mať loptu, ...).
c. Marián naznačil, že stratégia by mala mať sadu taktík a konkrétnu taktiku vyberá situácia (táto druhá myšlienka bola neskôr diskutovaná a viac menej vyvrátená). V situácii je nejaký zámer, podcieľ. Ivan naznačil, že v každej taktike by nemala byť implementovaná každá situácia.
d. Marián ešte navrhol, že taktiky by mohli fungovať podobne ako stratégie. Taktika by dostala na vstup napríklad štyri situácie a potom sa rozhodne na základe fitness.
e. Ďalej bolo diskutované o anotáciách. Bolo ozrejmené, že high skill by sa na základe anotácie mal rozhodovať pre špecifický low skill. A z tohto dôvodu by high skill mal od taktiky prijímať parametre, na základe ktorých bude hľadať v anotáciách.
f. Predmetom ďalšej diskusie bola zmena stratégie. Otázkou teda bolo, kedy a ako sa bude stratégia meniť. Stratégia sa mení keď napríklad prehrávame, alebo odhalíme stratégiu protihráča. V prípade, že stratégii sa nedarí spĺňať jej ciele, bude jej postupne znižovaná fitness hodnota. Zmena stratégie môže byť implementovaná pomocou „observer-a".
g. Rovnako bolo diskutované o zmene taktiky. Ivan identifikoval štyri podstatné otázky:
    - Kedy môže začať?
    - Kedy môže skončiť?
    - Kedy nemôže skončiť?
    - Kedy musí skončiť?

    Bolo ozrejmené, že taktika sa nemení nutne v prípade, že sa zmení niektorá situácia. No táto možnosť sa nevylučuje. Treba teda identifikovať prípady kedy sa mení. Môže to byť podobné, ako rozhodovanie o ukončení high skillu. Istá zmena situácií teda môže zmeniť aj taktiku, no nemusí. Taktiku možno chápať aj ako plán. Ak sa úplne niečo pokazí, tak hneď menim taktiku (plán), keď sa niečo pokazí alebo zmení nie až tak dôležité, nemusím hneď meniť taktiku. High skilly teda musia implementovať niečo na ich rýchle a stabilné ukončenie a teda musím ich vedieť ukončiť z taktiky.

    Ani taktika, ani stratégia, by nemala byť vyhodnocovaná v každom takte simulácie. Ideálne by možno bolo napríklad v každom 5-8 takte. Treba teda počítať s viacerými vláknami (taktiky a stratégie), ktoré budú vstupovať do hlavného behu programu (taktu) menej často.

- Marián navrhol, aby sme si vymysleli viaceré scenáre a na tom sa snažili dotvárať a overovať architektúru. Vychádzať môžu napríklad z nasledovných taktík:
    - Chcem obsadiť dvoch hráčov a tretí zoberie loptu
    - Chcem blokovať spoluhráča (keď ma obehne, alebo bude pri ňom niekto iný, menim taktiku)

- Ďalej bol diskutovaný problém s interakciou medzi high skillmi a taktikou. Je otázne kde sa bude riešiť čo. Ak pri kope do lopty hráč spadne, tak chceme aby sa sám high skill postaral o postavenie hráča, alebo to necháme na taktike? Niekedy totiž možno taktika chce reagovať na takýto prípad, a potrebuje napríklad zmeniť high skill (alebo iba jeho parametre), aby zvolil rýchlejší kop, pretože protihráč je už bližšie pri našom hráčovi, ktorý je pri lopte. Boli navrhnuté riešenia napríklad použitím výnimiek, ktoré budú odosielané vyššej vrstve, alebo pridaním ďalšej medzi-vrstvy. Cieľom je zatiaľ iba vytvoriť rozhranie, ktoré bude pripravené na rôzne druhy situácií.

B Vzorové príklady dokumentov

B-1. Príklad zápisnice

V kapitole je uvedený príklad zápisnice zo stretnutia tímu. Všetky oblasti písané červenou farbou slúžia ako vzorová ukážka obsahu danej časti dokumentu.

Zápis z 1. stretnutia tímu č. 9

Dátum: 2.10.2013
Miestnosť: Jobsové softvérové štúdio (lab 1.31) (FIIT-STU)
Prítomní:

Pedagóg :	Ing. Ivan Kapustík
Členovia tímu:	Bc. Filip Blanárik, Bc. Michal Blanárik, Bc. Štefan Horváth Bc. Štefan Linner, Bc. Martin Markech, Bc. Roman Moravčík Bc. Tomáš Nemeček

Stretnutie viedol: Ing. Ivan Kapustík
Zápis: Bc. Martin Markech
Zvolený nasledovný zapisovateľ: Bc. Martin Markech

Téma stretnutia

Zhodnotenie stavu úloh z prvého šprintu, zadefinovanie úloh na ďalší šprint.

Vyhodnotenie úloh z predchádzajúceho stretnutia

Číslo úlohy	Zhrnutie úlohy	Riešiteľ	Zodpovedný	Stav úlohy
ROBOCUPTP-1	Vytvoriť zápisnicu zo sedenia	Martin Markech	Martin Markech	Splnená / Nesplnená

Opis stretnutia

1. Na úvod tím zhodnotil stav úlohy ROBOCUPTP-32 „Hĺbková analýza zahraničných tímov“. Úloha bola splnená, výstupný dokument sa nachádza na webovej stránke tímu. Niektorí z členov tímu zreferovali výsledky svojich analýz.
2. .....
    a. .....

Úlohy do ďalšieho stretnutia

Číslo úlohy	Zhrnutie úlohy	Riešiteľ	Zodpovedný	Body	Čas
ROBOCUPTP-1	Vytvoriť zápisnicu zo sedenia	Martin Markech	Martin Markech	13b	20 hod

Popis s úlohám

- K úlohe ROBOCUPTP-38 „Vytvoriť architektúru Highskills“ treba aj implementovať nejaký sample. Pri návrhu treba uvažovať aj výsledky analýz zahraničných tímov, napr. otáčanie hlavy pre neustále zisťovanie stavu ihriska.

Poznámky

B-2. Príklad dokumentácie

V kapitole je uvedený príklad dokumentácie s titulnou stranou a ukážkou všetkých tipov nadpisov.

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

Fakulta informatiky a informačných technológií

Ilkovičova 2, 842 16 Bratislava 4

Tímový projekt
RoboCup
Názov úlohy, ktorej výsledkom je daný dokument

Bc. Filip Blanárik

Bc. Michal Blanárik

Bc. Štefan Horváth

Bc. Štefan Linner

Bc. Martin Markech

Bc. Roman Moravčík

Bc. Tomáš Nemeček

Tím č. 9:	Gitmen
Vedúci projektu:	Ing. Ivan Kapustík
Predmet:	Tímový projekt I
Ročník:	1
Akademický rok:	2013/2014, zimný semester
Mailový kontakt:	gitmen09@gmail.com

Obsah

1 Nadpis prvej úrovne

1.1 Nadpis druhej úrovne

Nadpis tretej úrovne

Podnadpis

Bežný text – tu je uvedený bežný text
Zvýraznené časti buď

- **Zvýraznený text 1**
- *Zvýraznený text 2*
- Zvýraznený text 3

C Preberacie protokoly

Preberací protokol

Tím č.9: Gitmen

Tímový projekt 2013/2014

Predmet odovzdávania:

- **Dokumentácia k inžinierskemu dielu**
  - opis doteraz vykonanej práce za prvé tri šprinty

- **Dokumentácia k riadeniu projektu**
  - uvažujúc prvé tri šprinty

Vedúci projektu: Ing. Ivan Kapustík

Vedúci projektu podpisom potvrdzuje prebratie vyššie uvedených častí projektu.

V Bratislave

....

Dátum Podpis