Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

Ilkovičova 2, 842 16 Bratislava 4

Trojdimenzionálne UML

Dokumentácia k riadeniu projektu

Tím č. 2

Členovia tímu: Bc. Brndiarová Gabriela, Bc. Kyseľ Peter, Bc. Martoš Ivan,
Bc. Štajer Andrej, Bc. Štetiar Matej, Bc. Šuta Erik,
Bc. Valko Andrej, Bc. Kolačkovský Tomáš (do 6. týždňa ZS)

Študijný program: IS/SI

Ročník: 1. Ing.

Predmet: Tímový projekt

Vedúci tímu: Ing. Polášek Ivan, PhD.

Ak. rok: 2013/14

Obsah

1 Úvod

Tento dokument bol vytvorený ako dokumentácia riadenia k projektu Trojdimenzionálne UML v rámci predmetu Tímový projekt. Štruktúra dokumentu je v súlade so zadanými požiadavkami predmetu.

Táto dokumentácia poskytuje ucelený obraz o členoch tímu. Každému členovi tímu bola vybraná jedna rola v rámci manažmentu projektu, ktorej sa má venovať. Roly nie sú pevne dané a počas semestra sa môžu meniť. Z konkrétnej role následne vyplývajú aj zodpovednosti člena tímu na bezproblémovom chode projekt.

Členovia tímu vytvorili plán, čo má byť súčasťou projektu. Okrem toho načrtli približný časový odhad implementácie jednotlivých častí funkcionality.

Metodiky ako súčasť riadenia projektu slúžia na zjednotenie a zlepšenie kvality práce členov tímu, ktorí nie sú odborníkmi danej oblasti. Ide o formu zdieľania vedomostí a zručností medzi členmi tímu. Každá metodika sa venuje vybranej téme, ktorá bola dôležitá pri práci na projekte.

V prílohách dokumentu sa nachádzajú zápisy zo stretnutí. V zápisoch možno sledovať plnenie plánov a na koľko si jednotliví členovia tímu plnia zverené úlohy.

2 Ponuka

2.1 Predstavenie členov tímu

Gabriela Brndiarová

Téme vizualizácie sa venovala počas svojej bakalárskej práce a pokračuje v tomto smere aj v diplomovej práci. Mimo školy pracovala na niekoľkých webových aplikáciách v J2EE a momentálne pracuje s integračným nástrojom TIBCO.

Tomáš Kolačkovský

Bakalárska práca Redakčný systém pre univerzálne použitie (UniSys). V súčasnosti okrem školy aj pracuje, kde je na pozícii analytik. Venuje sa tvorbe web stránok.

Peter Kyseľ

Počas štúdia skúsenosti hlavne s programovacím jazykom Java a technológiami UML, MySQL. Bakalárska práca zameraná na elektronické skladové hospodárstvo.

Ivan Martoš

Študent softvérového inžinierstva pričom o informačné technológie sa zaujíma vyše 6 rokov, z toho 4 v praxi.

Andrej Štajer

Študent so záľubou pre nové technológie. Aktívne sa venuje objektovo-relačným databázam v podnikovom prostredí ako SQL konzultant.

Matej Štetiar

Počas štúdia si osvojil základy programovacích jazykov Java, C#, C a technológií MySQL, UML. Jeho bakalárska práca bola zameraná na integráciu knižnično-informačného systému a RFID čítačky. Mimo školy sa špecializuje na vývoj enterprise aplikácií v nástroji IBM BPM.

Erik Šuta

Pracuje ako java developer v oblasti open-source identity management. Vo všeobecnosti má veľmi rád nové technológie.

Andrej Valko

Skúsenosti s metódami špecifikácie softvéru, databázovými technológiami, programovacími jazykmi C, Java, a aj s Java EE technológiami a rámcami. Počas bakalárskej práce získal skúsenosti s technológiami sémantického webu. Orientuje sa aj v HTML, CSS, JavaScript-e a Photoshop-e.

2.2 Motivácia

UML, ako dôležitý nástroj softvérového inžiniera, slúži na návrh, špecifikáciu a aj zdokumentovanie vytváraného alebo už existujúceho softvéru. Pri týchto činnostiach sa často stáva, že výsledné diagramy sú veľmi zložité. Preto často dochádza ku kríženiam a prekrývaniu sa častí diagramov. Pridaním jednej dimenzie by sa mohlo množstvo s týchto problémov odstrániť. Zobrazením v trojdimenzionálnom priestore by sa mohla zlepšiť nie len čitateľnosť, ale dali by sa pridať nové pohľady alebo rôzne porovnania viacerých diagramov.

3 Úlohy členov tímu

3.1 Manažérske úlohy členov tímu

Manažérske úlohy boli členom tímu rozdelené na prvom stretnutí. Na druhom stretnutí došlo k prehodnoteniu z dôvodu zmeny predpokladaných používaných technológií. Rozdelenie sa uskutočnilo na základe spoločnej diskusie, kde boli zohľadnené preferencie jednotlivých členov tímu. Výsledok po druhom stretnutí sa nachádza v tabuľke Tabuľka 3.1.

Člen tímu	**Manažérska rola**
Gabriela Brndiarová	Manažér rozvrhu a plánovania
Tomáš Kolačkovský	Manažér monitorovania produktu do 6. týždňa ZS
Peter Kyseľ	Manažér monitorovania produktu
Ivan Martoš	Vedúci tímu; Manažér komunikácie a ľudských zdrojov
Andrej Štajer	Manažér rizík
Matej Štetiar	Manažér kvality
Erik Šuta	Manažér dokumentovania
Andrej Valko	Manažér kvality

Tabuľka 3.1 Priradenie manažérskych úloh členom tímu.

3.2 Hlavné zodpovednosti členov tímu

Tabuľka Tabuľka 3.2 obsahuje hlavné zodpovednosti jednotlivých členov tímu počas celého projektu.

Člen tímu	**Hlavné zodpovednosti**
Gabriela Brndiarová	• správa systému JIRA • Vytváranie reportov na konci šprintov • kontrola zdrojového kódu Mateja Štetiara
Tomáš Kolačkovský	Do 6. týždňa:

	• správa internetovej stránky tímu • správa virtuálneho servera projektu
Peter Kyseľ	• kontrola zdrojového kódu Andreja Valka Od 6. týždňa: • správa internetovej stránky tímu • správa virtuálneho servera projektu
Ivan Martoš	• kontrola zdrojového kódu Andreja Štajera
Andrej Štajer	• kontrola zdrojového kódu Gabriely Brndiarovej
Matej Štetiar	• kontrola zdrojového kódu Petra Kyseľa
Erik Šuta	• správa verziovacieho systému použitím nástroja GIT a Bitbucket • správa dokumentácie • kontrola zdrojového kódu Ivana Martoša
Andrej Valko	• kontrola zdrojového kódu Erika Šutu

Tabuľka 3.2 Hlavné zodpovednosti členov tímu.

3.3 Práca na dokumentácii

V Tabuľka 3.3 možno vidieť prácu jednotlivých členov tímu na projektovej dokumentácii. Položka „Všetci“ v tabuľke označuje, že členovia prispievali ku kapitolám rovnomerne. Táto položka sa týka dokumentácie k šprintom, kde každý člen tímu vypracoval dokumentáciu pre úlohy, na ktorých počas konkrétneho šprintu pracoval.

Člen tímu	**Práca na dokumetácii (zoznam vypracovaných kapitol)**
Gabriela Brndiarová	• Inžinierske dielo: 3 • Riadenie projektu: 3.1, 3.2, 4, 4.1, 4.2, 5.2, 5.4, 8, A.4 • Príprava dokumentov, formátovanie
Peter Kyseľ	• Inžinierske dielo: 2, príloha A • Riadenie projektu: 9, A.5

Ivan Martoš	• Inžinierske dielo: 1.1, 1.2, 1.3, 8.1 • Riadenie projektu: 5.6, 6, A.6
Andrej Štajer	• Inžinierske dielo: 8.2, 11 • Riadenie projektu: 1, 5.3, A.7
Matej Štetiar	• Inžinierske dielo: • Riadenie projektu: 5.1, 10, A.1, A.8
Erik Šuta	• Inžinierske dielo: • Riadenie projektu: 3.3, 5.5, 5.7, 12, A.2, A.9 • Formátovanie dokumentov, integrácia textov od členov tímu, kontrola pravopisu, finalizácia
Andrej Valko	• Inžinierske dielo: 8.2, 8.3 • Riadenie projektu: 5.1, 7, A.3
Všetci	• Inžinierske dielo: 4, 5, 6, 7 • Riadenie projektu: 2.1

Tabuľka 3.3: Práca členov tímu na dokumentácii

4 Plány

V nasledujúcej kapitole sú opísané plány tímu v zimnom a letnom semestri. V zimnom semestri bol vytvorený iba dlhodobý plán na celý semester. Pre letný semester bol okrem dlhodobého plánu vypracovaný aj krátkodobý podrobnejší po jednotlivých šprintoch. Krátkodobý plán je vypracovaný vždy pre najbližšie 3 šprinty.

4.1 Zimný semester

Produktový *backlog*

Na začiatku zimného semestra bol vytvorený zoznam používateľských príbehov, ktoré sa môžu zapracovať do aplikácie – tzv. produktový *backlog*. Jednotlivým používateľským príbehom bola priradená ich priorita vyjadrená v celých číslach, kde 1 predstavuje najvyššiu prioritu. Produktový *backlog* znázorňuje Tabuľka 4.1.

Používateľský príbeh	**Priorita**	**Oopis**
Vytvorenie objektu	1	Pridávanie čiary života reprezentujúcej objekt konkrétnej triedy do diagramu. Do vytvárania objektu je zahrnuté aj vytváranie triedy.
Vytvorenie interakcie	1	Pridávanie interakcie medzi dvoma čiarami života. Možnosť pridať viacero typov interakcií (synchrónne a asynchrónne správy). Interakcia môže spájať aj čiary života, ktoré sa nenachádzajú na rovnakej vrstve.
Serializácia	1	Ukladanie diagramu do súboru a spätné načítavanie zo súboru.
Automatické ukladanie	2	Automatické ukladanie diagramu v pravidelných časových intervaloch.
Vytvorenie fragmentu	2	Pridávanie fragmentu. Fragment bude typov LOOP, OPT, ALT a PARALEL.
Aktivačný blok	2	Pridávanie aktivačného bloku na čiaru života.
Vymazávanie prvkov z diagramu	3	Po vymazaní prvky nie sú zobrazené v diagrame.
Vymazanie prvkov z modelu	3	Po vymazaní prvky nie sú dostupné v repozitáry prvkov.
Strom prvkov	3	Používateľovi by sa zobrazoval strom prvkov diagramu.
Kopírovanie vrstiev	3	Vytvorenie kópie vrstvy aj so všetkými prvkami na nej.
Kopírovanie prvkov diagramu	3	Vytvorenie kópie konkrétneho prvku diagramu. Kópia by sa vložila na rovnakú vrstvu ako jej predloha.
Filtrovanie	3	Filtrovanie zobrazených prvkov diagramu.
Animácie	3	Animácie, ktoré by zvýšili atraktívnosť grafického používateľského rozhrania.
Ovládacie prvky	3	Vylepšenie ovládania aplikácie.
Editovanie veľkosti	3	Zmenšovanie a zväčšovanie prvkov diagramu.

prvkov		
Editovanie vzdialeností medzi vrstvami	3	Zmenšovanie a zväčšovanie vzdialenosti medzi jednotlivými vrstvami.
Editovanie pozície prvkov	3	Presúvanie prvkov diagramu po vrstve aj medzi vrstvami.
Klávesové skratky pre pohyb	3	Možnosť používať klávesové skratky pri práci s aplikáciou. Konkrétne pri označovaní prvkov diagramu a prechodmi medzi vrstvami.
Zobrazenie na celú obrazovku	3	Zobrazenie vrstvy na celú obrazovku.
Ovládanie kamery myšou	3	Pohyb kamerou dopredu a dozadu ovládané kolieskom myši.

Tabuľka 4.1: Produktový *backlog* na začiatku zimného semestra. Úlohy sú zoradene podľa priority.

Dlhodobý plán

Na zimný semester sú naplánované nasledujúce používateľské príbehy:

- Vytvorenie objektu
- Vytvorenie interakcie
- Serializácia
- Vytvorenie fragmentu

Pri poslednom používateľskom príbehu „Vytvorenie fragmentu“ bolo naplánovanie začatie práce a dokončenie v letnom semestri.

4.2 Letný semester

Produktový backlog

V produktovom *backlog-u* boli po zimnom semestri vykonané zmeny a odstránené tie používateľské príbehy, ktoré sa podarilo implementovať.

Pre používateľský príbeh „Vytvorenie interakcie“ boli znížené nároky na riešenie. V novej verzii nie je potrebné mať dva typy správy. Preto bol z produktového *backlog-u* odstránený ako hotový. Používateľský príbeh „Vytvorenie fragmentu“ je rozpracovaný. Po zistení, že súčasné riešenie nemá vytvorený repozitár prvkov a jeho vytvorenie je nad rozsah tohto projektu, bol odstránený aj scenár „Vymazanie prvkov z modelu“.

Produktový *backlog* v súčasnom stave znázorňuje Tabuľka 4.2.

Používateľský príbeh	Priorita	Opis
Automatické ukladanie	2	Automatické ukladanie diagramu v pravidelných časových intervaloch.
Vytvorenie fragmentu	2	Pridávanie fragmentu. Fragment bude typov LOOP, OPT, ALT a PARALEL.
Aktivačný blok	3	Pridávanie aktivačného bloku na čiaru života.
Strom prvkov	3	Používateľovi by sa zobrazoval strom prvkov diagramu.
Kopírovanie vrstiev	3	Vytvorenie kópie vrstvy aj so všetkými prvkami na nej.
Kopírovanie prvkov diagramu	3	Vytvorenie kópie konkrétneho prvku diagramu. Kópia by sa vložila na rovnakú vrstvu ako jej predloha.
Filtrovanie	3	Filtrovanie zobrazených prvkov diagramu.
Animácie	3	Animácie, ktoré by zvýšili atraktívnosť grafického používateľského rozhrania.
Ovládacie prvky	3	Vylepšenie ovládania aplikácie.
Editovanie veľkosti prvkov	3	Zmenšovanie a zväčšovanie prvkov diagramu.
Editovanie vzdialeností medzi vrstvami	3	Zmenšovanie a zväčšovanie vzdialenosti medzi jednotlivými vrstvami.
Editovanie pozície prvkov	3	Presúvanie prvkov diagramu po vrstve aj medzi vrstvami.
Klávesové skratky pre pohyb	3	Možnosť používať klávesové skratky pri práci s aplikáciou. Konkrétne pri označovaní prvkov diagramu a prechodmi medzi vrstvami.
Zobrazenie na celú obrazovku	3	Zobrazenie vrstvy na celú obrazovku.
Ovládanie kamery myšou	3	Pohyb kamerou dopredu a dozadu ovládané kolieskom myši.

Tabuľka 4.2: Produktový *backlog* na začiatku letného semestra

Dlhodobý plán

V letnom semestri sa chceme sústrediť na ovládacie prvky používateľského rozhrania a na dokončenie fragmentu. Z produktového *backlog-u* boli vybrané nasledovné používateľské príbehy:

- Vytvorenie fragmentu
- Automatické ukladanie
- Animácie
- Ovládacie prvky
- Ovládanie kamery myšou

Krátkodobý plán

Šprint č. 1

- vytvorenie generického fragmentu, od ktorého by sme mohli dediť my aj diplomant Matej Škoda
- prebratie vybratých častí fragmentu od Mateja Škodu
- spojenie nášho fragmentu a fragmentu od Mateja Škodu do jedného celku

Šprint č. 2

- automatické ukladanie
- ovládanie kamery myšou

Šprint č. 3

- animácie

5 Manažment tímu

5.1 Manažment kvality

Pre kontrolu kvality nášho systému sme zaviedli prehliadky zdrojových kódov. Kontrola kódu a odobrenie kvality bolo určené ako nutné kritérium pre ukončenie úlohy. Keďže máme v tíme 7 členov, prehliadky kódov sú vykonávané do kruhu. Takto sme eliminovali aj možnosť dohody dvoch členov, ktorá by mohla nastať pri prehliadkach kódov vo dvojici. Prehliadky kódov vykonávame v systéme bitbucket, ktorý nám umožňuje prehľadne sledovať vykonané zmeny a pridávať k nim komentáre. Pri prehliadke kódu sa sleduje, či boli dodržané menné konvencie, či je kód správne komentovaný, upravený, či sa v ňom nenachádzajú žiadne nepresnosti a či je použitý prístup k riešeniu úlohy správny.

Chyby alebo nejasnosti však môžu byť nájdené aj mimo prehliadky kódu. Aby riešenie takýchto chýb a nejasností prebehlo čo najrýchlejšie, metódy a triedy majú v zdrojovom kóde svoje označenie, na základe ktorého je možné určiť okrem popisu aj autora (pri prehliadke je na základe autora verzie jasné, kto zmenu vykonal).

Spolu s prehliadkou kódu prebieha aj používateľské testovanie. Tu si kontrolór kódu stiahne aktuálnu verziu zdrojových kódov so zmenami od riešiteľa úlohy. Skúsi tento kód skompilovať a ako používateľ otestuje, či pridaná funkcionalita funguje korektne. Otestuje popri tom nie len bežný scenár, ale aj hraničné scenáre.

Pre udržiavanie kvality nášho produktu sme na začiatku zvažovali použitie unit testov pre automatické testovanie. Tento prístup sme však po dôkladnejšej analýza zamietli, lebo písanie unit testov pre grafický software je zložité a časovo náročné. Preto sme ostali len pri testovaní v rámci prehliadok kódu.

Z dôvodu hromadenia chýb sme sa rozhodli do každého šprintu pridať čas vyhradený na ich riešenie. Takýmto spôsobom sa nám darí opravovať chyby, ktoré sa objavia počas používania produktu alebo počas implementácie.

Zvýšenie kvality sme sa pokúsili dosiahnuť aj pri dokumentovaní. Aby dokumentácia k inžinierskemu dielu opisovala riešenú úlohu čo najlepšie, každý člen ju musí napísať hneď po dokončení práce na danej úlohe. Úlohu považujeme za uzatvorenú len v tom prípade, keď je aj riadne zdokumentovaná.

V priebehu projektu nastalo niekoľko situácií, pri ktorých sme museli vykonať rozsiahlejšie zmeny v zdrojovom kóde, ktoré síce neprinášali novú funkcionalitu, ale pre zvýšenie kvality boli nevyhnutné. Jedným z takých refaktorovacích prípadov bolo premiestnenie „include“ direktív z hlavičkových súborov do zdrojových súborov, čím sa nám podarilo zrýchliť kompiláciu a zostavovanie aplikácie.

Do budúcnosti sme naplánovali zmenu implementácie časti aplikácie týkajúcej sa vykresľovania fragmentu v 3D diagrame. Keďže ide o projekt, ktorý sa vyvíja vo viacerých paralelných vetvách a v budúcnosti majú byť tieto vetvy zlúčené, bolo potrebné naplánovať refaktorovanie nami navrhovanej implementácie fragmentu v 3D diagrame tak, aby túto časť bolo možné v budúcnosti jednoducho zlúčiť s inouvývojovou vetvou projektu.

5.2 Manažment monitorovania projektu

Manažment monitorovania si počas celého semestra všímal nepriaznivý fakt, že členovia tímu nestíhajú vypracovať svoje úlohy do skončenia šprintu. Počas celého semestra bola snaha tento problém eliminovať. V prvom šprinte bola ako problém označená zlá granularita rozdelených úloh. Dohodlo sa, že úlohy budú rozdeľované na omnoho menšie časti, ktoré si môžu medzi sebou členovia tímu lepšie prerozdeliť.

V rámci monitorovania projektu sa v druhom šprinte začali vypracovávať reporty k projektom (aj spätne pre prvý šprint). Tieto reporty obsahujú informácie o tom kto robil na akých úlohách, koľko hodín a v akom stave sú jednotlivé úlohy po ukončení šprintu. Okrem informácií pre pedagogického vedúceho o práci v tíme poskytujú reporty užitočné informácie aj nám. Prostredníctvom nich si sumarizujeme v akom stave máme na konci šprintu jednotlivé úlohy a čo si ešte vyžaduje našu pozornosť.

Od prvého šprintu sa na monitorovanie úloh používal nástroj JIRA. Na začiatku slúžil len na zadávanie úloh, vedenie údajov o tom koľko času kto strávil na úlohe a komentovanie úloh. V treťom šprinte sme začali využívať aj podporné nástroje, ktoré JIRA pre agilné metódy vývoja ponúka. Bol vytvorený *burndown chart*, čo si vyžadovalo sofistikovanejší spôsob zapisovania úloh do nástroja JIRA (značkovanie úloh, dodržiavanie životného cyklu úlohy, zadávanie blokovaných úloh). *Burndown chart* nám ukázal ďalšie nedostatky, ktoré spôsobujú nedodržiavanie termínov. Jedným z problémov bolo pridávanie úloh počas behu šprintu. Rozhodli sme to eliminovať tak, že na opravu chýb, ktoré sa vyskytnú sa naplánoval čas a do plánovania bol zahrnutý aj čas na kontrolu kódu a písanie dokumentácie.

Problémom bolo aj ustráženie vykonávania revízii zdrojových kódov a dokumentovania úloh po ich vyriešení. Preto sa zaviedol nový spôsob akým sa hodnotil stav úlohy. Úloha nebola označená za vyriešenú, kým nebola skontrolovaná a zdokumentovaná. Toto motivovalo samotných členov tímu, aby si zdokumentovali svoju úlohu a dohliadli na to, že ich kód bude skontrolovaný.

Medzi ďalšie monitorovacie činnosti, ktorým sa pravidelne venujeme patrí retrospektíva na konci šprintu, ktorú zahŕňame do zápisnice. Pomocou hodnoty *work ratio*, ktorú nám ponúka JIRA vieme o častiach, ktoré prekračujú plánovaný počet hodín. *Work ratio* predstavuje funkciu odhadovaného a stráveného času na úlohe. K týmto úlohám pristupujeme špecificky, ich vyriešenie berieme ako veľkú prioritu a snažíme sa odhaliť problémy, pre ktoré táto situácia nastala, aby sme sa z nich mohli do budúcna poučiť.

Venujeme sa aj neformálnym spôsobom monitorovania. Na neformálnych aj formálnych stretnutiach sa špeciálne pýtame na problémy v tíme. Je v silnom záujme všetkých členov aby sme si navzájom pomáhali a aby sa problémom nevenoval nikto ako jednotlivec.

5.3 Manažment rizík a rozsahu

Analýza rizík

Súčasťou manažmentu tímu v tímovom projekte je aj manažment rizík. V tímovom projekte sú však isté špecifiká, ktoré ovplyvňujú možné reakcie na riziká. Jednotlivé riziká boli identifikované a analyzované z pohľadu zodpovednosti, dopadu, podmienok vzniku a možnej prevencie pred rizikom (Tabuľka 5.1).

Riziko nedodania požadovanej funkcionality	
Zodpovedná osoba	Všetci
Popis rizika	Bude dodaný prototyp, ktorý nie je v súlade s predstavami zadávateľa alebo neobsahuje požadovanej funkcionalitu
Časové ohraničenie	Dlhodobé
Pravdepodobnosť vzniku	Nízka
Spúšťače	Zmeny dlhodobých plánov, zadávanie úloh nad schopnosti tímu, vyčerpanosť tímu
Prevencia	Plnenie plánu postupne, časté stretnutia s dodávateľom a jeho informovanosť
Náprava	-
Nesplnenie plánu šprintu	
Zodpovedná osoba	Manažér monitorovania, manažér plánovania
Popis rizika	Naplánovaný šprint nebude splnený v plnom rozsahu
Časové ohraničenie	Dlhodobé

Pravdepodobnosť vzniku	Stredná
Spúšťače	Nerovnomerné pálenie „burn-down chart“-u, neplnenie úloh jednotlivými členmi tímu
Prevencia	Plnenie plánu postupne, realistické plánovanie, kompromis medzi požiadavkami zadávateľa a schopnosťami tímu
Náprava	-
Riziko nesprávneho ohodnotenia úlohy	
Zodpovedná osoba	Manažér plánovania, jednotliví členovia tímu
Popis rizika	Úloha je podhodnotená a je jej naplánovaný nízky časový rozsah
Časové ohraničenie	Dlhodobé
Pravdepodobnosť vzniku	Vysoké(Stredné)
Spúšťače	-
Prevencia	Rozdeľovanie úloh na menšie celky, odhady pomocou Planning pokru
Náprava	Naplánovanie dopracovania úlohy, poučenie pri hodnotení úloh
Riziko nesplnenia zadanej úlohy	
Zodpovedná osoba	Všetci
Popis rizika	Člen tímu nestihne dodať zadanú úlohu
Časové ohraničenie	Krátkodobé
Pravdepodobnosť vzniku	Stredné
Spúšťače	Odkladanie úloh blízko ku dátumu odovzdania, vyčerpanosť člena tímu, podhodnotená alebo podcenená úloha
Prevencia	Priebežná práca, komunikácia medzi členmi tímu, plánovanie s rezervou na nečakané udalosti, rozdeľovanie úloh na menšie celky
Náprava	Naplánovanie dopracovania úlohy, poučenie pri hodnotení úloh
Riziko nesplnenia blokovanej úlohy	
Zodpovedná osoba	Všetci
Popis rizika	Člen tímu nestihne dopracovať úlohu, ktorá blokuje úlohu ďalšieho člena tímu
Časové ohraničenie	Krátkodobé
Pravdepodobnosť vzniku	Nízke
Spúšťače	Odkladanie úloh blízko ku dátumu odovzdania, nízka komunikácia medzi členmi tímu
Prevencia	Plánovanie závislých úloh do rôznych šprintov, komunikácia členov tímu a ich vzájomné poháňanie
Náprava	Naplánovanie dopracovania úlohy
Neschopnosť implementácie úlohy	
Zodpovedná osoba	Všetci
Popis rizika	
Časové ohraničenie	Dlhodobé
Pravdepodobnosť vzniku	Nízke
Spúšťače	Odkladanie úloh blízko ku dátumu odovzdania, nízka komunikácia medzi členmi tímu
Prevencia	Plánovanie závislých úloh do rôznych šprintov, komunikácia členov tímu a ich vzájomné poháňanie

Náprava	Naplánovanie dopracovania úlohy
Nekonzistencia modelov	
Zodpovedná osoba	Architekt tímu
Popis rizika	Architekt nezapracuje zmeny do dátového modelu projektu
Časové ohraničenie	Strednodobé
Pravdepodobnosť vzniku	Nízke
Spúšťače	Neznalosť návrhu, návrhy vytvorené inou osobou ako architektom
Prevencia	Konzultovanie a analýza návrhov s architektom tímu
Náprava	Doplnenie a oprava modelu
Nevhodnosť člena tímu na danú manažérsku rolu	
Zodpovedná osoba	Vedúci tímu
Popis rizika	Člen tímu si nezvláda plniť manažérsku rolu
Časové ohraničenie	Dlhodobé
Pravdepodobnosť vzniku	Vysoká
Spúšťače	Oneskorenie odovzdaní dokumentov, zlá kvalita vypracovaných dokumentov
Prevencia	Komunikácia vedúceho s členmi tímu či zvládajú prácu
Náprava	Pomoc členovi tímu s jeho úlohou, výmena rolí podľa dohody
Plnenie úloh, ktoré neboli naplánované prioritnejšie	
Zodpovedná osoba	Všetci
Popis rizika	Člen tímu vypracuje úlohu, ktorá nie je naplánovaná a nezostane mu čas na naplánované úlohy
Časové ohraničenie	Dlhodobé
Pravdepodobnosť vzniku	Stredná
Spúšťače	Nechuť člena tímu pracovať na pridelených úlohách, vlastná motivácia riešiť iné úlohy
Prevencia	Splniť prioritne naplánované úlohy
Náprava	-

Tabuľka 5.1: Analyzované riziká

5.4 Manažment rozvrhu a plánovania

V prvom šprinte bol vytvorený produktový *backlog,* o ktorý sa manažment plánovania celý semester výrazne opieral. Používateľské príbehy v ňom boli pri vytvorení ohodnotené prioritou. Tento produktový *backlog* sa nachádza v kapitolách 4.1 a 4.2 v rámci plánovania. Na rovnakom stretnutí ako produktový *backlog* vznikol aj neformálny dlhodobý plán na zimný semester, ktorý je tiež popísaný v kapitole 4.1.

Plánovanie jednotlivých šprintov prebiehalo vždy na stretnutí na začiatku šprintu. Na začiatku projektu mal pri plánovaní hlavné slovo vlastník produktu. V neskorších šprintoch sme sa ako tím do plánovania zapájali stále aktívnejšie, čo vyplývalo aj s nadobudnutia bližších vedomostí

o metodike *scrum*, ktorá plánovanie šprintu opisuje ako dohodu vlastníka produktu s členmi tímu. V zimnom semestri sa nikdy neplánoval viac ako jeden nasledujúci šprint.

Odhadovanie času na jednotlivé úlohy sa postupne vyvíjalo. Kým prvý šprint jednotlivé úlohy nemali pridelený časový odhad, od druhého sa prešlo k odhadovaniu zložitosti. Zložitosti sú vyjadrené pomocou veľkostí tričiek (XS, S, M, L, XL) a na základe nich sa vypočítavajú časové odhady (XS = 1h, S = 2h, M = 4h, L = 8h, XL = 16h). Zložitosti sa úlohám najprv priraďovali na základe dohody členov tímu. Odhadovanie času potrebného na vypracovanie úloh sa vo štvrtom šprinte prvýkrát určoval pomocou techniky zvanej *planning poker cards*.

Na konci zimného semestra bol dlhodobý plán formálne zapísaný do dokumentácie, vznikol krátkodobý plán na prvé tri šprinty letného semestra a bol aktualizovaný produktový *backlog*.

5.5 Manažment podpory vývoja a integrácie

V kontexte podpory vývoja produktu za účelom verziovania tím používal distribuovaný verziovací nástroj git s online úložiskom Bitbucket. Niektoré náležitosti verziovania a vykonávané procesy sú opísané v metodike „*Manažment verziovania zdrojových súborov*“. Každá vývojárom vytvorená verzia bola do repozitára pridaná spoločne so správou o verzii (*commit message*), ktorej formát je ucelený. Repozitár nachádzajúci sa na Bitbucket-e bol integrovaný so systémom na správu úloh Jira a výhody tohto prepojenia členovia tímu využívali pri kontrolách kvality a revíziach zdrojových súborov (Každá úloha zaznamenaná v systéme Jira obsahuje zoznam odkazov na verzie v systéme Bitbucket a každá správa o verzii v systéme Bitbucket obsahuje odkaz na príslušnú úlohu).

V rámci verziovania sme využívali populárny model vetvenia vývoja, každá implementovaná funkcionalita sa vytvárala v samostatnej vetve vývoja a po úspešnej validácii kvality bola integrovaná s projektom. Tento postup bol zvolený najmä z toho dôvodu, že na projekte pracuje okrem tímu aj niekoľko bakalárov a diplomantov. Ak by tí mali potrebu využiť niektorú z implementovaných funkcionalít, stačí v ich vetvách vývoja integrovať niektorú z vetiev vytvorenú naším tímom. Počas vytvárania funkcionality v jednotlivých funkcionálnych vetvách vývoja sme volili prístup častého vytvárania verzií a častej aktualizácie vývojovej vetvy o obsah z hlavnej vývojovej vetvy tímu, čím sa nám darilo predchádzať konfliktom počas integrácie vetvy vývoja do produktu.

5.6 Manažment komunikácie a ľudských zdrojov

Pre výmenu informácií boli použité nástroje a služby Skype, Google Groups a e-mailová komunikácia.

Neformálne formy komunikácie

V nástroji Skype prebiehala neformálna komunikácia medzi členmi tímu. Táto komunikácia prebiehala v spoločnej diskusii (pre všetkých členov tímu) a aj v samostatných diskusiách medzi jednotlivými členmi. V tomto nástroji prebiehala taktiež neformálna komunikácia s autorom prototypu.

Taktiež pomocou Google Group prebiehala neformálna komunikácia. V Google Group prebiehala táto komunikácia medzi členmi celého tímu, nebolo možné tu komunikáciu oddeľovať pre samostatné rozhovory.

Dôležitou zložkou neformálnej komunikácie boli osobné stretnutia. Tieto stretnutia mohli byť v rámci celého tímu, alebo jeho časti.

Efektívnosť neformálnej komunikácie bola veľmi vysoká, pričom z toho dôvodu to bola taktiež najpoužívanejšia a najpreferovanejšia forma komunikácie. Najpreferovanejším spôsobom boli osobné stretnutia, pričom najpoužívanejším bola komunikácia cez Skype.

Formálne formy komunikácie

Medzi formálne formy komunikácie sa radí e-mailová komunikácia, Google Group, pravidelné stretnutia tímu s vedúcim a taktiež aj zápisnice. Tieto formy komunikácie boli formálne vždy keď sa ich zúčastnil vedúci tímu.

Medzi najdôležitejšie formy tejto komunikácie patrili zápisnice a stretnutia s vedúcim tímu. Stretnutia boli dôležité z hľadiska budúceho smerovania tímu, pričom zápisnice uchovávali obsah týchto stretnutí. Formálna a aj výpovedná úroveň zápisníc bola veľmi vysoká. Bol zavedený presný formát každej zápisnice, ktorý sa následne striktne dodržiaval.

Efektívnosť tejto formy komunikácie bola na strednej úrovni a z toho dôvodu nebola medzi členmi tímu preferovaná.

V rámci formálne formy komunikácie bola zohľadnená aj dokumentácia na ktorú sa dávalo veľkého významu. Bolo to z dôvodu zachovania čo najlepšej úrovne projektu.

Pravidlá komunikácie

Pri komunikácií bolo pravidlom uviesť aktuálny problém ktorý sa rieši, číslo task-u v JIRA-e pod ktorý problém spadá a popis problému. Taktiež bolo vítané pokiaľ autor problému prišiel s navrhovaným riešením, ktoré chcel prekonzultovať.

Rozdeľovanie úloh členom tímu

Počas rozdeľovania úloh ktoré boli naplánované pre aktuálny šprint sa rozdeľovanie riadilo na základe preferencií každého člena tímu. Tieto preferencie sa vytvárali primárne podľa znalostí člena, avšak stávalo sa že preferencie člena boli ovplyvňované jeho túžbou pre spoznanie novej znalosti ktorej naštudovanie bolo potrebné pre vyriešenie danej úlohy.

Rozdeľovanie úloh prebiehalo v kolách, pričom v každom kole si musel každý člen tímu vybrať práve jednu úlohu z množiny ponúkaných. Počet kôl závisel od počtu úloh, pričom proces bol ukončený keď všetky úlohy boli pridelené.

Riešenie problémov v time

Pokiaľ nastal problém pre člena tímu, daný člen vždy tento problém oznámil buď celému tímu, alebo jednotlivej osobe podľa jeho preferencií. Následne sa o probléme diskutovalo a pokiaľ člen tímu zažiadal o pomoc, tak akýkoľvek iný člen tímu mu mohol s daným problémom pomôcť. Mohol to urobiť tak, že mu danú úlohu vyrieši, avšak preferovaný spôsobom bola výpomoc a konzultácia pre nájdenie a odporučenie riešenia problému.

5.7 Manažment tvorby dokumentácie

Dokumentácii sa v tíme prikladal od začiatku práce na projekte veľký význam. V kontexte manažmentu dokumentácie vykonávame nasledujúce činnosti.

Dokumentácia zdrojových kódov

Každý člen tímu dokumentuje ním vytvárané zdrojové súbory podľa procesov definovaných v metodike „*Dokumentácia v zdrojových kódoch*“.

Projektová dokumentácia

Počas plánovania šprintov bol vždy vyčlenený čas aj na dokumentáciu. V každej úlohe, ktorá má význam pre používateľa a teda je potrebné ju zdokumentovať, bol na tento fakt braný ohľad a čas potrebný na vytvorenie tejto dokumentácie bol zahrnutý v časovom odhade úlohy. Na konci každého šprintu prebehlo komplexné zlučovanie dokumentácie k úlohám a jej pričlenenie

k dokumentu o inžinierskom diele. Opis jednotlivých šprintov teda obsahuje dokumentáciu k jednotlivým úlohám, na ktorých členovia tímu pracovali. Po treťom šprinte sme začali do dokumentácie k riadeniu projektu, špecificky k zápisom zo stretnutí prikladať aj časť o vykonávaní retrospektívy.

Okrem dokumentácie jednotlivých úloh členovia tímu priebežne vytvárali aj ostatné časti dokumentu podľa požiadaviek definovaných na webovej stránke predmetu a vedúceho projektu, pričom autorstvo je uvedené v kapitole 3.2.

Na tvorbu dokumentácie používame program Microsoft Word 2007. Webové sídlo projektu je po každom stretnutí aktualizované (Za prípadné neaktualizovanie webového sídla z dôvodu nefunkčnosti servera počas semestra sa tím ospravedlňuje).

6 Metodika #1 – Dokumentácia v zdrojových kódoch

Ivan Martoš

6.1 Vymedzenie rozsahu

Táto metodika sa zameriava na dokumentáciu v zdrojových „.cpp“ a hlavičkových „.h“ súboroch vytváraného programu. Popisuje predpísanú štruktúru a obsah komentárov. V súboroch je potrebné komentovať premenné, triedy, metódy a taktiež aj jednotlivé bloky kódu ktoré by mohli byť náročnejšie na pochopenie, prípadne ich štruktúra je veľmi zložitá. Je tu popísané kedy písať dokumentáciu, za akým účelom a ako ju štruktúrovať.

Metodika taktiež popisuje prácu s nástrojom ktorý slúži ako pomôcka pri vytváraní komentárov.

Táto metodika neslúži na zadefinovanie formy a obsahu business dokumentácie ani dokumentácie pre používateľa. Dokumentácia popísaná v tejto metodike je technická, developerská, ktorá slúži pre vývojárov programu.

6.2 Dedikácia metodiky

Táto metodika je určená pre vývojárov programu. Je písaná za účelom jednotnej štruktúry a formy dokumentácie v zdrojových súboroch, ktorá sa píše z dôvodu lepšieho pochopenia funkcionality zdrojového kódu, účelu tried, práce metód, obsahu atribútov tried a premenných a vysvetlenia účelu jednotlivých častí zdrojového kódu. Jednotná forma dokumentácie popísaná v tejto metodike pomôže k rýchlejšiemu zorientovaniu sa v zdrojovom kóde a taktiež ku čistejšiemu kódu ktorý nebude obsahovať rôzne formy komentárov.

Každú zmenu v kóde treba zdokumentovať komentovaním z dôvodu prehľadnosti a aj udržateľnosti kódu. Kód ktorý nie je zdokumentovaný je veľmi ťažké udržiavať.

Role, ktoré sa riadia metodikou

Rola ktorá sa metodikou riadi	Kedy sa daná rola riadi metodikou
Programátor	Programátor sa metodikou riadi v prípade vývoja nových častí programu, modifikáciách existujúcich častí kódu, prípadne len prehliadke implementovaných súčastí zdrojového kódu.
Code reviewer	Code reviewer sa metodikou riadi v prípade revízie kódu za účelom lepšieho pochopenia implementovanej funkcionality.

Tabuľka 6.1 Roly riadiace sa metodikou

6.3 Role a zodpovednosti účastníkov

Programátor

Každý programátor je zodpovedný za dokumentovanie ním napísaného kódu. Táto dokumentácia musí byť napísaná pri vytvorení, respektíve počas vytvárania kódu. Musí tak učiniť najneskôr po dopísaní kódu a predtým ako ukončí svoju aktuálnu programátorskú prácu na projekte. Programátor je povinný dokumentáciu vytvárať priebežne, pričom sa pri vytváraní z dôvodu konzistencie musí riadiť touto metodikou. Dokumentácia sa musí vytvárať pri vytvorení súboru, deklarácií a implementácií triedy, metódy, atribútov a iných častí kódu.

Code reviewer

Code reviewer zodpovedá za review kódu, pričom pri tejto činnosti je potrebné kontrolovať aj obsah, formu a existenciu dokumentácie.

6.4 Zoznam nadväzujúcich metodík a dokumentov

Metodika nadväzuje na metodiku vytvorenú Petrom Kyseľom – „Zdieľanie zdrojov pomocou Google Drive“. Kedže konečná dokumentácia bude uložená na Cloud úložisku Google Drive, postup sprístupnenia samotnej dokumentácie je zapísaný v spomínanej metodike.

Táto metodika sa využije taktiež pri metodike Andreja Valka – „Metodika vytvárania unit testov“. Kedže vytváranie unit testov súvisí s programovaním a tieto testy môžu byť náročné na

pochopenie, je potrebné pre nich vytvoriť taktiež komentáre. Postup vytvorenia komentárov v tejto metodike si nájde uplatnenie v metodika Andrej Valka.

6.5 Vymedzenie pojmov

V tejto metodike sú použité nasledovné pojmy:

- Zdrojový kód, kód
  Zdrojovým kódom sa označuje akýkoľvek kompletný obsah súboru programu ktorého prípona je „.cpp“ alebo „.h“. V zdrojovom kóde sú deklarované a implementované triedy, atribúty tried, metódy a premenné.
- Hlavičkový súbor
  Hlavičkovým súborom sa označuje akýkoľvek kompletný obsah súboru programu ktorého prípona je „.h“. Takýto súbor obsahuje deklarované triedy a atribúty a metódy tried. Obsahuje taktiež aj zadefinované konštanty.
- Komentár (Comment)
  Komentárom (anglicky Comment) sa označuje akákoľvek súvislá časť zdrojového kódu ktorá slúži na opis funkcionality zdrojového kódu alebo jeho časti. Komentár môže byť :
  1. Jednoriadkový – v tomto prípade komentár sa začína dvoma za sebou idúcimi neoddelenými znakmi „/“ a končí koncom riadka v ktorom komentár začal. Tento komentár slúži na rýchle objasnenie funkcionality.
     Príklad: „// toto je komentár“
  2. Jedno a viacriadkový (blokový) – v tomto prípade rozsah komentára je na jeden alebo viac riadkov. Tento komentár môže byť dvoch typov :
     a. Javadoc komentár – Komentár ktorý začína znakmi „/**“ a končí znakmi „*/“. Tento komentár slúži na komentovanie metód, tried a premenných. Môže obsahovať taktiež anotácie.
        Príklad: „/** Javadoc komentár */
     b. Vysvetľovací komentár – Komentár ktorý začína znakmi „/*“ a končí znakmi „*/“. Tento komentár slúži na objasnenie určitej časti kódu, zväčša rozsiahlejšej, prípadne komplikovanejšej.
        Príklad: „/* V tomto komentáre objasním časť kódu */“
- Anotácia
  Anotáciou sa myslí časť Javadoc komentára ktorá je na samostatnom riadku. Začína sa znakom „@“ za ktorým bez medzery nasleduje slovo označujúce typ anotácie. Následne nasleduje medzera a text danej anotácie. Existujú rôzne typy anotácií v Javadoc komentároch, pričom v tomto projekte sú použité pri dokumentácií nasledovné
  1. „author“ – tento typ slúži na popis autora danej časti kódu. Príklad – „@author Ivan“
  2. „see“ – tento typ slúži ako odkaz na inú časť dokumentácie. Príklad – „@see Layer“
  3. „param“ – tento typ slúži pre označenie parametra metódy. Pokiaľ má metóda viacero parametrov, sú tieto anotácie zapísané v takom poradí v akom sú deklarované

parametre metódy. Za označením anotácie nasleduje typ parametra, jeho názov oddelené medzerou a popis parametra. Príklad:
„@param int num1 First parameter of method

@param char var1 Second parameter of method“

4. „return“ – tento typ slúži na popis hodnoty ktorú vracia metóda.
   Príklad: „@return Sum of parameters“
5. „exception“ (alebo „throws“) – tento typ slúži na popis výnimky ktorú metóda vracia.
   Príklad: „@throws IllegalArgumentException Parameter not in range“

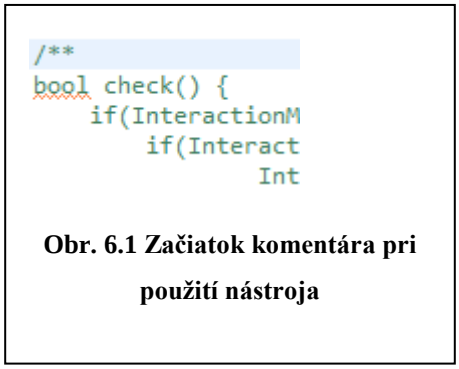
- Eclipse
  Pod Eclipse sa myslí textový editor vývojového prostredia Eclipse IDE for C/C++ Developers v ktorom je program vyvíjaní

6.6 Použítý nástroj

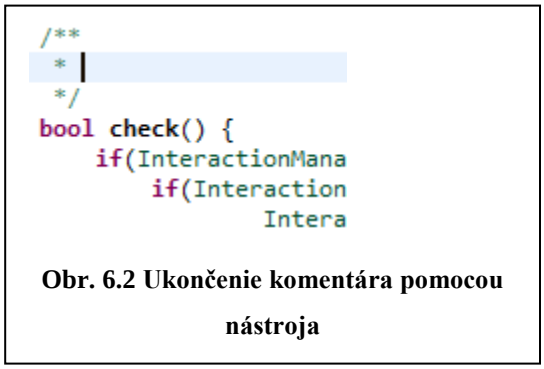
Vývoj programu prebieha vo vývojovom prostredí Eclipse IDE for C/C++ Developers. Toto prostredie obsahuje textový editor, ktorú okrem iného slúži aj pre potreby tejto metodiky.

Pri začatí písania blokového komentára a stisnutí tlačidla „Enter“ na klávesnici ja tento komentár dokončený.

Príklad:



Obr. 6.1 Začiatok komentára pri použití nástroja



Obr. 6.2 Ukončenie komentára pomocou nástroja

Tento nástroj taktiež napomáha ku vizuálnemu odlíšeniu komentára od iných častí kódu. Ako je vidno na obrázkoch, komentáre sú vyznačené zelenou farbou.

6.7 Opis postupov

Presný čas v ktorom komentár zdrojového kódu vytvorený nie je určený. Je ho avšak potrebné vytvoriť do ukončenia aktuálnej programátorskej činnosti. Je to z toho dôvodu, že každý

programátor radšej píše komentár v inom čase, prípadne napísaný kód sa môže časom zmeniť. Avšak vždy je potrebné vytvoriť komentár pre vytvorenú časť kódu.

Komentáre sú napísané výlučne v anglickom jazyku.

Kde písať jednotlivé druhy komentárov a ich štruktúra:

Pri používaní Javadoc komentára je komentár napísaný bezprostredne nad deklaráciou elementu pre ktorý je tento komentár určený. Pokiaľ je element deklarovaný v Hlavičkovom súbore, tak komentár je taktiež v hlavičkovom súbore. Pokiaľ tento Javadoc obsahuje anotácie, tak tie sú umiestnené v dolnej časti komentára. Pre niektoré komentáre sú určité anotácie povinné, avšak použitie ostatných anotácií je na zvážení autora. V prípade anotácií typu „see“ je použitie vždy na zvážení autora a nie je nikdy predpísané. V hornej časti komentára sa nachádza opis elementu ktorý popisuje. Anotácie a opis elementu sú v komentáre oddelené prázdnym riadkom.

Pri používaní jednoriadkového komentára je komentár umiestnený napravo od funkcionality ktorú popisuje, pričom od funkcionality je oddeleným minimálne jedným tabulátorom. Avšak pri vzniku príliš dlhého riadku (viac ako 150 znakov na riadok), je potrebné aby bol tento komentár bezprostredne nad funkcionalitou ktorú popisuje.

Pri používaní vysvetľovacieho komentára sa komentár nachádza bezprostredne nad kódom, alebo blokom kódu (napríklad cyklus) ktorý komentár komentuje.

Pri používaní akéhokoľvek blokového komentára, prvý riadok obsahuje len začiatočné znaky komentára „/**“, alebo „/*“. Na nasledujúcich riadkoch je obsah komentára, pričom každý riadok sa začína znakmi „*“ za ktorým nasleduje medzera. Tieto dva začiatočné znaky vytvorí použitý nástroj. Komentár sa končí ukončovacími znakmi „*/“ ktoré sa nachádzajú taktiež na samostatnom riadku.

V prípade že komentár sa nachádza nad kódom ktorý popisuje, tak jeho ľavé odsadenie od začiatku súboru je rovnaké ako odsadenie kódu ktorý popisuje.

Príklad:

```
/**
 * This comment describes class InteractionManager and it's purpose
 *
 * @author Shade
 */
class InteractionManager
{
    public:
```

Obr. 6.3 Ukážka pozície komentára

Nasleduje opis jednotlivých postupov komentovania pri vytvorení rôznych častí kódu:

Vytvorenie súboru

Vstupné podmienky	Nový súbor je vytvorený
Výstupné podmienky	Komentár v hlavičke nového súboru je vytvorený a kompletný
Povinné anotácie	žiadne

Tabuľka 6.2 Podmienky a anotácie pri vytvorení súboru

Pri vytvorení nového súboru pomáha pri komentovaní použitý nástroj. Pri vytvorení nového súboru obsahujúceho zdrojový kód, nástroj automaticky vytvorí blokový vysvetľovací komentár umiestnených na vrchu novo vytvoreného súboru. Tento komentár obsahuje názov súboru, dátum kedy bol súbor vytvorený a autora vytvoreného súboru.

Možné je taktiež do tohto komentára dodať čo súbor obsahuje a za akým účelom bol vytvorený, avšak táto možnosť je čisto voliteľná a na zváženie autora súboru.

Príklad:

InteractionManager.cpp | InteractionMana

```
1  /*
2   * InteractionManager.cpp
3   *
4   *  Created on: 25.6.2013
5   *      Author: Schade
6   */
7
8  #include "InteractionManager.h"
```

Obr. 6.4 Ukážka komentára vytvoreného súboru

Vytvorenie triedy

Vstupné podmienky	Nová trieda je deklarovaná
Výstupné podmienky	Existuje Javadoc komentár pri deklarácií triedy
Povinné anotácie	author

Tabuľka 6.3 Podmienky a anotácie pri vytvorení triedy

Nad každou deklaráciou triedy musí byť Javadoc blokový komentár ktorý popíše vytvorenú triedu. Tento komentár musí obsahovať anotáciu typu „author“ ktorá identifikuje autora triedy. Javadoc komentár musí obsahovať popis triedy pre ktorú je vytvorený, jej účel a opodstatnenie.

Príklad : viď. príklad ku Kapitole 7.1.1

Vytvorenie atribúty triedy

Nad každou deklaráciou atribútu triedy musí byť Javadoc blokový komentár ktorý popíše vytvorenú atribút. Tento komentár nemusí obsahovať anotácie, avšak môže obsahovať anotácie typu „author“ ktorá identifikuje autora triedy a „see“ v prípade že pre pochopenie atribútu je odporúčané prezretie iného komentára. Javadoc komentár musí obsahovať popis atribútu, jeho účel a akú hodnotu uchováva.

Príklad:

```
64        /**
65         * Camera that represents view of the user
66         */
67        Ogre::Camera* camera;
```

Obr. 6.5 Príklad komentára pri atribúte triedy

Vytvorenie metódy

Deklarácia

Vstupné podmienky	Nová metóda je deklarovaná
Výstupné podmienky	Existuje Javadoc komentár pri deklarácií metódy

Povinné anotácie	author, return (pokiaľ nie je typu „void“), param (pre každý parameter), throws (pokiaľ hádže výnimku)

Tabuľka 6.4 Podmienky a anotácie pri vytvorení metódy

Nad každou deklaráciou atribútu triedy musí byť Javadoc blokový komentár ktorý popíše vytvorenú metódu. Je potrebné zhrnúť aký je účel metódy a načrtnúť jej funkcionalitu. V tomto komentáre sa nepopisuje presná funkcionalita metódy. Komentár slúži pre zobrazenie informácií o metóde.

Príklad:

```
36          /**
37           * Method returns DrawingAlgorithm defined by parameter from stored array of DrawingAlgorithms
38           *
39           * @author Shade
40           * @param std::string name Name of the Drawing Algorithm to return
41           * @see DrawingAlgirithm
42           * @return DrawingAlgorithm defined by the name given in parameter
43           */
44          DrawingAlgorithm* getDrawingAlgorithm(std::string name);
```

Obr. 6.6 Príklad komentára metódy

Implementácia

V tomto prípade môžu byť použité dva druhy komentárov – blokový vysvetľovací a jednoriadkový. Slúžia na popis funkcionality metódy. Výber typu komentára je na zvážení autora. Jednoriadkový komentár sa používa pri jednoduchej funkcionalite, prípadne v prípade že pre vysvetlenie nie je potrebný dlhší opis. Blokový komentár slúži pre objasnenie zložitejšej alebo rozsiahlejšej časti kódu. Je potrebné komentovať tú funkcionalitu ktorá je ťažšia na pochopenie, komentár nemusí byť nutne pri všetkých príkazoch vnútri metódy. Tento komentár slúži ako pomôcka pri čítanú kódu, neslúži ako náhrada za štruktúru kódu a naming conventions.

Príklad:

```
89          //show layer
90          layers[i]->getGraphics()->getManualObject()->setVisible(true);
91
92          /*
93           * If class diagram features should be enabled display them,
94           * otherwise display sequence diagram features
95           */
96          if (Main::show_class_diagram_related){
97              //Class diagram - show classes
98              std::vector<Class*> classes = layers[i]->getClasses();
99              for(j = 0; j < classes.size(); j++) {
100                 classes[j]->getGraphics()->getManualObject()->setVisible(true);
101             }
102         } else{
103             //Sequence diagram - show lifelines
104             std::vector<Lifeline*> lifelines = layers[i]->getLifelines();
105             for(j = 0; j < lifelines.size(); j++) {
106                 lifelines[j]->getGraphics()->getManualObject()->setVisible(true);
107             }
```

Obr. 6.7 Príklad komentára vnútri metódy

7 Metodika #2 – Vytváranie unit testov pre jednotky kódu v jazyku C++

Andrej Valko

7.1 Úvod

Táto metodika slúži na definíciu postupu pri testovaní jednotiek zdrojového kódu vyvíjaného softvéru. Testovanie je zabezpečené pomocou Google C++ Testing frameworku v spojení s Google C++ Mocking frameworkom.

7.2 Vymedzenie obsahu (vybrané scenáre)

Unit testovanie nebude použité v kontexte test-driven development, ale v kontexte regresného testovania. Zámerom, nie je vytvoriť najprv testy, ktorých splnenie sa zabezpečí implementáciou. Zámerom je využiť unit testovania ako overenie, že po dodatočnej implementácií nebola predošlá implementácia nijako negatívne ovplyvnená.

7.3 Použité technológie

- Eclipse IDE for C/C++ Developers – (v tomto dokumente verzia: Kepler sr1)
- C/C++ Unit Testing Support – plugin pre Eclipse IDE (v tomto dokumente verzia: 7.2.0)
- Google C++ Testing Framework – framework pre unit testovanie C++ zdrojových kódov (v tomto dokumente verzia: 1.7.0)
- Google C++ Mocking Framework – framework pre simulovanie objektov (v tomto dokumente verzia: 1.7.0)

7.4 Použité pojmy

- jednotka zdrojového kódu – samostatne testovateľná časť softvéru, funkcionalita
- unit test – test jednotky zdrojového kódu, môže uplatňovať niekoľko test cases
- test case – súbor menších čiastkových testov, ktoré testujú jeden určitý prípad
- čiastkový test – test pre elementárnu časť funkcionality
- assertion – tvrdenie, ktorého vyhodnotenie ako nepravdivé znamená zlyhanie testu
- test fixture – nemenný stav systému v priebehu viacerých testov
- mock class – trieda simulovaného objektu

7.5 Dedikácia metodiky (účastníci riadiaci sa metodikou)

Metodika je určená pre každého programátora v tíme.

Programátor vystupuje v dvoch rolách:

- **tvorca unit testu** - Každý programátor okrem vytvorenia jednotky zdrojového kódu (funkcionality), vytvorí aj kód, ktorý zabezpečí test funkcionality, ktorú vytvoril.
- **tester** – Programátor po dokončení vlastnej jednotky zdrojového kódu, vykoná už existujúce unit testy pre jednotky kódu, ktoré pri práci ovplyvnil (tým overí, že sú aj naďalej funkčné)

Integrátor softvérového systému – Jeho úlohou je pripraviť zdrojové súbory projektu pre účely testovania.

7.6 Zoznam nadväzujúcich metodík

- Metodika ohlasovania implementačných chýb
- Metodika pre tvorbu zdrojového kódu

7.7 Metodika

Príprava

Príprava vývojového prostredia

Vystupujúce roly: : programátor v roly testera a tvorcu unit testu

Plugin poskytuje GUI pre jednoduchú prácu so spúšťaním a analýzou unit testov.

1. V Eclipse IDE je potrebné nainštalovať plugin „C/C++ Unit Testing Support“.

Príprava projektu

Vystupujúce roly: Integrátor softvérového systému

V osobitnej konfigurácií projektu je potrebné zahrnúť knižnice pre unit testovanie.

1. Je potrebné vytvoriť konfiguráciu (build target) s názvom „Test“.
2. Stiahnuť Google Test Framework z lokácie: http://googletest.googlecode.com/files/gtest-1.7.0.zip a Google C++ Mocking Framework z http://googlemock.googlecode.com/files/gmock-1.7.0.zip
3. Do projektu zahrnúť knižnice oboch frameworkov, ale len v konfigurácií „Test“.

4. Vytvoriť nový priečinok s názvom „tests“ na rovnakej úrovni ako priečinok „src“.
5. Priečinok „tests“ vylúčiť z konfigurácií iných ako „Test“.

Základný tok práce programátora na novej funkcionalite v súvislosti s testovaním

Vystupujúce roly: programátor v roly testera a tvorcu unit testu

1. Najprv sa spustia existujúce unit testy, ktoré overia funkčnosť doposiaľ implementovaných funkcionalít (Bližšie je tento proces rozobratý v kapitole Spustenie unit testov).
    a. Ak testy prebehnú úspešne, pokračuje sa krokom 2
    b. Ak niektorý z testov zlyhá, chyba sa ohlási (popísané v dokumente Metodika ohlasovania implementačných chýb).
2. Implementuje sa nová funkcionalita.
3. Opäť sa spustia existujúce unit testy.
    a. Ak testy prebehnú úspešne pokračuje sa krokom 4.
    b. Ak niektorý z testov zlyhá, je potrebné príčinu odstrániť.
        i. Ak je príčinou nedokonalá implementácia novej funkcionality, pokračuje sa krokom 2.
        ii. Ak je príčinou nekompatibilita novej funkcionality s už existujúcou, pričom nedokonalú implementáciu má už existujúca funkcionalita, toto sa ohlási ako chyba (popísané v dokumente Metodika ohlasovania implementačných chýb)
4. Vytvoria sa unit testy pre práve vyvinutú funkcionalitu. Je na programátorovi, aby zvoli vhodné testovacie scenáre (Bližšie je tento proces rozobratý v kapitole Tvorba unit testu) Tieto testy musia prebehnúť bez zlyhania.

Tvorba unit testu

Vystupujúce roly: tvorca unit testu

1. Každý unit test sa vytvorí v samostatnom súbore.
2. Všetky testy je potrebne umiestniť v súboroch lokalizovaných v priečinku „tests“ (tento bol vytvorený integrátorom softvérového systému pri nastavovaní projektu) .
3. Ak sú vytvorené triedy pre test fixtures, tieto sa tiež zahrnú do priečinku „tests“
4. Ak je potrebné vytvoriť simulované triedy (mock classes), tieto treba odvodiť od virtuálnych tried, ktoré sú v stromovej štruktúre priečinka „src“ (popísané v metodike pre Tvorbu zdrojového kódu), umiestniť ich však treba v priečinku „tests“.

5. Unit test pozostáva z viacerých testov, ktoré sa vytvoria makrom TEST(arg1, arg2) (prípadne TEST_F), pričom arg1 je názov pre test case, ktorý musí vystihovať testovanú funkcionalitu a musí končiť reťazcom "Test", arg2 je názov pre individuálny test, tento názov musí vystihovať časť testu funkcionality a nesmie končiť reťazcom "Test" (Ukážka 1).

```
TEST("LayerTest", MinimalLayerSize") { }
```

Ukážka 1

6. Pokiaľ negatívne vyhodnotenie assertion v teste negeneruje fatálne chyby, treba použiť expect variantu, tak aby bolo vyhodnotených čo najviac testov v jednom behu bez prerušenia v dôsledku fatálnej chyby.
7. Pri každom assertion je potrebné uviesť správu v anglickom jazyku vysvetľujúcu príčinu chyby (pri porovnaní vlastnosti dvoch objektov x a y je v správe potrebné uviesť okrem ich názvu aj ich typy - Ukážka 2)

```
ASSERT_EQ(x.size(), y.size()) << "Vectors x and y are of unequal length"
```

Ukážka 2

8. Vždy ak je možné, že pointer bude mať hodnotu NULL, treba vytvoriť najprv assertion na overenie, či daný pointer má hodnotu NULL.
9. Viaceré vstupné testovacie hodnoty treba aplikovať v rámci jedného čiastkového testu (Ukážka 3).

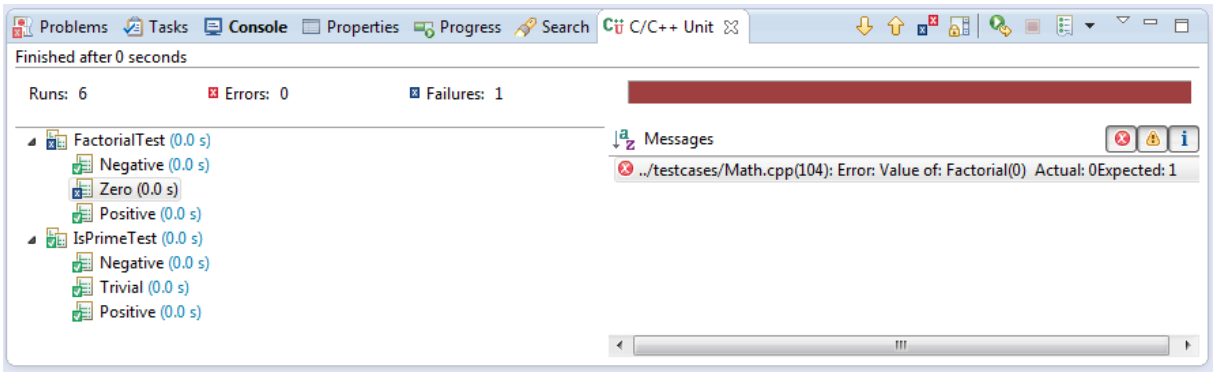
```
TEST(FactorialTest, Positive) {
 EXPECT_EQ(1, Factorial(1));
 EXPECT_EQ(2, Factorial(2));
 EXPECT_EQ(6, Factorial(3));
 EXPECT_EQ(40320, Factorial(8));
}
```

Ukážka 3

Spustenie unit testov

Vystupujúce roly: tester

1. Najprv treba skompilovať zdrojový kód pre konfiguráciu „Test“.
2. Potom ak ešte nebola vytvorená konfigurácia spustenia (Run Configuration) v Eclipse, tak:
    a. Je potrebné vytvoriť novú. Ako zdrojový spustiteľný súbor treba zvoliť <názov_aplikácie>.exe z priečinka “tests”.
    b. Ako Tests runner treba zvoliť Google Tests Runner.
3. Spustiť aplikáciu.
4. Skontrolovať výsledky (Príklad - Obr. 7.1)
5. Ak sa vo výsledkoch objaví chyba tak treba postupovať tak ako je uvedené v kapitole: Základný tok práce programátora na novej funkcionalite v súvislosti s testovaním (Predpokladom je, že test je vytvorený správne).



Obr. 7.1: Kontrola výsledkov testov

7.8 Zdroje

Google Test Framework pre vytváranie C++ testov:

http://googletest.googlecode.com/files/gtest-1.7.0.zip

Dokumentácia pre Google Test Framework:

http://code.google.com/p/googletest/w/list

Google C++ Mocking Framework:

http://googlemock.googlecode.com/files/gmock-1.7.0.zip

Dokumentácia pre Google C++ Mocking Framework:

http://code.google.com/p/googlemock/w/list

8 Metodika #3 – Manažment zadávania úloh

Gabriela Brndiarová

8.1 Úvod

Metodika sa venuje problematike vytvárania úloh a jej priraďovania konkrétnym osobám. Predpokladá sa existencia product backlog. Rozsah tejto metodiky končí v stave kedy je úloha zadaná v nástroji JIRA. Neobsahuje ani informácie o priradení úloh ku konkrétnym šprintom.

Zoznam nadväzujúcich metodík

Metodiky priamo nadväzujúce na postupy z tejto metodiky:

- Metodika vytvárania zápisnice
- Metodika vytvárania a spravovania product backlog
- Metodika spracovania pridelených úloh

Iné metodiky zmieňované v tejto metodike:

- Metodika komunikácie

Vymedzenie pojmov

Product backlog – List s používateľskými príbehmi pre celú aplikáciu. V tomto dokumente označovaný aj skrátene „backlog“.

JIRA – Nástroj na správu úloh od spoločnosti Atlassian.

Planning poker cards – Spôsob prideľovania zložitosti úlohám za pomoci kartičiek.

Používateľský príbeh – Funkcionalita, pre používateľa.

8.2 Roly

Vlastník produktu – Určuje si požiadavky na projekt (t.j. pedagogický vedúci).

Člen tímu – Ľubovoľný človek v tíme. Za člena tímu sa nepovažuje vlastník produktu. V rámci tohto projektu ide o študenta.

Zodpovedná osoba za úlohu – Primárne riešiteľ danej úlohy, ale v prípade potreby ju prerozdeľuje. V maximálnej možnej miere sa usiluje o to aby úloha bola vyhotovená načas a kvalitne.

Zadávateľ úlohy – Človek, ktorý zadáva do nástroja JIRA nové úlohy. Manažér plánovania, ak metodika neurčuje inak.

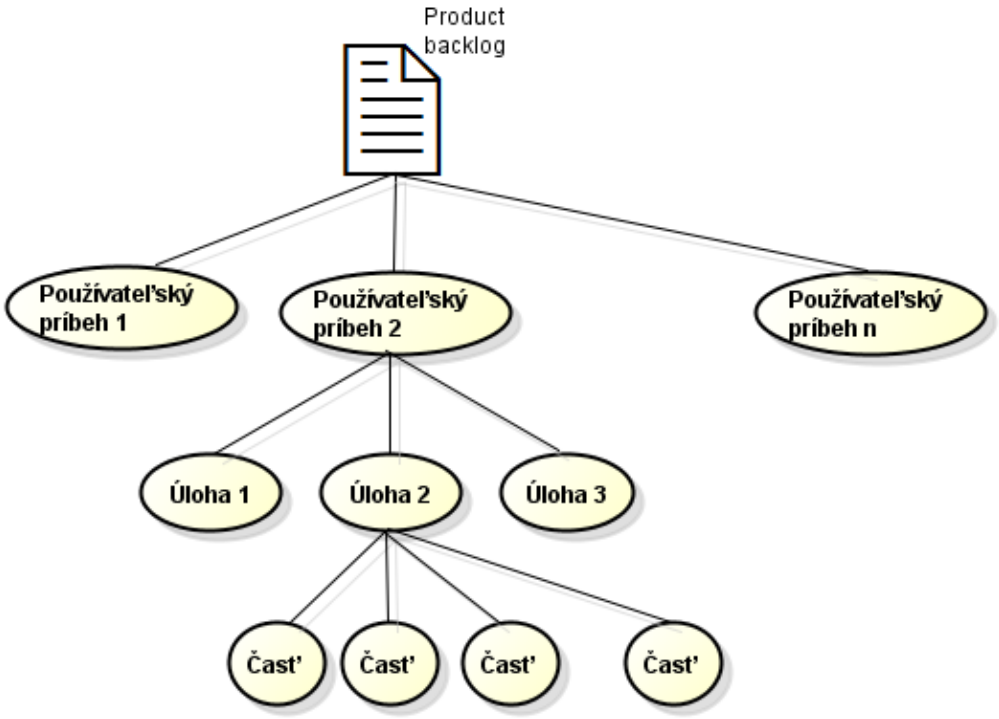
Vedúci tímu – Člen tímu, ktorý v tejto metodike koordinuje stretnutia.

8.3 Postupy

V tejto kapitole sú podrobne opísané jednotlivé procesy spadajúce pod zadávanie úloh. Nová úloha môže vzniknúť na začiatku šprintu (väčšina úloh), ale aj počas šprintu. Počas behu šprintu už nevznikajú úlohy, ktoré patria do iného používateľského príbehu než aké boli vybrané na začiatku šprintu. Sú to novoobjavené chyby a nevývojárske úlohy, ktoré je nutné urýchlene vyriešiť. Vzhľadom výrazne menší počet než úloh zadaných na začiatku šprintu, si pridelenie takejto úlohy nevyžaduje osobné stretnutie.

Definované sú 3 úrovne úloh (Obr. 8.1):

- **Používateľský príbeh z backlog**
    - označený prioritou 1-n (1 pre najväčšiu)
    - rozdelený na úlohy na vypracovanie (v počte 1-5)
- **Úloha na vypracovanie**
    - priradená zložitosť na základe analógie s veľkosťou tričiek (XS, S, M, L, XL)
    - priradená zodpovedná osoba
    - rozdelená na množstvo menších častí
    - dva typy: **vývojárska** (je časťou používateľského príbehu) a **nevývojárska** (nepatrí pod používateľský príbeh)
- **Časti úlohy**
    - nie sú formálne zapisované
    - určené príslušnou zodpovednou osobou
    - slúži predovšetkým na odhad stavu rozpracovanosti úlohy, ale aj na prípadné prerozdelenie v situácii, že zodpovedná osoba potrebuje pomôcť s úlohou



Obr. 8.1: Hierarchia úloh na projekte

Proces vytvárania a prideľovania úloh na začiatku šprintu

Vstupný stav:

- Existuje backlog s používateľskými príbehmi a ich prioritami.
- Príbehy z backlog sa nachádzajú v nástroji JIRA.
- Prebieha stretnutie členov tímu s vlastníkom produktu.
- Začína nový šprint.

Výstupný stav:

- Sú zadané úlohy pre nasledujúci šprint.
- Úlohy majú svoje zodpovedné osoby.
- O rozdelení úloh medzi členov tímu existuje fotografický záznam.

Technológie:

- **JIRA** – prezeranie a výber používateľských príbehov z backlog

- **Papieriky 10cm x 5cm** – fyzická reprezentácia úlohy
- **Kartičky so zložitosťami úloh** – určovanie zložitosti úloh technikou planning poker cards

Postup:

1. Vlastník produktu vyberie z backlog používateľské príbehy pre ďalší šprint. Členovia tímu môžu tento výber pripomienkovať.
2. Používateľské príbehy sa rozdelia na menšie úlohy na vypracovanie, ktoré sa napíšu na papieriky.
3. Jednotlivé úlohy sa na princípe planning poker cards ohodnotia zložitosťou. Tieto zložitosti sa pripíšu na príslušné papieriky.
4. Členovia tímu si zaradom berú po jednom papieriku s úlohou, ktorú majú záujem v ďalšom šprinte riešiť. Úlohy sa vyberajú v niekoľkých iteráciách až kým nie sú všetky úlohy priradené.
5. Porovná sa, či sa rovnajú súčty zložitostí úloh každého člena tímu. Ak to nie je možné dosiahnuť, tak aspoň s minimálnymi odchýlkami medzi jednotlivými členmi.
6. Každý člen tímu si napíše na svoj papierik meno, čím sa stáva zodpovednou osobou pre príslušnú úlohu.
7. Papieriky sa položia na stôl a odfotia. (Pre potreby spísania zápisnice – metodika písania zápisnice a zadania úloh do nástroja JIRA).
8. Do konca dňa stretnutia sú všetky úlohy zadané do nástroja JIRA (kapitola 0.0).

Proces vytvárania a prideľovania úloh počas šprintu

Vstupný stav:

- Šprint beží.
- Člen tímu si myslí, že je potrebné vytvoriť novú úlohu.

Výstupný stav:

- Úloha má svoju zodpovednú osobu a zložitosť.

Technológie:

Kanál pre skupinovú interaktívnu komunikáciu – definovaný v metodike komunikácie a slúži na komunikáciu členov tímu počas tohto procesu

Postup:

1. Člen tímu osloví ostatných členov tímu pomocou kanála pre skupinovú interaktívnu komunikáciu. Napíše čo a prečo by malo byť obsahom novej úlohy.
2. Na základe diskusie s ostatnými členmi sa rozhodne, či sa vytvorí nová úloha. Ak sa nová úloha zamietne proces končí, inak pokračuje v bode 3.
3. Určí sa zložitosť úlohy na základe dohody prítomných členov tímu.
4. Vyberá sa zodpovedná osoba. Pri výbere sa zohľadňujú odpovede na otázky:
    - Kto má záujem úlohu riešiť?
    - Kto mal na stretnutí na začiatku šprintu úlohy s nižšou zložitosťou?
    - S kým úloha súvisí? (Na základe manažérskej pozícii a riešenia podobných úloh v minulosti.)
5. Člen tímu, ktorý inicioval vytvorenie novej úlohy sa stáva zadávateľom. Nová úloha je neodkladne zadaná zadávateľom do nástroja JIRA.

Proces zadávania úlohy do nástroja JIRA

Vstupný stav:

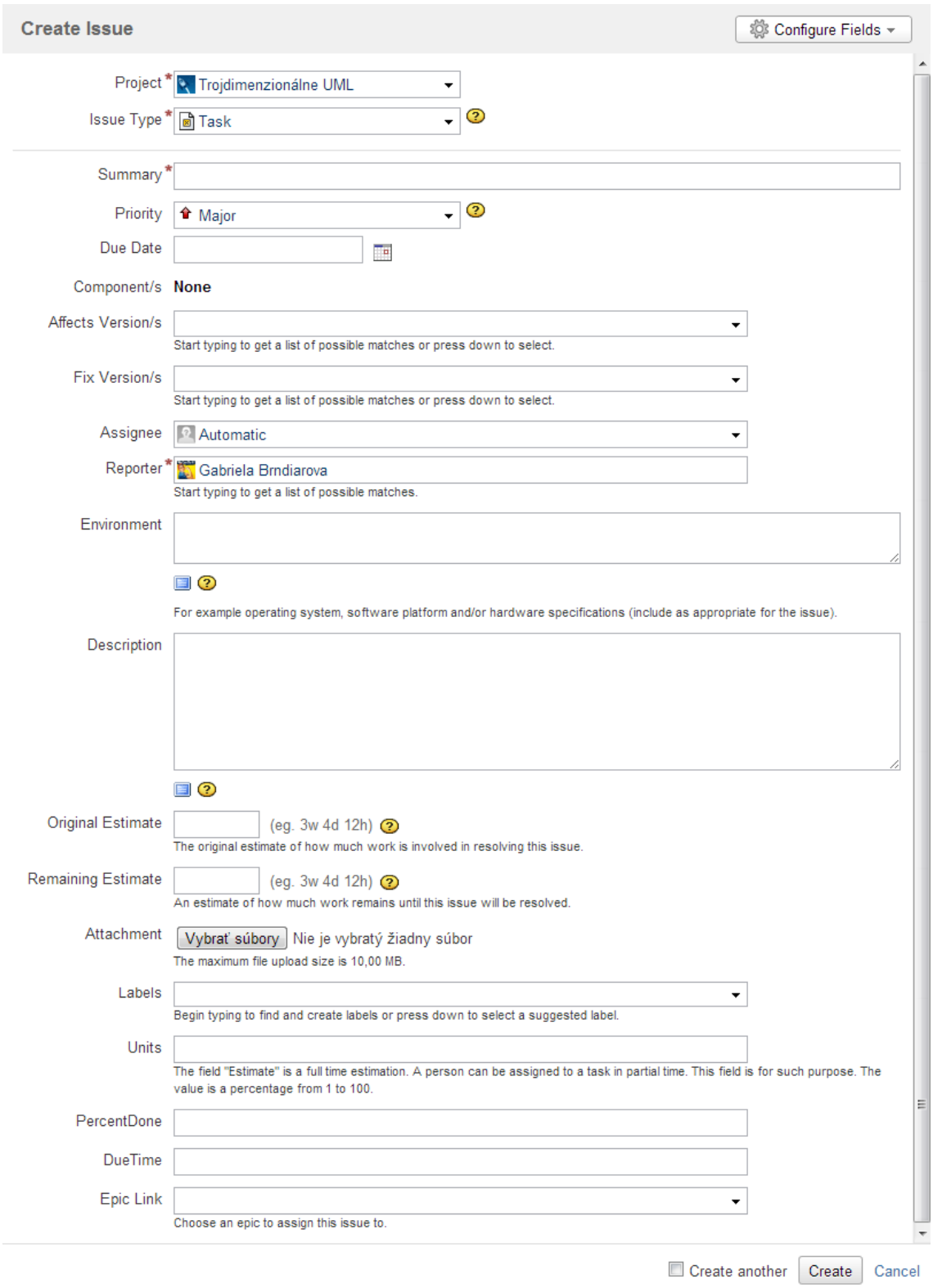
- Vznikla povinnosť pridať úlohu do nástroja JIRA.
- Úloha ma definovanú zodpovednú osobu a zložitosť.
- Zadávateľ je prihlásený v nástroji JIRA.

Výstupný stav:

- Úloha je zadaná v nástroji JIRA.

Technológie:

JIRA – pridanie novej úlohy



Obr. 8.2: Formulár na pridávanie novej úlohy v nástroji JIRA

Postup:

1. Zadávateľ klikne na tlačidlo „Create issue“ (v pravom hornom rohu pod svojim menom).
2. Zobrazí sa mu formulár na pridávanie novej úlohy (Obr. 8.2).
3. Zadávateľ vyplní nasledujúce polia (ostatné polia sa nevypĺňajú):

    - **Issue type**
        - „task“ – úloha
        - „bug“ – chyba v prototype
    - **Summary**
        - názov úlohy v slovenskom jazyku
        - použitie diakritiky
        - čo najjednoznačnejšie
        - 3-7 slov
    - **Priority**
        - „blocker“ – úloha blokuje splnenie inej úlohy
        - „critical“ – ukončenie je nutné do 5tich dní
        - „major“ – ukončenie je nutné do konca šprintu
        - „minor“ – ukončiť úlohu je možné aj po ukončení šprintu
        - „trivial“ – úloha nemá určený koniec a nevyplýva z požiadaviek vlastníka produktu, úloha je len aktivitou nad rámec povinností
    - **Due Date**
        - dátum, do kedy musí byť úloha hotová
        - dátum konca šprintu – úlohy, ktoré sa odovzdávajú na konci šprintu (aj blokujúce úlohy)
        - prázdne – úlohy, ktoré trvajú počas celého projektu alebo majú nejasný čas skončenia
        - iné - úlohy, ktoré treba oficiálne odovzdať do konkrétneho dátumu majú zadaný tento dátum

- **Assignee**
  - meno zodpovednej osoby
  - „unassigned“ - ak sú zodpovedný všetci
  - ak je zodpovedných viac ľudí, tak sa z nich vyberie 1 ľubovoľná osoba a ostatné sa pripíšu do „description“
- **Reporter**
  - meno zadávateľa úlohy
- **Description**
  - vyplnené len v prípade potreby
  - dodatočné informácie k úlohe, podrobný popis, užitočné odkazy

4. Zadávateľ stlačí tlačidlo „Create“.

9 Metodika #4 – Zdieľanie zdrojov pomocou Google Drive

Peter Kyseľ

9.1 Úvod

Účelom tejto metodiky je definícia postupu zdieľania dát v rámci tímového projektu pomocou nástroja Google Drive. Podrobne popisuje zdieľanie dát, formát, tvar a všetky potrebné informácie potrebné pre korektné zdieľanie. Na zdieľanie sa používa nástroj Google Drive.

Metodika je určená pre: každého člena tímu, ktorý chce zdieľať informácie s členmi tímu (upload zápisnice, zdieľanie screenshotov, písanie návodu na určitú oblasť a rôzne ďalšie formáty, ktorých zdieľanie bude prospešné pre členov tímu).

9.2 Pojmy a skratky

OS - operačný systém

SCRUM - Agilný rámec pre vývoj software, pre riadenie softvérových projektov a výrobkov alebo vývoj aplikácií.

2D - dvojdimenzionálny priestor

3D - trojrozmerný priestor

Google Drive - webová služba alebo aplikácia od spoločnosti google, ktorá umožňuje zdieľať súbory pomocou cloudu.

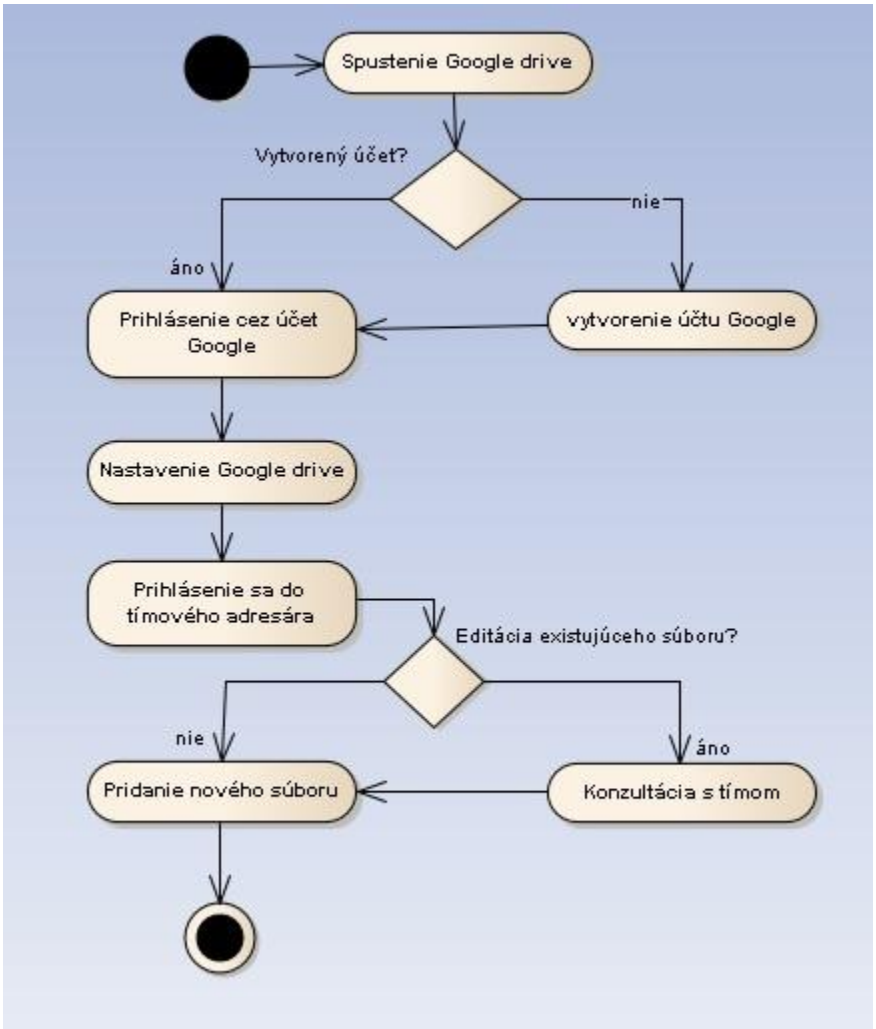
9.3 Podrobný popis krokov

Zdieľanie dát je procesom pozostávajúcim z viacerých krokov vypísaných v tabuľke:

Krok	Kapitola
Nastavenie používateľského konta	1.
Príprava nástroja Google Drive	2.

Organizácia a orientácia v priečinku Tímového projektu	3.
Pridanie nového súboru, priečinku	4.
Editácia súborov	5.

Postup vypísaný nižšie je pre lepšie pochopenie ilustrovaný aj activity diagramom (Obr. 9.1).



Obr. 9.1: Activity diagram metodiky

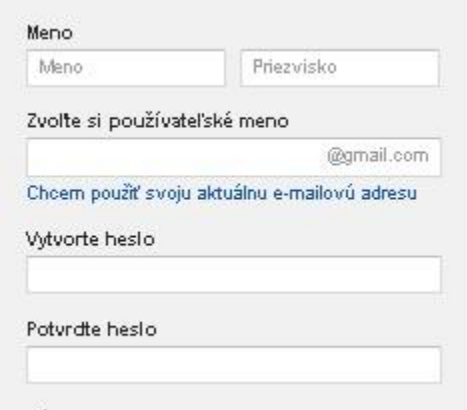
Nastavenie používateľského konta

Vstup: web stránka google.com

Výstup: účet na google a zdieľaný priečinok v rámci tímového projektu

Pokiaľ nemáte vytvorený účet na google tak účet si vytvoríte na web stránke www.google.com kde kliknete vpravo hore na položku prihlásiť sa a následne v strede dole na položku vytvoriť

účet. Vyplníte jednoduchý formulár pričom dôležitá položka pre vás bude používateľské meno tá bude potrebná pre nastavenie zdieľania priečinku. Účet google vám umožní prístup do všetkých služieb spoločnosti google.



Obr. 9.2: Registrácia položka Zvoľte si používateľské meno je kľúčovým prvkom pre nasledujúcu prácu

Pokiaľ účet máte pokračujete od tohto kroku.

Po registrácií kontaktujete Ivana Martoša, ktorému pošlete vaše používateľské meno zadané v predchádzajúcom kroku (napríklad pavol.mrkvicka @gmail.com) na základe ktorej vám do služby gmail.com (prihlasovacie údaje sú rovnaké do všetkých služieb google) bude poslaný e-mail s informáciou že priečinok TIMAK2013-14 je pre vás zdieľaný.

Príprava nástroja google drive

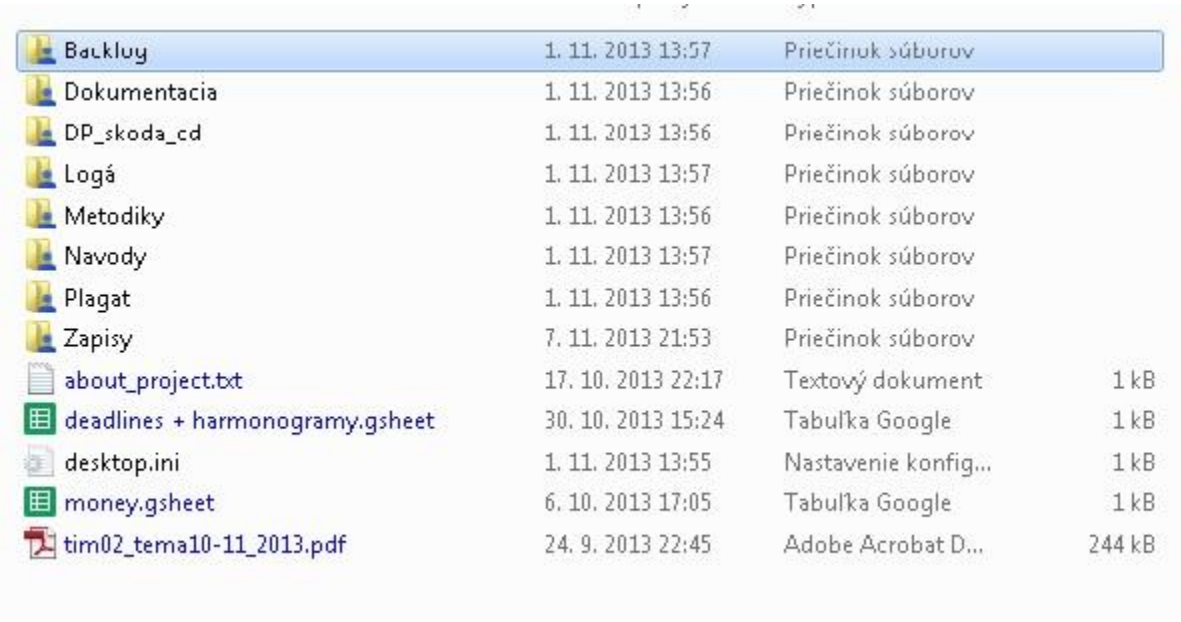
Vstup: web stránka drive.google.com

Výstup: korektne nainštalovaný nástroj Google Drive.

Pred pridávaním súborov v rámci tímového projektu navštívte web stránku drive.google.com ,ktorá bude vyžadovať pripojenie cez účet Google pokiaľ účet vytvorený v kroku 1. Následne po prihlásení v strede obrazovky vám prehliadač ponúkne možnosť stiahnutia aplikácie Google Drive do vášho počítača. Po stiahnutí nasleduje jednoduchá inštalácia, ktorá automaticky nainštaluje aplikáciu do vášho OS. Pri otvorení možností počítač (Tento počítač) vám pribudne k diskovým jednotkám aj Google Drive. V rámci tímového projektu vám bude sprístupnený priečinok TIMAK2013-14 od používateľa Ivan Martoš.

Organizácia a orientácia v priečinku Tímového projektu

Pri pridávaní nového obsahu je nutné zoznámiť sa s organizáciou priečinku (Obr. 9.3).



Obr. 9.3: Obsah Google Drive dňa 9.11.2013

Organizácia priečinku ku dňu 09.11.2013:

Backlog: naša virtuálna tabuľa zavedená vzhľadom na SCRUM.

Dokumentácia: všetky súbory týkajúce sa akejkoľvek dokumentácie.

DP_skoda_cd: prototyp diplomanta, na ktorého práci naša aplikácia pokračuje.

Logá: logá nášho tímu (2D, 3D).

Metodiky: akékoľvek metodiky v rámci nášho projektu.

Návody: návody využívané členmi v rámci tímu.

Plagát: plagát a fotky tímu.

Zápisy: zápisnice zo stretnutí.

Pridanie nového súboru, priečinku

V prípade že vás nový súbor alebo priečinok spadá pod kategóriu, ktorá sa už nachádza v aktuálnej štruktúre tak nový element bude uložený do podpriečinku ktorého sa prvok týka (návody, metodiky, screenshoty, backlog, dokumentácia, zápisy).

V opačnom prípade je nutné vytvoriť nový priečinok, ktorý bude čo najkonkrétnejšie reprezentovať kategóriu alebo skupinu súborov do nej vložených. Tento krok je nutné prediskutovať s členmi tímu až potom zrealizovať.

Názvy pridávaných súborov je nutné udržiavať v konzistentnej podobe v súlade už s existujúcim štandardom.

Zápisnice sa odovzdávajú v PDF formáte a ich názov je tvorený nasledovne Zapisnica_deň.mesiac.rok.pdf

Návody, metodiky: konkrétny stručný popis čoho sa návod alebo metodika týka.

Screenshoty : formát ukladania je názov riešeného problému prototypu, dátum a čas kedy bol screenshot vytvorený vo formáte JPG,GIF,PNG. uloha_11.11.2013_13-23.jpg

Editácia súborov

Ak je nutné editovať akýkoľvek súbor je nutné túto skutočnosť oznámiť členom tímu a danú zmenu s nimi prediskutovať. Po schválení návrhu o editáciu priečinka/značenia/súboru alebo akejkoľvek inej skutočnosti administrátor (osoba ktorá zdieľa obsah v tomto prípade Ivan Martoš) vykoná dohodnutú zmenu.

10 Metodika #5 – Spracovávanie pridelených úloh

Matej Štetiar

10.1 Úvod

Metodika spracovávanie pridelených úloh popisuje, ako správne pracovať s úlohami zadanými v systéme JIRA. Definuje činnosti, ktoré je potrebné vykonať počas práce na úlohe, tak aby bolo možné sledovať postup na tejto úlohe. Metodika objasňuje aj to, na koho je potrebné úlohu, po jej dokončení, preradiť na revíziu.

10.2 Role a zodpovednosti

Kapitola obsahuje popis rolí, ktorých sa táto metodika týka. Taktiež je naznačená zodpovednosť každej role v rámci tejto metodiky.

1. Riešiteľ úlohy	- Osoba zodpovedná za vyriešenie úlohy.
2. Kontrolór kvality	- Osoba zodpovedná zo kontrolu kvality a správnosti vyriešenia úlohy.

10.3 Použité pojmy

V tejto kapitole sú zozbierané cudzie pojmy, ktoré sa vyskytujú v texte.

- JIRA – systém na správu úloh
- Dashboard – informačná tabuľa. Jedná sa o skupinu filtrov systému JIRA zobrazených na jednej stránke a zdieľaných medzi používateľmi.
- Dashboards – množné číslo od Dashboard

10.4 Nadväzujúce metodiky

- Metodika pre manažment zadávania úloh - výsledkom tejto metodiky je úloha zadaná v systéme JIRA. Túto úlohu následne spracováva jej riešiteľ na základe tejto metodiky. Metodika pre manažment zadávanie úloh taktiež definuje rozdeľovanie úloh na podúlohy.
- Metodika pre revíziu vykonaných úloh – výsledkom tejto metodiky je rozhodnutie, či je úloha vykonaná v dostatočnej kvalite alebo je ju potrebné vrátiť na prerobenie.
- Metodika pre riešenie problémov s priradenou úlohou – má za úlohu popísať ako postupovať v prípade, že sa pri mojej úloh vyskytne problém.

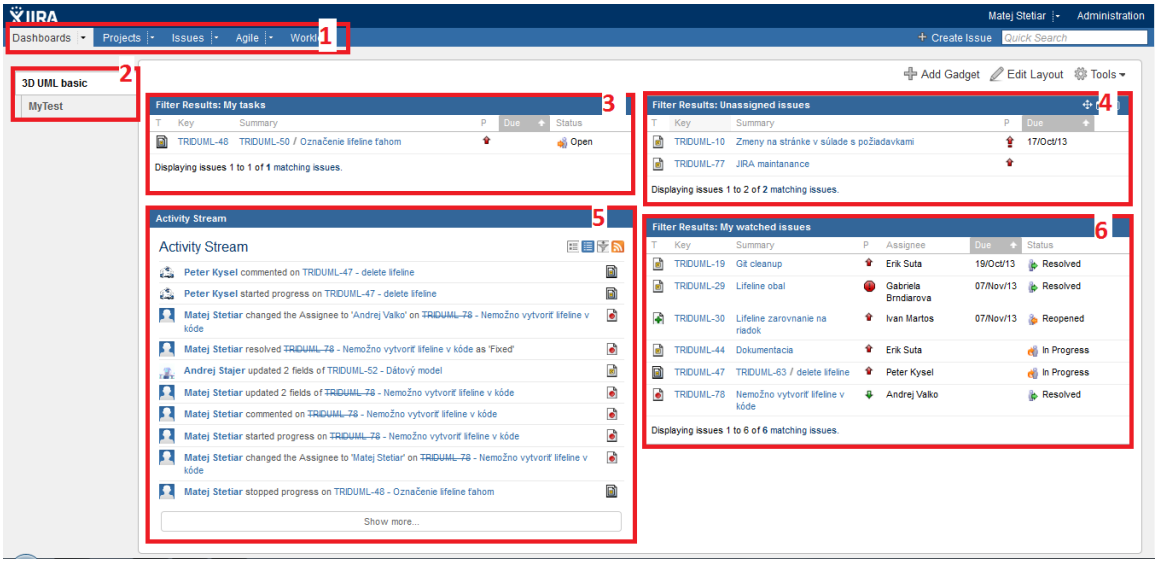
- Metodika pre manažment verziovania zdrojových súborov – má za úlohu objasniť ako pracovať pri verziovaní zdrojového kódu. Okrem iného objasňuje aj to, ako sa dostať k číslu commitu do git repozitára.

10.5 Popis prostredia

V tejto kapitole sa popisuje základný vzhľad nástroja JIRA, ktoré používame na manažment úloh. V tomto nástroji sme si vytvorili spoločný informačný panel, takzvaný „Dashboard" (viď. Obr. 10.1), ktorý používajú všetci členovia tímu.

Kostra záložky „Dashboards"

Základným znakom systému JIRA je prehľadnosť a intuitívne ovládanie. Toto sa autori snažia dosiahnuť záložkovým systémom, tabuľkami a grafmi. Preto sa tieto ovládacie prvky sa nachádzajú všade v systéme.



Obr. 10.1: Informačný panel 3D UML basic

1. **Základné záložky** (Obr. 10.1 časť 1) sú zobrazené v riadku tesne pod logom a prihlasovacím menom. Tieto umožňujú prepínať medzi funkcionalitami v systéme.
2. Druhý typ záložiek sú **záložky informačných panelov** (Obr. 10.1 časť 2). Tieto umožňujú prepínanie sa medzi jednotlivými panelmi. Nachádzajú sa na ľavej strane obrazovky a sú pomenované názvami informačných panelov. Zobrazujú sa tu panely, ktoré má prihlásený používateľ označené ako obľúbené.

Informačný panel „3D UML basic"

Tento informačný panel má uľahčiť prácu so systémom JIRA používateľom, ktorí s ním ešte nepracovali. Mal by obsahovať všetky údaje potrebné pre spracovávanie úloh členmi tímu. Je rozdelený na 4 základné časti.

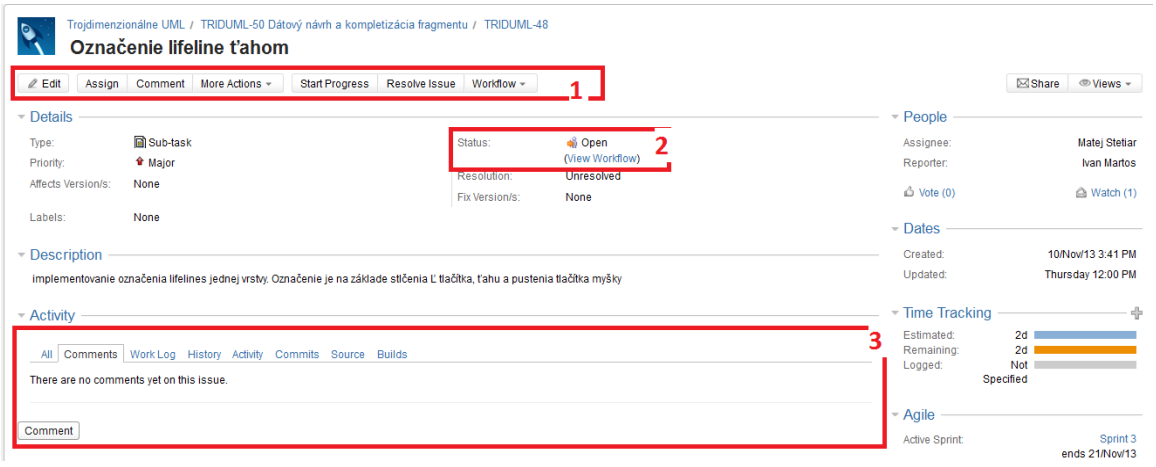
1. **Filter „Moje úlohy“** (Obr. 10.1 časť 3) – obsahuje všetky úlohy, ktoré sú na mňa priradené a sú aktívne. To, že sú aktívne znamená, že ešte neboli uzatvorené a teda je potrebné na ich vykonať nejakú aktivitu.

Filter „Úlohy bez riešiteľa“ (

2. Obr. 10.1: Informačný panel 3D UML basic časť 4) – obsahuje všetky úlohy, ktoré nemajú prideleného riešiteľa a sú aktívne. Tieto úlohy si môže používateľ priradiť na seba a začať ich riešiť, alebo budú priradené riešiteľovi manažérom zodpovedným za oblasť, ktorej sa úloha týka.
3. **Filter „Úlohy, ktoré sledujem“** (Obr. 10.1 časť 6) – obsahuje všetky úlohy, v ktorých prihlásený používateľ niekedy vystupoval. To znamená, že ju vytváral, komentoval, riešil alebo sám nastavil, že ma táto úloha zaujíma. Taktiež je v tomto filtri možné vidieť úlohy, v ktorých sa bolo meno prihláseného používateľa citované v komentároch alebo v popise úlohy.
4. **Záznam aktivít** (Obr. 10.1 časť 5) – v tejto tabuľke je možné vidieť poslednú aktivitu všetkých členov tímu v tomto systéme.

Detail úlohy

Detail úlohy je obrazovka, na ktorej za zobrazuje podrobnejší popis úlohy, jej prílohy, komentáre a rôzne iné podrobnosti. Väčšina aktivít spojených s postupom v rámci úlohy sa vykonáva a zobrazuje práve tu.



Obr. 10.2: Detail úlohy

1. **Sekcia s riadiacimi elementmi** (Obr. 10.2 časť 1) – je dynamická sekcia, ktorá sa mení vzhľadom na to v ako je úloha stave. V tejto sekcii je možné meniť stavy úlohy, komentovať úlohy, zaznačovať čas strávený riešením úlohy, preradzovať úlohy na iného riešiteľa a mnoho ďalších vecí.
2. **Sekcia so stavom úlohy** (Obr. 10.2 časť 2) – sekcia zobrazuje v akom stave s úloha práve nachádza. Po kliknutí na odkaz „View Workflow“ je sa zobrazí stavový diagram, ktorý ukazuje možné prechody medzi stavmi.
3. **Sekcia s komentármi** (Obr. 10.2 časť 3) – v tejto sekcia sa zobrazujú komentáre používateľov k tejto úlohe. Taktiež je tu možné pridať komentár a to pomocou tlačidla „Comment“ (komentovať).

Na tejto obrazovke je zobrazených ešte omnoho viac informácií, ale tie ne sú relevantné vzhľadom k tejto metodike.

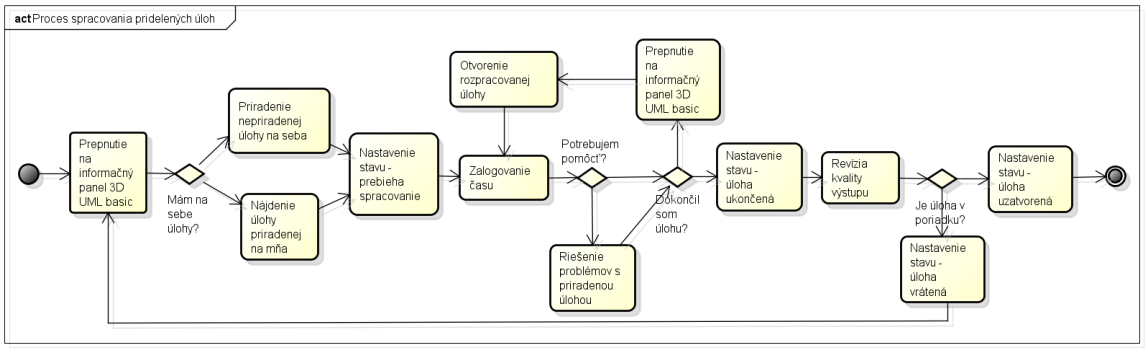
10.6 Proces spracovávania pridelených úloh

Proces spracovávania pridelených úloh môže začínať len v prípade, že sú splnené vstupné požiadavky. Taktiež sa začiatok procesu môže líšiť podľa toho, či ide prihlásený používateľ riešiť úlohu, ktorá je priradená priamo na neho alebo začína s nepriradenou úlohou.

Samotný proces je možné vidieť nižšie (Obr. 10.3). Jednotlivé aktivity procesu sú podrobne popísané v nasledujúcej kapitole. Proces začína prezeraním si úloh v JIRE riešiteľom úlohy. To kedy si úlohy prezerá nie je predmetom tejto metodiky.

Stručný popis procesu

1. Riešiteľ úlohy sa rozhodol, že ide vyriešiť nejakú úlohu.
2. Pozrie sa do systému JIRA.
3. Ak má úlohy priradené na seba vezme si jednu z nich.
4. V prípade, že riešiteľ na sebe nemá žiadne úlohy, vezme si jednu z nepriradených úloh.
5. Úlohe nastaví stav prebieha spracovanie a začne pracovať na úlohe.
6. Po dokončení práce na úlohe zaloguje čas. Úloha ešte nemusí byť hotová, je potrebné aby zaznačil svoj pokrok.
7. V prípade, že riešiteľ potrebuje pri riešení úlohy pomôcť, kontaktuje ostatných. Toto však nie je predmetom tejto metodiky.
8. Ak na úlohe začne opätovne pracovať, lebo ju ešte nedokončil, opätovne si ju otvorí a pokračuje krokom 6.
9. Po dokončení úlohy ju nastaví ako dokončenú a posunie ju na revíziu kvality výstupu.
10. Pri revízii kvality výstupu sa kontrolór kvality rozhodne či je úloha v poriadku alebo nie.
11. V prípade, m že je úloha v poriadku, uzatvorí ju. Ak je v nej niečo nevyhovujúce vráti úlohu riešiteľovi a proces začína od bodu 1.



Obr. 10.3: Proces spracovania pridelených úloh

10.7 Popis úloh vyskytujúcich sa v procese

Proces spracovávanie úloh je založený na viacerých rozhodovacích bodoch a je možné prechádzať viacerými úlohami viac ako raz. Jednotlivé úlohy a ich rozhodovacie body sa preto nachádzajú tu pre zlepšenie prehľadnosti dokumentu.

Vstupné požiadavky:

1. Osoby v oboch roliach sú prihlásené na stránke jira.fiit.stuba.sk (nemusia byť súčasne).
2. Obe osoby majú pridaný informačný panel „3D UML basic“ medzi svojimi obľúbenými.

Úlohy v procese

Podrobný popis jednotlivých úloh v procese.

Prepnutie sa na informačný panel „3D UML basic“

1. Používateľ sa prihlási do systému JIRA na adrese jira.fiit.stuba.sk.
2. Používateľ systému klikne na záložku „Dashboards“ medzi základnými záložkami (Obr. 10.1 časť 1).
3. V prípade, že používateľ má viac ako jeden informačný panel, klikne na záložku informačného panelu „3D UML basic“ (Obr. 10.1 časť 2).

Priradenie nepriradenej úlohy na seba

1. Vo filtri úlohy bez riešiteľa (Obr. 10.1 časť 4) si riešiteľ vyberie vhodnú úlohu. Pod vhodnou úlohou sa rozumie úloha, ktorú si riešiteľ myslí, že vie vyriešiť. Pri výbere úlohy postupuje zhora nadol, pričom sa snaží vybrať si úlohu čo najvyššie.
2. Po výbere úlohy otvorí detail úlohy kliknutím na jej kľúč (stĺpec Key) alebo popis (stĺpec Summary).
3. V sekcii s riadiacimi elementmi (Obr. 10.2 časť 1) riešiteľ klikne na tlačidlo s nápisom „Assign To Me“ (priraď na mňa).

Nájdenie úlohy priradenej na mňa

1. Vo filtri moje úlohy si riešiteľ vyberie prvú úlohu, ktorá je v stave „Open“ alebo „Reopened“, kliknutím na jej kľúč alebo popis. Úlohy sú totiž zoradené podľa priority a času dokončenia.
2. Riešiteľovi sa zobrazí detail úlohy.

Nastavenie stavu – prebieha spracovanie

1. Riešiteľ sa oboznámi s detailmi úlohy a poprípade s komentármi, ak sa pri úlohe nejaké nachádzajú.
2. Klikne v sekcii s riadiacimi elementmi (Obr. 10.2 časť 1) na tlačidlo „Start Progress“ (začni prácu).
3. Začne pracovať na úlohe.

Zalogovanie času

Používateľ potrebuje zalogovať čas, ktorý strávil riešením úlohy. Úloha nemusí byť hotová, ale dnes sa jej už nebude venovať. Takýmto spôsobom dá ostatným najavo, že sa na úlohe niečo deje. Používateľ sa pri tejto aktivite nachádza na detaile úlohy, na ktorú si chce zalogovať čas.

1. Používateľ v sekcii s riadiacimi elementmi (Obr. 10.2 časť 1) klikne na tlačidlo „More Actions“ (viac možností).
2. Zobrazí sa mu zoznam možností, ktoré môže v systéme vykonať. Vyberie si možnosť na 4. mieste „Log Work“ (vykáž čas).
3. Zobrazí sa mu malé okienko, kde vyplní polia „Time Spent“ (strávený čas) a „Work Description“ (popis práce).

Políčko pre strávený čas vyplní takto:

- Celým číslom napíše počet strávených hodín a za číslo napíše znak „h“.
- Napíše počet strávených minúť zaokrúhlených matematicky na štvrťhodiny (t.j. 15, 30, 45) a napíše znak „m“. V prípade, že má 0 minút minúty nepíše.

Políčko pre popis práce vyplní stručným popisom vykonanej práce.

4. Používateľ klikne na tlačidlo „Log“ (vykáž).

Otvorenie rozpracovanej úlohy

V prípade, že používateľ potrebuje otvoriť úlohu, na ktorej už pracoval, postupuje rovnako ako pri úlohe „Nájdenie úlohy priradenej na mňa“. V tomto prípade však neotvorí prvú úlohu, ale úlohu so stavom „In progress“ (prebieha spracovanie).

Nastavenie stavu – úloha ukončená

1. V prípade, že riešiteľ považuje svoju úlohu za ukončenú, v sekcii s komentármi (Obr. 10.2 časť 3) klikne na tlačidlo „Comment“ (komentuj).

2. Otvorí sa mu editovateľná sekcia, do ktorej vpíše všetko čo na danej úlohe vykonal.
3. Po vypísaní všetkých podrobností klikne na tlačidlo „Add“ (pridaj).
4. V sekcii s riadiacimi prvkami (Obr. 10.2 časť 1) klikne na tlačidlo „Resolve Issue“ (vyrieš úlohu).
5. Zobrazí sa modálne okno s formulárom. Riešiteľ vyplní polia „Assignee“ (priradený používateľ) a „Comment“ (komentár).
    a. Do políčka pre priradeného používateľa začne písať meno osoby, ktorá reviduje jeho kód. Zoznam osôb je uvedený nižšie.
    b. Systém JIRA mu ponúkne používateľov na základe časti mena, ktorú zadal.
    c. Riešiteľ vyberie osobu kliknutím na jej meno.
    d. V prípade, že išlo o programátorskú úlohu do sekcie pre komentár vpíše číslo commitu do git repozitára (bližšie informácie poskytuje metodika Manažment verziovania zdrojových súborov).
6. Riešiteľ klikne na tlačidlo „Resolve“ (vyrieš).

Riešiteľ úlohy	**Kontrolór kvality**
Gabriela Brnadiarová	Andrej Štajer
Peter Kyseľ	Matej Štetiar
Ivan Martoš	Erik Šuta
Andrej Štajer	Ivan Martoš
Matej Štetiar	Gabriela Brndiarová
Erik Šuta	Andrej Valko
Andrej Valko	Peter Kyseľ

Revízia kvality výstupu

1. Kontrolór kvality vykoná úlohu prepnutie sa na informačný panel 3D UML basic.
2. Vo filtri moje úlohy nájde úlohu v stave „Resolved“ (vyriešená).
3. Zobrazí si detail úlohy kliknutím na kľúč alebo popis úlohy.
4. Vykoná revíziu úlohy na základe metodiky pre revíziu úlohy.
5. Zaloguje si čas na základe činnosti zalogovanie času.

Nastavenie stavu – úloha uzatvorená

1. Kontrolór kvality v sekcii s riadiacimi elementmi klikne na tlačidlo „Close Issue“ (zatvor úlohu).

2. Do políčka Assignee začne písať meno osoby, ktorá úlohu riešila. Toto meno nájde na základe tabuľky 1.
3. Systém JIRA mu zobrazí mena na základe časti meno, ktoré zadal.
4. Vyberie meno riešiteľa kliknutím na myš.
5. Do sekcie „Comment“ (komentár) napíše krátke zhodnotenie.
6. Klikne na tlačidlo „Close Issue“ (zatvor úlohu).

Nastavenie stavu – úloha vrátená

1. Používateľ postupuje rovnako ako v aktivite nastavenie stavu – úloha zatvorená, s tým rozdielom, že kliká na „Reopen Issue“ (otvor úlohu) namiesto „Close Issue“ (zatvor úlohu).

Riešenie problémov s priradenou úlohou

V prípade, že riešiteľ potrebuje s úlohou, ktorú rieši pomôcť, postupuje podľa metodiky pre riešenie problémov s úlohou.

Rozhodovacie body v procese

Mám na sebe úlohy?

V tomto rozhodovacom bode sa riešiteľ úlohy rozhoduje, či má na sebe priradené úlohy. To zistí pri pohľade do filtra moje úlohy. V prípade, že na sebe nemá žiadne úlohy, filter je prázdny, môže si zobrať úlohu z nepriradených úloh.

Potrebujem pomôcť?

V prípade, že riešim úlohu a neviem sa ďalej pohnúť alebo som zistil, že nestíham pokračujem na aktivitu Riešenie problémov s priradenou úlohou.

Dokončil som úlohu?

Ak si myslím, že úloha, ktorú som riešil je hotová a vypracovaná v dostatočnej kvalite, môžem úlohu považovať za vyriešenú. Ak ide o programátorskú úlohu zdrojové kódy musia byť v zdieľanom repozitári a popis úlohy v dokumentácií. V tom prípade môžem pokračovať úlohou Nastavenie stavu – úloha ukončená.

Je úloha v poriadku?

Toto rozhodnutie robí kontrolór kvality na základe výstupov z metodiky pre kontrolu kvality.

11 Metodika #6 – Zapracovávanie zmien do modelu s použitím nástroja Enterprise Architect

Andrej Štajer

11.1 Úvod

Metodika bola napísaná pre zjednotenie postupov a princípov pri zapracovávaní zmien do dátového modelu. Ako podporný CASE nástoj bol vybraný Enterprise Architect. Role pre, ktoré je metodika určená sú role návrhára a architekta.

Metodika rozširuje existujúcu metodiku Manažment verziovania zdrojových súborov a vychádza z predpokladu zadania úlohy na zmenu modelu alebo návrhu novej funkcionality.

11.2 Role a zodpovednosti

Architekt(návrhár)	- pridávanie, úprava a odstraňovanie artefaktov modelu - zachovanie názvoslovnej konvencie - aktuálnosť modelu

11.3 Zoznam nadväzujúcich metodík

- Manažment verziovania zdrojových súborov

11.4 Použité pojmy a skratky

Enterprise Architect (EA) – CASE nástroj, so širokou podporou pre analýzu, návrh a implementáciu softvéru. Pre podporu „Code Engineering“ je potrebná minimálna verzia Professional vo verzií 9.0

Code engineering – schopnosť nástroja generovať zdrojový kód na základe návrhu modelu a vytvorenie modelu na základe zdrojového kódu.

Unified Modeling Language (UML) – modelovací jazyk vizualizujúci požiadavky na softvér v rôznych pohľadoch pomocou rôznych diagramov

Workspace – pracovný priečinok, obsahuje všetky zdrojové kódy

Repozitár – priestor určený na verziovanie dokumentov a zdrojových kódov, jeho súčasťou je aj workspace

11.5 Proces zapracovania zmeny

	Krok	**kapitola**
1	Otvorenie nástroja	11.6
2	Aktualizovanie lokálneho repozitára	11.6
3	Otvorenie projektu	11.6
4	Otvorenie dátového modelu	11.6
5	Synchronizácia so zdrojovým kódom	11.6
6	Vykonanie zmien modelu	11.6
7	Kontrola názvoslovia	11.6
8	Vygenerovanie zdrojového kódu	11.6
9	Aktualizovanie spoločného repozitára	11.6

11.6 Podrobné popisy

Otvorenie nástroja

- Spustenie nástroja EA.
- Nástroj automaticky otvorí formulár Open Enterprise Architect Project

Aktualizovanie lokálneho repozitára

- Aktualizovať lokálny repozitár pomocou metodiky Manažment verziovania zdrojových súborov.

Otvorenie projektu

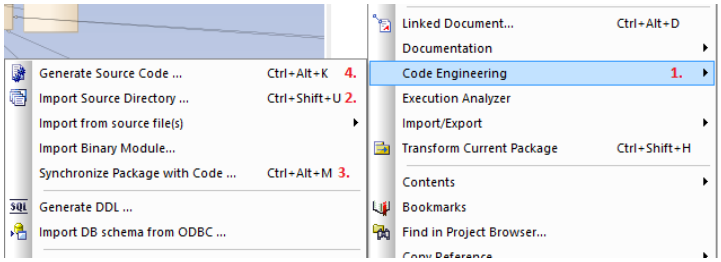
- Otvorenie projektu s modelom nazvaného „TimakModel.eap“. Existujú dve možnosti ako to vykonať
  a. Výber projektu z ponúkaných možností

b. Ak sa projekt nenachádza v ponuke je nutné kliknúť na tlačidlo „Browse for projects“ a vyhľadať cestu do repozitára projektu. EA projekt sa nachádza v koreňovom adresári.

Otvorenie dátového modelu

- V časti project browser rozkliknúť model s názvom „DataModel“.
- Rozbaliť balík pomenovaný „Classes“.
- Dvojklikom otvoriť diagram tried „Class model“

Synchronizácia so zdrojovým kódom



Obr. 11.1: Práca s „Code engineering”

- Otvoriť kontextové menu balíka „Classes“
- Vybrať rozšírené menu „Code Engineering“ viď Obr. 11.1a červené číslo 1
- Vybrať možnosť „Import Source Directory“ viď Obr. 11.1 a červené číslo 2
    - vo formulári nastaviť „Root Directory“ cestu k workspace v lokálnom repozitári ([repozitár]\workspace\timak)
    - „source type“ nastaviť C++
    - potvrdiť tlačidlom „OK“
- Nástroj spustí proces reverzného generovania tried modelu na konci ktorého, sa zobrazí formulár neexistujúcich tried v zdrojovom kóde.
    - Triedu je možné odstrániť z modelu. Vtedy keď bola trieda odstránená na úrovni zdrojového kódu, kvôli funkčnej oprave a existujem k tomu potrebná dokumentácia. Druhá možnosť je v prípade požiadavky na odstránenie tejto triedy z modelu.
        - Odstránenie je vykonané tak, že je konkrétna trieda označená
        - Následne je kliknuté na tlačidlo „Delete“ v časti „Set action“
    - V ostatných prípadoch je potrebné triedu ponechať v modeli
        - Ponechanie je vykonané, tak že je konkrétna trieda označená
        - Následne je kliknuté na tlačidlo „Ignore“ v časti „Set action“
- Potvrdiť tlačidlom „OK“ a následne po úspešnom importe stlačiť tlačidlo „Close“

Vykonanie zmien modelu

- V závislosti zadania úlohy môžu nastať tri typy zmien pre elementy modelu:

- Vytvorenie
  - V časti „Toolbox“ je potrebné vybrať typ elementu a vložiť ho na požadovanú pozíciu v modeli
  - Nástroj otvorí formulár elementu, tu je zadaný názov elementu podľa kapitoly 0.0
  - Položku „Language“ je potrebné zmeniť na „C++“
  - Do poľa „Notes“ vložiť dokumentačné poznámky na vysvetlenie vlastností a správania objektu
  - Vytvorenie potvrdiť tlačidlom „OK“
- Úprava
  - Ak sú upravované vlastnosti elementu
    - dvojklikom na element sa otvorí formulár elementu.
    - Pokračovať ako pri vytváraní elementu
  - Ak je upravovaná pozícia v modeli element je označený a presunutý na požadované miesto
- Odstránenie
  - Otvoriť kontextové menu elementu
  - Vybrať možnosť „Delete“
    - POZOR tlačidlom „delete“ sa element odstráni iba z diagramu

Kontrola názvoslovia

- Pri úpravách v modeli je potrebné dodržiavať stanovené pravidlá názvoslovia tak ako to zobrazuje Tabuľka 11.1.
- Všetky pomenovania sú v anglickom jazyku.

Oblasť	**Názvoslovné pravidlá**

Trieda	- Základ podstatné meno s veľkým začiatočným písmenom
	- Pri viacslovnom pomenovaní,
	  o bez medzier
	  o nové slovo veľkým začiatočným písmenom

Rozhranie	- Na začiatku pomenovania je veľké „I“ spojené s názvom vytvoreným rovnako ako pre triedu
Metóda	- Sloveso s malým začiatočným písmenom s možným doplňujúcim slovom alebo slovami - Pri viacslovných pomenovaniach každé nové slovo s veľkým začiatočným písmenom

Tabuľka 11.1

Vygenerovanie zdrojového kódu

Tento krok je relevantný iba v prípade vytvorenia nových elementov modelu

- Po ukončení úprav modelu Otvoriť kontextové menu balíka „Classes“

- Vybrať rozšírené menu „Code Engineering“ viď Obr. 11.1 a červené číslo 1
- Vybrať možnosť „Generate Source Code“ viď Obr. 11.1 a červené číslo 4
- Otvorí sa formulár generovania zdrojového kódu
- Označia sa vytvorené objekty
- Voľba sa potvrdí tlačidlom „Generate“
- Vybrať cestu uloženia postupne pre jednotlivé vytvorené objekty
- Po ukončení formulár vypíše úspešnosť operácie
- Následne zavrieť formulár generovania kódu

Aktualizovanie spoločného repozitára

- Aktualizovať spoločný serverový repozitár o vykonané zmeny pomocou metodiky Manažment verziovania zdrojových súborov.

12 Metodika #7 – Manažment verziovania zdrojových súborov

Erik Šuta

12.1 Úvod do manažmentu verziovania

V tomto dokumente je prezentovaná metodika, ktorej cieľom je podať ucelený opis vybraných operácií súvisiacich s verziovaním zdrojových súborov. Metodika sa sústreďuje predovšetkým na operácie súvisiace s implementáciou novej funkcionality do existujúceho produktu, resp. opravou chýb vzniknutých počas predošlej implementácie v kontexte verziovania zdrojových súborov. Je určená pre všetkých členov tímu, ktorí sa podieľajú na implementácii v projekte.

Koncept metodiky je stavaný tak, aby ju bolo možné využívať predovšetkým v predmete Tímový projekt. Metodika je upravená s ohľadom na agilné metódy vývoja softvéru. Pri opisoch postupov je ako nástroj zvolený nástroj Git GUI, ktorý je súčasťou štandardnej inštalácie nástroja Git. Metodika ponúka aj alternatívu vo forme príkazov, ktoré je potrebné vykonať na príkazovom riadku, pričom táto forma vykonania je preferovaná.

Metodika nepriamo súvisí s nasledujúcimi metodikami:

- Martoš I. – Dokumentácia v zdrojových kódoch
- Štajer A. – Zapracovanie zmien do modelu
- Štetiar M. – Spracovávanie pridelených úloh – manažment úloh v systéme JIRA
- Valko A. – Metodika vytvárania unit testov

Dôležité pojmy

Git – nástroj používaný na verziovanie zdrojových súborov.

Vetva vývoja (branch) – samostatná organizačná jednotka vývoja skladajúca sa z jednej alebo viacerých revízií obsahujúcich zmeny v zdrojových kódoch, ktoré spolu obsahovo súvisia.

Zlúčenie vetiev (merge) – spojenie dvoch vývojových vetiev do jednej.

Hlavná vetva vývoja (development branch) – vetva, na ktorej prebieha všeobecný vývoj v projekte.

Koreňová vetva (master/origin) – vývojová vetva, ktorá sa neustále nachádza v stave pripravenom na nasadenie do produkcie.

Role a zodpovednosti

V tejto časti metodiky uvádzame zoznam procesov v kontexte verziovania zdrojových súborov a role členov tímu prislúchajúce k jednotlivým procesom. Pre úplnosť uvádzame aj procesy, ktoré nie sú opísané v tejto metodike, avšak v kontexte verziovania zdrojových súborov majú význam.

Proces	**Roly v procese**
1. Vytvorenie hlavnej vývojovej vetvy *(kap. 2.1)*	Manažér podpory vývoja: - Zodpovedný za vytvorenie, kontrola správnosti
2. Vytvorenie novej funkcionálnej vetvy vývoja *(kap. 2.2)*	Manažér podpory vývoja: - Dohliadnutie na správnosť vykonaného úkonu Programátor: - Vykonanie vytvorenia vývojovej vetvy
3. Vytvorenie vetvy vývoja za účelom opravy chýb *(kap. 2.3.)*	Manažér podpory vývoja: - Dohliadnutie na správnosť vykonaného úkonu Programátor: - Vykonanie vytvorenia vývojovej vetvy
4. Zlúčenie vývojových vetiev *(kap. 2.4)*	Manažér podpory vývoja: - Dohliadnutie na správnosť vykonaného úkonu Programátor: - Vykonanie vytvorenia vývojovej vetvy Manažér kvality: - Kontrola revízií a zdrojových súborov, resp. ich zmien v danej vetve vývoja ešte pred zlúčením

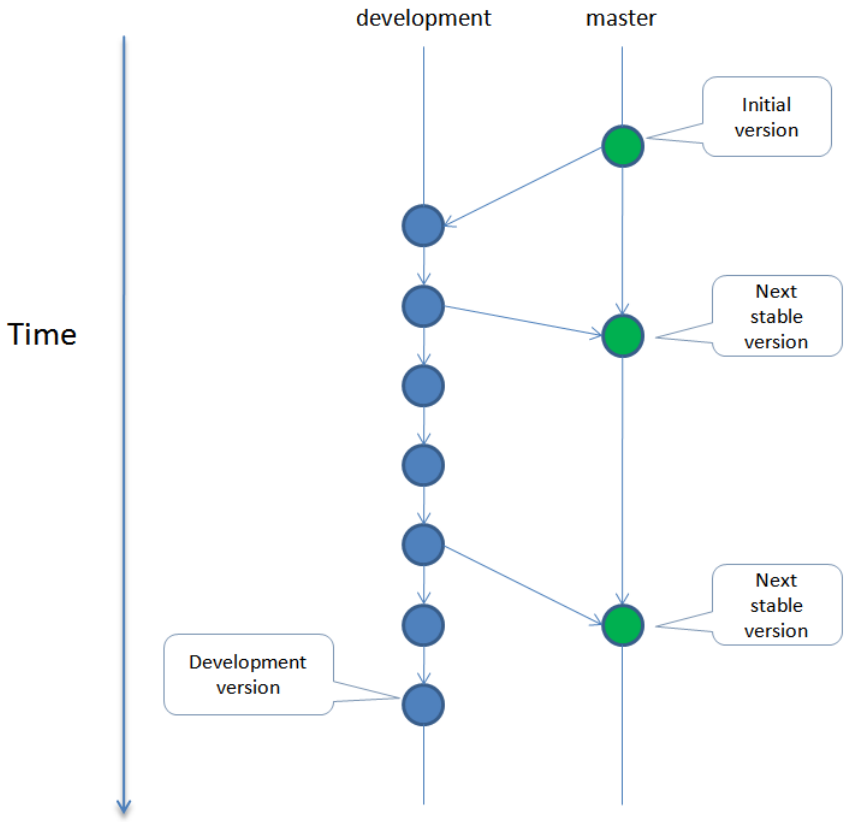
Tabuľka 12.1: zoznam procesov v kontexte verziovania zdrojových súborov a prislúchajúce role členov tímu

12.2 Verziovanie

Hlavné vetvy vývoja

Manažér kvality v spolupráci s manažérom plánovania neustále dohliada na stav zdrojových súborov z pohľadu verziovania za účelom udržania konzistentnosti a bezchybovosti koreňovej vetvy vývoja projektu. Túto vetvu možno požadovať za produkčnú, a preto je nutnosťou udržiavať ju v stave, ktorý kedykoľvek umožňuje jej reálne nasadenie. Musí za každých okolností obsahovať len dokončenú a riadne otestovanú funkcionalitu.

Hlavný vývoj projektu prebieha na hlavnej vývojovej vetve, ktorej obsah môžu tvoriť aj rozpracované, nedokončené časti funkcionality, prípadne obsah, ktorý je pripravený na testovanie. Iba za predpokladu, že je táto vetva stabilná (je riadne odovzdaná funkcionalita špecifického časového úseku), je možné vykonať zlúčenie tejto vetvy s koreňovou vetvou vývoja, čím sa zachová jej produkčný stav (Obr. 12.1).



Obr. 12.1: Koreňová a hlavná vývojová vetva v kontexte času a stability

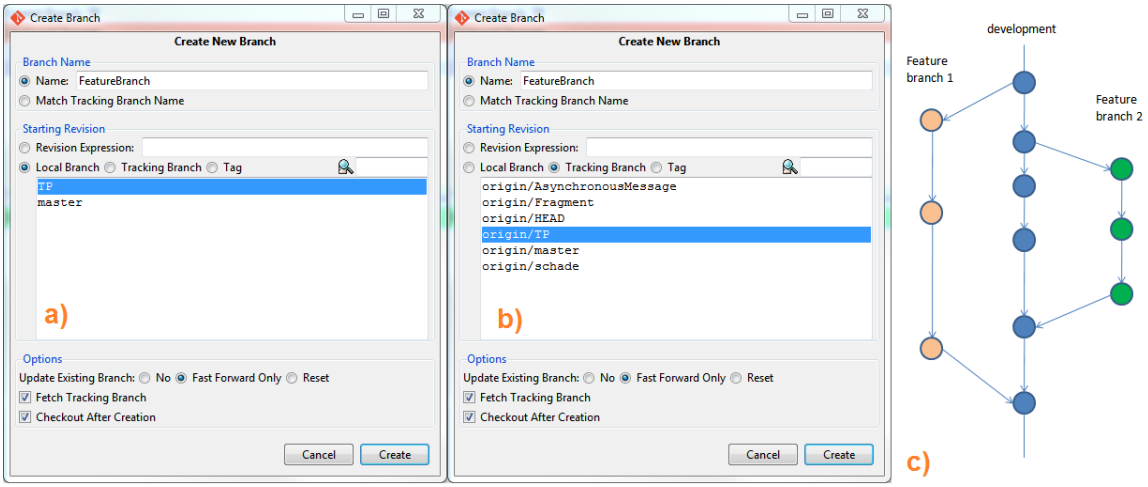
Vývoj novej funkcionality v kontexte verziovania

Vstup: úloha obsahujúca potrebu vývoja novej funkcionality

Výstup: vetva vývoja obsahujúca sériu revízií obsahujúcich vypracovanú funkcionalitu

Vývoj novej funkcionality si vždy vyžaduje vytvorenie novej vetvy vývoja. Tento druh vetvy je v kontexte času považovaný za dočasný, jeho existencia zaniká spoločne s ukončením vývoja danej funkcionality. Vývojár má dve možnosti, ako vytvoriť novú vetvu:

- Vytvorenie vetvy vývoja z lokálnej vývojovej vetvy (Obr. 12.2 a),
- vytvorenie novej vetvy vývoja ako odnož zo vzdialenej serverovej vetvy (Obr. 12.2 b).



Obr. 12.2: a) Vytvorenie novej vetvy vývoja z lokálnej vetvy, b) vytvorenie novej vetvy vývoja zo vzdialenej vetvy, c) názorný náčrt použitia funkcionálnej vývojovej vetvy

Túto operáciu je možné vykonať aj pomocou príkazového riadku. Na vytvorenie novej lokálnej vetvy vývoja z hlavnej vývojovej vetvy je nutné vykonať nasledujúci príkaz. (Všetky príkazy je nutné vykonávať v pracovnom adresári rovnajúcemu sa koreňovému adresáru lokálneho git repozitára alebo v jeho podadresároch):

```
git checkout -b <feature_branch_name> <development_branch_name>
```

V prípade vytvorenia vetvy zo vzdialenej serverovej vetvy je nutné vykonať nasledujúce príkazy v poradí, v akom sú uvedené:

```
git fetch origin
```

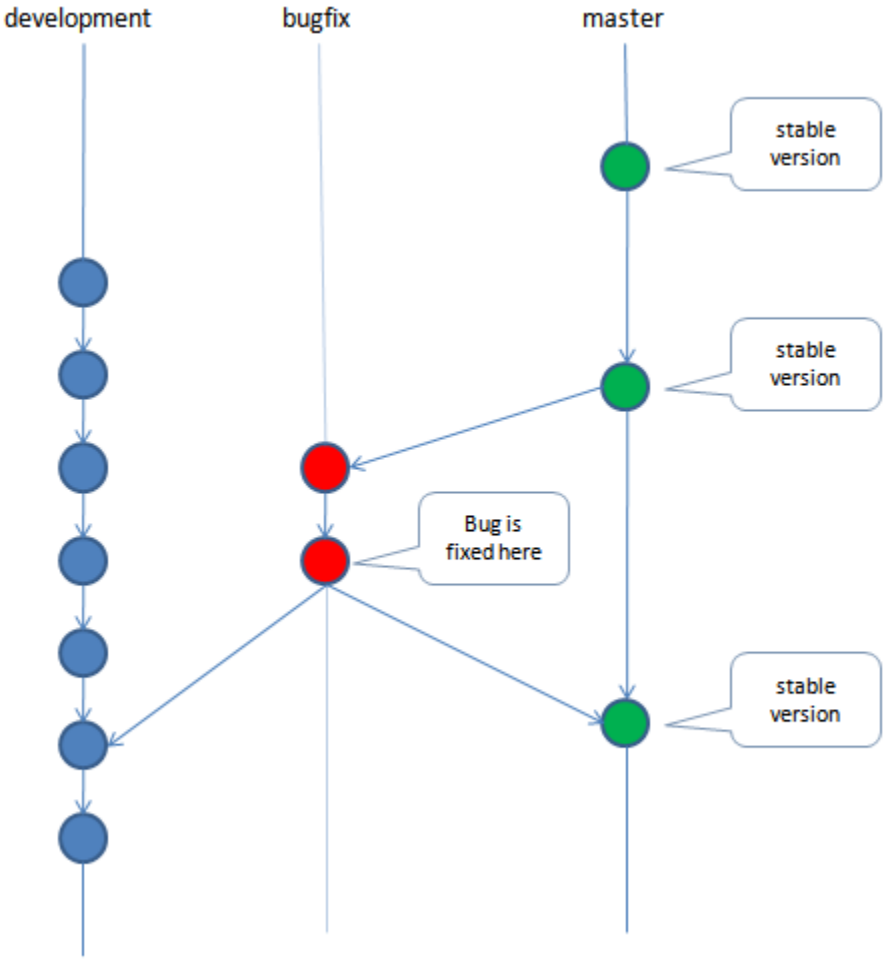
```
git checkout -b <feature_branch_name> origin/<development_branch_name>
```

Oprava chýb v kontexte verziovania

Vstup: chyba nachádzajúca sa v koreňovej vetve vývoja

Výstup: vetva obsahujúca revízie s opravou chyby

Potreba opravy chýb vzniká najmä vo fáze testovania produktu. Podľa definície ide o neočakávaný stav vyskytujúci sa na koreňovej vetve. Z tejto vetvy je aj vytváraná nová vetva určená na opravu nájdenej chyby. Proces vytvorenia vývojovej vetvy je priblížený v kapitole 2.2. Schému použitia možno vidieť na Obr. 12.3. Vetvu, ktorá po oprave chyby obsahuje jednu a viac revízií je potrebné zlúčiť ako do koreňovej vetvy vývoja, tak aj do hlavnej vývojovej vetvy. Proces zlučovania vetiev je priblížený v kapitole 2.4.



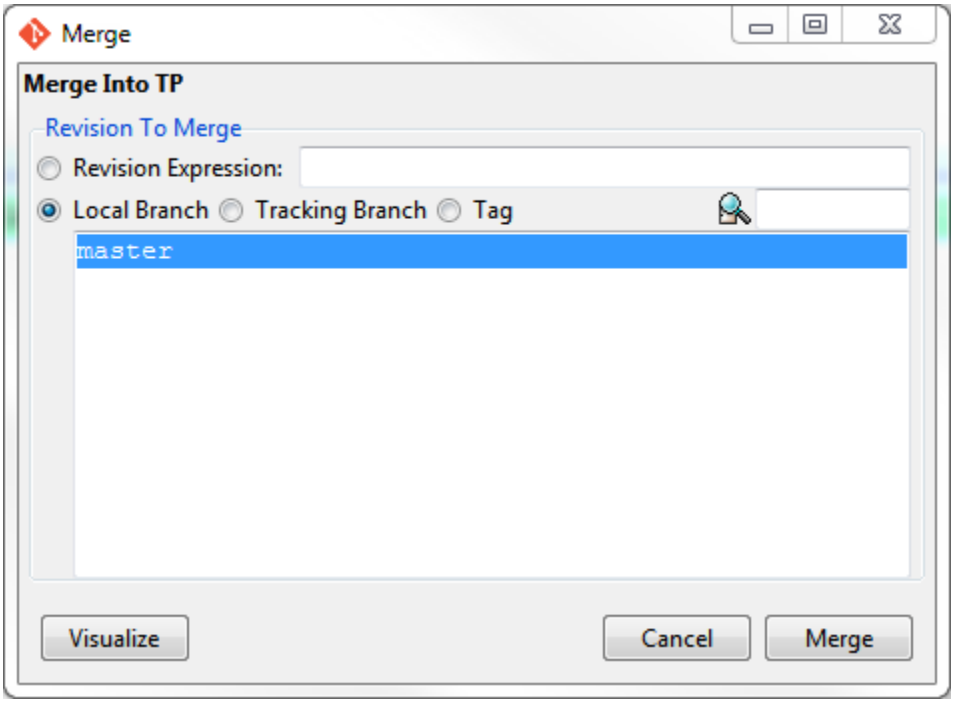
Obr. 12.3: Náčrt použitia vývojovej vetvy za účelom opravy chýb

Zlučovanie vývojových vetiev

Vstup: ukončená vetva vývoja pripravená na zlúčenie

Výstup: hlavná vetva vývoja alebo koreňová vetva vývoja obohatená o zdrojové kódy zlučovaných vetiev

Výstupmi procesov uvedených v kapitolách 2.2 a 2.3 sú vývojové vetvy obsahujúce novú funkcionalitu alebo opravu chýb. Ďalším postupom v kontexte verziovania je ich zlúčenie s ich rodičovskými (v niektorých prípadoch aj s ďalšími) vetvami. Pri samotnom procese zlučovania môžu nastať konflikty v zdrojových súboroch, avšak ich riešenie je nad rámec tejto metodiky. Postup možno vidieť na Obr. 12.4.



Obr. 12.4: Postup zlúčenia vývojových vetiev pomocou štandardného nástroja Git GUI

Alternatívou je vykonanie nasledujúceho príkazu v príkazovom riadku, pričom aktuálna pracovná vetva musí byť tá, do ktorej majú zmeny z ukončenej vetve smerovať:

```
git merge --no-ff <branch_to_merge>
```

Je dôležité uviesť aj nastavenia `--no-ff`, ktoré zabezpečujú uchovanie záznamu o existencii zlučovanej vetvy, čo umožňuje prácu s ňou aj v budúcnosti.

12.3 Sumarizácia procesov

V tejto časti metodika ponúka prehľadný súhrn a následnosť procesov opísaných v kapitolách 2.2, 2.3 a 2.4.

Použitie vetvy na vývoj novej funkcionality	**Použitie vetvy na opravu existujúcich chýb**
Podproces 1: Vytvorenie novej vetvy vývoja z hlavnej vývojovej vetvy *(Obr. 12.2)*	*Podproces 1:* Vytvorenie novej vetvy vývoja z koreňovej vetvy vývoja *(Obr. 12.2)*
Podproces 2: Pridanie revízie	*Podproces 2:* Pridanie revízie
Podproces N-1: Pridanie revízie	*Podproces N-1:* Pridanie revízie
Podproces N: Zlúčenie vetvy obsahujúcej novú funkcionalitu do hlavnej vývojovej vetvy *(Obr. 12.4)*	*Podproces N:* Zlúčenie vetvy obsahujúcej novú funkcionalitu do koreňovej vetvy vývoja ako aj do hlavnej vývojovej vetvy *(Obr. 12.4)*

Tabuľka 12.2: príklady procesov a ich následností v kontexte metodiky

A Zápisnice zo stretnutí

A.1 Stretnutie č. 01 (03.10.2013)

Dátum: 03.10.2013
Miestnosť: Jobsovo softvérové štúdio (FIIT STU)
Vyhotovil: Bc. Matej Štetiar

Prítomní:

Pedagóg:	Ing. Ivan Polášek, PhD.		
Členovia tímu:	Bc. Brndiarová Gabriela,	Bc. Kyseľ Peter,	Bc. Martoš Ivan,
	Bc. Štajer Andrej,	Bc. Štetiar Matej,	Bc. Šuta Erik,
	Bc. Valko Andrej		

Téma stretnutia

Úvod do projektu

Vyhodnotenie úloh z predchádzajúceho stretnutia

Doposiaľ neboli zadané žiadne úlohy.

Priebeh stretnutia

1. Roly v tíme si rozdeľujú členovia tímu medzi sebou na základe dohody.
2. Do pondelka treba spraviť aspoň základy web stránky. Na stránku by sa mali dať uploadovať dokumenty.
3. Pri tvorbe plagátu má tím voľnú ruku.
4. Prototyp, na ktorý bude projekt nadväzovať, má študent ing. štúdia Matej Škoda. Vedúci ho kontaktoval, aby nám sprístupnil zdrojové kódy.
5. V prípade problémov môžeme kontaktovať Mateja Škodu a to:
    a. mailom – iba dôležité otázky a vedúci musí byť v kópií,
    b. osobne – iba v prípade núdze,
    c. ísť za ním do Gratex-u – po dohode s vedúcim.
6. Prototyp, na ktorom budeme pracovať, sa bude vyvíjať týmito smermi:
    a. import/export dát do XMI – študenti v rámci záverečných prác,
    b. plugin do Eclipse – študenti v rámci záverečných prác,
    c. pridanie funkcionality pre editáciu – študent v rámci záverečnej práce,
    d. rendering obrazu,
    e. sekvenčný diagram – my v rámci tímového projektu.

Úlohy do budúceho stretnutia

#	Popis úlohy	Zodpovedné osoby	Dátum dokončenia
1.	Získať prístup k prototypu	Všetci	ASAP
2.	Naštudovať si prototyp	Všetci	ASAP
3.	Štúdium OGRE	Všetci	ASAP
4.	Vytvoriť web stránku	Tomáš	07.10.2013
5.	Vytvoriť logo	Andrej V.	07.10.2013
6.	Vytvoriť plagát	Andrej Š., Erik, Gabika, Matej	07.10.2013

A.2 Stretnutie č. 02 (10.10.2013)

Dátum: 10.10.2013
Miestnosť: Jobsovo softvérové štúdio (FIIT STU)
Vyhotovil: Bc. Erik Šuta

Prítomní:

Pedagóg: Ing. Ivan Polášek, PhD.
Členovia tímu: Bc. Brndiarová Gabriela, Bc. Kolačkovský Tomáš, Bc. Kyseľ Peter, Bc. Martoš Ivan, Bc. Štajer Andrej, Bc. Štetiar Matej, Bc. Šuta Erik, Bc. Valko Andrej

Téma stretnutia

Riešenie situácie spôsobenej zmenou špecifikácie – použitej technológie a iné.

Vyhodnotenie úloh z predchádzajúceho stretnutia

#	Popis úlohy	Zodpovedné osoby	Dátum dokončenia	Status
1.	Získať prístup k prototypu Všetci	Všetci	ASAP	OK
2.	Naštudovať si prototyp	Všetci	ASAP	Nesplnené
3.	Naštudovať si grafickú knižnicu ORGE	Všetci	ASAP	Nesplnené
4.	Vytvoriť web stránku (základný model, ktorý musí umožňovať aspoň upload dokumentov)	Tomáš	07.10.2013	OK
5.	Vytvoriť logo	Andrej V.	07.10.2013	OK
6.	Vytvoriť plagát	Andrej Š., Erik, Gabika, Matej	06.10.2013	OK

Priebeh stretnutia

1. Na úvod stretnutie prebehla debata o dostupných kolaboratívnych nástrojoch (jira, version control system)
2. V ďalšom semestri je možná spolupráca s Matejom (autor prototypu) – cieľom by bola implementácia prepojenia 3D class diagramov a naších sekvenčných diagramov do jedného fungujúceho celku,
3. Prebehla debata ohľadom existujúcej dokumentácie k implementácii prototypu – po komunikácii s Matejom máme ako zdroje vytlačenú diplomovú prácu a ďalšia (technická) dokumentácia sa nachádza priamo v zdrojových súboroch.
4. Prebehla rýchla prehliadka vytlačenej diplomovej práce.
5. Prebehla debata o možnostiach použitia nami preferovaného vývojového prostredia MS Visual Studio namiesto Eclipse
6. Vzhľadom na zásadnú zmenu používanej technológie prebehli zmeny na jednotlivých manažérskych pozíciách:
    - Manažér kvality: Matej, Andrej V.,
    - Manažér rozvrhu a plánovania: Gabika,
    - Vedúci tímu: Ivan,

- Manažér rizík: Andrej Š., Erik,
- Manažér monitorovania produktu: Tomáš,
- Manažér dokumentovania: Peter

Úlohy do budúceho stretnutia

#	Popis úlohy	Zodpovedné osoby	Dátum dokončenia
1.	Porovnanie bitbucket jira a FIIT jira (SCRUM plugin dôležitý)	Matej	11.10.2013
2.	Zistiť dostupnosť dokumentačného nástroja Confluence na FIIT	Erik	17.10.2013
3.	Alternatívy umožňujúce tvorbu bussiness dokumentácie (export do .pdf dôležitý)	Peter	17.10.2013
4.	Overiť techniky automatických testov pod C++	Tomáš	17.10.2013
5.	Rozbehanie Matejovho prototypu v MS VS	Ivan, Gabika	17.10.2013
6.	Analyzovať základnú architektúru prototypu	Andrej	17.10.2013
7.	Preštudovať základné OGRE príklady a dokumentáciu	Všetci	ASAP
8.	Migrácia stránky na virtuálku	Ivan, Erik, Tomáš	24.10.2013
9.	Zvážiť možnosť zmeny distribúcie na FIIT virtuálke – Ubuntu	Ivan, Erik, Tomáš	17.10.2013
10.	Raz týždenne migrovať obsah nášho google drive na virtuálku	Peter	Každý týždeň
11.	Pridať na stránku zápisnice a zaslať link spoločne s plagátom vedúcemu	Erik, Tomáš	10.10.2013
12.	Pripraviť dokument a krátke školenie o základoch git-u	Erik	15.10.2013
13.	Naštudovať, pochopiť a aplikovať techniky SCRUM a ich best practices (vybrané)	Všetci	17.10.2013

A.3 Stretnutie č. 03 (17.10.2013)

Dátum: 17.10.2013
Miestnosť: Jobsovo softvérové štúdio (FIIT STU)
Vyhotovil: Bc. Andrej Valko

Prítomní:

Pedagóg: Ing. Ivan Polášek, PhD.
Členovia tímu: Bc. Brndiarová Gabriela, Bc. Kyseľ Peter, Bc. Martoš Ivan, Bc. Štajer Andrej, Bc. Štetiar Matej, Bc. Šuta Erik, Bc. Valko Andrej Bc. Tomáš Kolačkovský

Téma stretnutia

Zhrnutie šprintu č.1 a naplánovanie úloh do šprintu č.2.

Vyhodnotenie úloh z predchádzajúceho stretnutia

#	Popis úlohy	Zodpovedné osoby	Dátum dokončenia	Status
1.	Porovnanie bitbucket jira a FIIT jira (SCRUM plugin dôležitý)	Matej	11.10.2013	OK
2.	Zistiť dostupnosť dokumentačného nástroja Confluence na FIIT	Erik	17.10.2013	OK
3.	Alternatívy umožňujúce tvorbu bussiness dokumentácie (export do .pdf dôležitý)	Peter	17.10.2013	OK
4.	Overiť techniky automatických testov pod C++	Tomáš	17.10.2013	Uzavreté
5.	Rozbehanie Matejovho prototypu v MS VS	Ivan, Gabika	17.10.2013	Uzavreté
6.	Analyzovať základnú architektúru prototypu	Andrej	17.10.2013	Prebieha
7.	Preštudovať základné OGRE príklady a dokumentáciu	Všetci	ASAP	Prebieha
8.	Migrácia stránky na virtuálku	Ivan, Erik, Tomáš	24.10.2013	Prebieha
9.	Zvážiť možnosť zmeny distribúcie na FIIT virtuálke – Ubuntu	Ivan, Erik, Tomáš	17.10.2013	OK
10.	Raz týždenne migrovať obsah nášho google drive na virtuálku	Peter	Každý týždeň	Prebieha
11.	Pridať na stránku zápisnice a zaslať link spoločne s plagátom vedúcemu	Erik, Tomáš	10.10.2013	OK
12.	Pripraviť dokument a krátke školenie o základoch git-u	Erik	15.10.2013	OK
13.	Naštudovať, pochopiť a aplikovať techniky SCRUM a ich best practices (vybrané)	Všetci	17.10.2013	OK

Priebeh stretnutia

1. V úvode sa preberal výber zvyšných podporných nástrojov. Rozhodlo sa, že bussiness dokumentáciu bude integrovať a spravovať jeden človek (Peter). Navrhlo sa vybrať aj nástroj na tvorbu dokumentačných UML diagramov. Zvažované boli Astah a Software ideas modeler.
2. Krátko sa vyhodnotili niektoré minulé záležitosti (Vytvorenie branch v repozitári s existujúcim prototypom, spustenie a funkcionalita prototypu, webová stránka tímu).
3. Definitívne sa rozhodlo pre prácu vo vývojovom prostredí Eclipse.
4. Zhodnotila sa efektivita práce v tíme.
5. Matej zhrnul teóriu SCRUMu.
6. Prebehla debata o ďalšom smerovaní projektu. Práca bude prebiehať na module pre sekvenčný diagram.
7. Zvažovalo sa zapojenie do TPCupu.
8. Vytvoril sa prvotný backlog projektu. Každý člen navrhol niekoľko funkcionalít. Pedagogický vedúci (Ivan Polášek) predložil svoju víziu. Funcionality boli napísané ako backlog úlohy zatiaľ v papierovej podobe. Plánuje sa prepis do JIRA (Agile)
9. Úlohy boli rozdelené do kategórií a boli identifikované najdôležitejšie.
10. Stretnutia sa zúčastnil aj autor existujúceho prototypu, s ktorým prebehla konzultácia.

Úlohy do budúceho stretnutia

#	Popis úlohy	Zodpovedné osoby	Dátum dokončenia
1.	Tvorba dokumentácie k dielu a k riadeniu projektu	Peter	Počas celého projektu
2.	Vyčistenie Git repozitára	Erik	19.10.2013
3.	Overiť techniky automatických testov pod C++	Andrej V.	20.10.2013
4.	Vytvorenie tlačidla na vytvorenie objektu, vytvorenie lifeline objektu	Gabika	24.10.2013
5.	Úprava webovej stránky	Tomáš	
6.	Príprava prototypu na implementáciu funkcionalít pre sekvenčný diagram	Ivan	24.10.2013
7.	Analyzovať základnú architektúru prototypu	Andrej Š.	24.10.2013
8.	Výber nástrojov na tvorbu UML	Erik, Andrej Š.	18.10.2013
9.	Štúdium UML	Všetci	24.10.2013
10.	Štúdium prototypu	Všetci	24.10.2013

Príloha

Backlog úlohy:

- Create object
- Create class (zahrnúť do create object)
- Create interaction (+ typy)
- Activation block
- Create fragment
- Delete object (from layer)
- Delete object (from model)
- Delete interaction (from layer)
- Delete interaction (from model)
- Update object

- Update interaction
- Object tree
- Copy layer
- Copy object
- Filter
- Animations
- Control interface
- Serialization (save, load)
- Autosave
- Full screen layers
- Edit size
- Edit layer distance
- Edit object distance
- Key shortcuts, moving
- Zooming
- Edit fragment (resize, move, copy, edit condition, edit fragment type...)

A.4 Stretnutie č. 04 (24.10.2013)

Dátum: 24.10.2013
Miestnosť: Jobsovo softvérové štúdio (FIIT STU)
Vyhotovil: Bc. Gabriela Brndiarová

Prítomní:

Pedagóg: Ing. Ivan Polášek, PhD.
Členovia tímu: Bc. Brndiarová Gabriela, Bc. Kyseľ Peter, Bc. Martoš Ivan, Bc. Štajer Andrej, Bc. Štetiar Matej, Bc. Šuta Erik, Bc. Valko Andrej

Téma stretnutia

Zhrnutie šprintu č.1 a naplánovanie úloh do šprintu č.2.

Vyhodnotenie úloh z predchádzajúceho stretnutia

#	Popis úlohy	Zodpovedné osoby	Dátum dokončenia	Percentá
1.	Tvorba dokumentácie k dielu a k riadeniu projektu	Peter	Počas celého projektu	-
2.	Vyčistenie Git repozitára	Erik	19.10.2013	100%
3.	Overiť techniky automatických testov pod C++	Andrej V.	20.10.2013	100%
4.	Vytvorenie tlačidla na vytvorenie objektu, vytvorenie lifeline objektu	Gabika	24.10.2013	100%
5.	Úprava webovej stránky	Tomáš	24.10.2013	100%
6.	Príprava prototypu na implementáciu funkcionalít pre sekvenčný diagram	Ivan	24.10.2013	100%
7.	Analyzovať základnú architektúru prototypu	Andrej Š.	24.10.2013	0%
8.	Výber nástrojov na tvorbu UML	Erik, Andrej Š.	18.10.2013	100%
9.	Štúdium UML	Všetci	24.10.2013	100%
10.	Štúdium prototypu	Všetci	24.10.2013	100%

Priebeh stretnutia

1. Boli dohodnuté niektoré nové metodiky práce:
    - do nástroja JIRA sa k programátorským úlohám do komentára zapíše aj príslušný hashcode (revision number) commit-u do Git-u
    - nebudú sa využívať komentáre commit-ov a pripomienkovať sa bude prostredníctvom komentárov ku úlohe (JIRA)
2. Zhrnutie doterajších výstupov našej práce:
    - dokumentácia je hotová aj so zapracovanými pripomienkami, ktoré sa behom týždňa vyskytli
    - vytváranie nového grafického objektu reprezentujúceho lifeline má problém s osvetlením (vzniká esteticky nepríjemný tieň) a prerušovaná čiara je fixnej dĺžky
    - všetci boli informovaní, že sa v zdrojovom kóde zaviedla statická premenná, pomocou ktorej sa prototyp prepína medzi už existujúcim diagramom tried

a našim sekvenčným diagramom a budeme ju v prípade potreby naďalej pri našej práci používať

3. Odsúhlasili sme, že nechceme používať knižnicu MyGui. Erik sa podujal, že nájde alternatívu (kým sa nenájde niečo vyhovujúcejšie, tak pokračujeme s MyGui, aby sa nebrzdil vývoj).
4. Dohodla sa scrum realizácia platná od dnes:
    - user stories sa definujú s menšou granularitou ako na poslednom stretnutí
    - priority úloh – 1, 2, 3, atď. (1 znamená najväčšiu prioritu)
    - zložitosti úloh – veľkosti tričiek (XS, S, M, L, XL, XXL)
    - stav úlohy na konci šprintu (dokončenie) – percentá (100% znamená, že úloha je hotová)
5. Názor vedúceho na webovú stránku tímu:
    - Tomáš dostal pochvalu za prevedenie stránky
    - vyjadrený negatívny postoj ku stolčeku, na ktorom stojí logo aj k logu samotnému (odporúčaná zmena stolčeka a zostrenie hrán loga, ktoré by malo eliminovať asociáciu s televízorom)
    - návrh novej funkcionality: zmeny projekcie v tieni loga po kliknutí naň (nutná ešte konzultácia s Tomášom)
6. Dohodnutie formy ukončovania šprintu:
    - vedúcemu budú prvýkrát reprezentované výstupy na štvrtkovom stretnutí o 12:00
    - neoficiálny termín ukončenia sme si stanovili na štvrtok 6:00 ráno
    - výstup šprintu pozostáva z:
        - ukážky bežiacej aplikácie vedúcemu na stretnutí
        - dokumentácia (má obsahovať aj screenshot) na stránke
        - tabuľka so stavom úloh na stránke
7. Dal sa priestor pre riešenie problémov nášho tímu. Nikto nič nehlásil.
8. Zmeny v kóde, ktoré súvisia aj s diagramom tried budú zapisované do samostatného dokumentu a následne referované Matejovi Škodovi.
9. Zadefinovali sa ciele pre šprint č.2:
    - lifeline čiara nepresahuje vrstvu
    - zarovnávanie lifeline box-ov do riadkov
    - definovanie lifeline atribútov
    - ukladanie a načítavanie diagramu
    - znázornenie interakcie medzi objektami
10. Pri vývoji nebudeme využívať automatické testovanie.
11. Každý si zdokumentuje úlohu, ktorú mal na starosti (pre dokumentáciu k inžinierskemu dielu) a v manažérsku rolu, ktorú zastáva (pre dokumentáciu riadenia).

Úlohy do budúceho stretnutia

#	Popis úlohy	Zodpovedné osoby	Dátum dokončenia
1.	Získať prístup k prototypu	Všetci	ASAP
2.	Naštudovať si prototyp	Všetci	ASAP
3.	Naštudovať si grafickú knižnicu ORGE	Všetci	ASAP
4.	Vytvoriť web stránku (základný model, ktorý musí umožňovať aspoň upload dokumentov)	Tomáš	07.10.2013
5.	Vytvoriť logo	Andrej V.	07.10.2013
6.	Vytvoriť plagát	Andrej Š., Erik, Gabika, Matej	07.10.2013

A.5 Stretnutie č. 05 (7.11.2013)

Dátum: 07.11.2013
Miestnosť: Jobsovo softvérové štúdio (FIIT STU)
Vyhotovil: Bc. Peter kyseľ

Prítomní:

Pedagóg: Ing. Ivan Polášek, PhD.
Členovia tímu: Bc. Brndiarová Gabriela, Bc. Kyseľ Peter, Bc. Martoš Ivan, Bc. Štajer Andrej, Bc. Štetiar Matej, Bc. Šuta Erik, Bc. Valko Andrej

Téma stretnutia

Vyhodnotenie druhého šprintu a špecifikácia tretieho šprintu

Vyhodnotenie úloh z predchádzajúceho stretnutia

#	Popis úlohy	Zodpovedné osoby	Dátum dokončenia	Percentá
1.	Hľadanie alternatívy ku knižnici MyGui	Erik, Peťo	7.11.2013	60%
2.	Ukladanie, načítavanie diagramu	Erik	7.11.2013	70%
3.	Lifeline pretekanie	Matej	7.11.2013	100%
4.	Lifeline obal	Matej	7.11.2013	100%
5.	Lifeline zarovnanie na riadok	Ivan	7.11.2013	90%
6.	Interakcia – asynchrónne správy	Andrej Š., Andrej V.	7.11.2013	100%
7.	Zdokumentovať šprint č.1	Peťo, Gabika	7.11.2013	100%
8.	Spojenie menších celkov do dokumentácie pre šprint č.2	Gabika	7.11.2013	100%
9.	Napísanie prihlášky do TP Cup-u.	Peťo	28.10.2013	100%
10.	Lifeline atribúty	Gabika	7.11.2013	100%
11.	Popis k manažérskej role	Všetci	7.11.2013	100%
12.	Pridanie novej vrstvy	Gabika	7.11.2013	100%

Priebeh stretnutia

1. Erik podal návrh na vytvorenie dátového modelu nášho prototypu. Preberala sa metodika komentovania kódu. Takisto aj odpovedal na nejasnosti ohľadom gitu.
2. Preberanie problémov počas šprintov, vylepšení a úloh budúcej práce. Po príchode vedúceho tímu sme prebrali niekoľko vecí ohľadom TP CUP-u , web stránku a formu vyhodnotenia šprintu.
3. Zhrnuli sme úlohy vykonané počas šprintu diskusiou o ich priebehu, problémoch a riešeniach.
4. Dôležité úlohy, ktoré je potrebné vykonať v nasledujúcom šprinte:
    - vytváranie class diagramu popri sekvenčnom diagrame (pridanie metódy, triedy)
    - abstrahovanie informácií do repozitára
    - rozlíšenie asynchrónnej interakcie
    - formátovanie textu objektov (nízka priorita)
    - základný skelet fragmentov (vysoká priorita)

5. Kontaktovať vedúceho tímu ohľadom mien bakalárov, ktorí sa zaoberajú serializáciou. Takisto je nutné nahrať zápisnicu a výsledky šprintu na web stránku.
6. Po návrhu úloh pre nasledujúci šprint (tabuľka nižšie), vedúci navrhol stretnutie s bakalármi ako aj s autorom pôvodného prototypu (bližšie informácie o konaní stretnutia budú dodatočne doručené).

Úlohy do budúceho stretnutia

#	Popis úlohy	Zodpovedné osoby	Dátum dokončenia	Priorita/ Zložitosť	
1.	Delete (lifeline,layer,interaction)	Peťo	21.11.2013	1	L
2.	Opraviť server a pridať (zápisnicu,šprint1-2,do galérie fotky)	Peťo	21.11.2013 ASAP	1	S
3.	Max width lifeline	Ivan	21.11.2013	1	S
4.	Diagram dátový model	Gabika	21.11.2013	1	M
5.	Otázky na refaktorizáciu	všetci	21.11.2013	1	
6.	Width heigth	Ivan	21.11.2013	1	XS
7.	Serializacia	Erik	21.11.2013	1	M
8.	Označnie lifelines ťahom	Maťo	21.11.2013	1	L
9.	Návrh fragmentu	Andrej V.	21.11.2013	1	M
10.	Algoritmus states pridania fragmentu	Andrej Š.	21.11.2013	1	M
11.	Text zalamovanie	Ivan	21.11.2013	1	S
12.	Dátový návrh a implementácia fragmentov	Andrej Š.	21.11.2013	1	M
13.	Dokumentácia	Erik	21.11.2013	1	M

A.6 Stretnutie č. 06 (14.11.2013)

Dátum: 14.11.2013
Miestnosť: Jobsovo softvérové štúdio (FIIT STU)
Vyhotovil: Bc. Ivan Martoš

Prítomní:

Pedagóg: Ing. Ivan Polášek, PhD.
Členovia tímu: Bc. Brndiarová Gabriela, Bc. Kyseľ Peter, Bc. Martoš Ivan, Bc. Štajer Andrej, Bc. Štetiar Matej, Bc. Šuta Erik, Bc. Valko Andrej

Téma stretnutia

Zhrnutie doterajšieho priebehu šprintu, zmeny v riadení tímu a stretnutie celého 3D UML projektu

Vyhodnotenie úloh z predchádzajúceho stretnutia

#	Popis úlohy	Zodpovedné osoby	Dátum dokončenia	Status
1.	Údržba JIRY, spustenie Agile časti vrátane Burndown chart	Andrej Š.	14.11.2013	OK
2.	Vytvorenie Poker Cards	Andrej V.	14.11.2013	OK

Priebeh stretnutia

1. Andrej Š. ukázal vykonané zmeny v JIRE a vysvetlil ako sa pracuje s Agile Dashboard
2. Andrej V. poukázal na chyby v kóde
3. Plánovalo sa budúco-týždňové stretnutie celého 3D-UML projektu. Do plánovania bol zahrnutý aj Matej Škoda cez Skype. Počas rozhovoru sa diskutovalo a dospelo ku nasledujúcim záverom:
    a. Potreba zahrnúť taktiež bakalárov do stretnutia
    b. Potreba prediskutovať počas stretnutia vytváranie Fragmentov. Ivan Polášek navrhol samostatné riešenie pre všetky strany ktoré sa neskôr spojí na základe super-class
    c. Potreba prediskutovať Layers
    d. Potreba pripraviť sa na stretnutie a spísať naše návrhy na zmeny
    e. Potreba zavedenia jednotného naming conventions
    f. Potreba dokumentovať kód
    g. Požiadavka o to aby Matej Škoda pushoval jeho verziu do repozitára

Úlohy do budúceho stretnutia

#	Popis úlohy	Zodpovedné osoby	Dátum dokončenia
1.	Pokračovanie na práci v aktuálnom šprinte	GAMATEPI	21.22.2013
2.	Spoločné stretnutie s Matejom Škodom a bakalármi	GAMATEPI	21.22.2013

A.7 Stretnutie č. 07 (21.11.2013)

Dátum: 21.11.2013
Miestnosť: Jobsovo softvérové štúdio (FIIT STU)
Vyhotovil: Bc. Andrej Štajer

Prítomní:

Pedagóg: Ing. Ivan Polášek, PhD.
Členovia tímu: Bc. Brndiarová Gabriela, Bc. Kyseľ Peter, Bc. Martoš Ivan, Bc. Štajer Andrej, Bc. Štetiar Matej, Bc. Šuta Erik, Bc. Valko Andrej

Téma stretnutia

Odovzdanie šprintu, príprava na nasledujúci šprint

Vyhodnotenie úloh z predchádzajúcich šprintov

#	Popis úlohy	Zodpovedné osoby	Dátum dokončenia	Zložitosť	Stav
1.	Delete (lifeline,layer,interaction)	Peťo	21.11.2013	L	40%
2.	Opraviť server a pridať (zápisnicu,šprint1-2,do galérie fotky)	Peťo	21.11.2013 ASAP	S	0%
3.	Max width lifeline	Ivan	21.11.2013	S	95%
4.	Diagram dátový model	Gabika	21.11.2013	M	100%
5.	Otázky na refaktorizáciu	všetci	21.11.2013		
6.	Width heigth	Ivan	21.11.2013	XS	95%
~~7.~~	~~Serializacia~~	~~Erik~~	~~21.11.2013~~	~~M~~	
8.	Označnie lifelines ťahom	Maťo	21.11.2013	L	60%
9.	Návrh fragmentu	Andrej V.	21.11.2013	M	95%
10.	Algoritmus states pridania fragmentu	Andrej Š.	21.11.2013	M	95%
11.	Text zalamovanie	Ivan	21.11.2013	S	95%
12.	Dátový návrh a implementácia fragmentov	Andrej Š.	21.11.2013	M	80%
13.	Dokumentácia	Erik	21.11.2013	M	69%

*95% chýba review

**90% chýba dokumentácia a review

Priebeh Stretnutia

1. Vyhodnotenie úloh predchádzajúceho šprintu
    a. Server nie je stabilný – nutné skontrolovať. Peter skúšal viackrát počas šprintu a nepodarilo sa mu pripojiť.
    b.
    c. Počas šprintu prebehol telefonát ohľadom refaktorizácie. Matej Škoda mal pushnúť commit na server. Commity neboli analyzované. Nakoľko boli dodané neskoro.
    d. Serializácia premenovaná a zmenený aj charakter úloh tomu určených.

2. Ivan pridal nový stav do algoritmu stavov na výpis chyby a vysvetlil algoritmus
3. Počas šprintu prebehla oprava nájdených chýb na úrovni kódu – Matej, Gabika, Andrej,Ivan
4. Konzultácie ohľadom commitovania a merge commitov jednotlivých branches. – Matej Andrej S, Andrej V
5. Úloha k GUI bude uzavretá
6. Komentovanie kódu je nekonzistentné, tak isto aj zadávanie úloh
    a. Jira – úlohy pridáva iba jedna(dve) poverené osoby.
    b. Kód – komentovanie podľa vytvorenej špecifikácie.
7. Matej a Gabika pracovali na makre pre automatizáciu exportu reportu z Jiry do Excelu.
8. Retrospespetíva šprintu
    a. Start –
        i. Začať planning poker s kartičkami vytvorenými na to určenými.
        ii. Používať burn-down chart tak ako sa má
            - Diskusia ohľadom prioritnej práce. Najprv je potrebné splniť naplánované a potom ďalšieho úlohy,
        iii. Používanie googleGroup ako úložisko znalostí
        iv. Používať prepojenie Jira-Bitbucket repozitár
        v. Oficiálne začať manažmentom rizík - riziká sa kontrolovali bez
        vi. Plánovať v šprinte čas na opravu chýb, dokumentáciu a code review.
        vii. Dokončiť a používať makro na automaticky export z Jiry
        viii. Formalizovať monitorovanie projektu
    b. Stop
        i. Používanie GoogleGroups ako komunikačného kanála
        ii. Neustále menenie plánu určeného na začiatku semestra
        iii. Robenie úloh na rámec plánu na základe vlastného uváženia
        iv. Otvárať rovnaké témy, ktoré boli uzavreté bez nových znalostí
    c. Continue
        i. Používanie nástroja Jira
        ii. Odhad User stories s veľkosťami tričiek
        iii. Neformálne stretnutia a komunikácia medzi členmi tímu
        iv. Dokumentovanie úloh v každom šprinte postupne
        v. Commitovanie po častiach a často do vlastných vetiev
9. Plánovanie nového šprintu
    a. Výber úloh do ďalšieho špritnu
    b. Odhad veľkosti User stories pomocou metódy planning poker

Úlohy do budúceho šprintu

kód úlohy	[nadradená úloha] popis úlohy	Zodpovedná osoba	veľkosť
TRIDUML-48	TRIDUML-61 Označenie lifeline ťahom	Matej Stetiar	L
TRIDUML-47	TRIDUML-63 Vytvorenie stavov a tlačidla	Peter Kysel	S
TRIDUML-44	Dokumentácia	Erik Suta	S
TRIDUML-94	TRIDUML-66 Zvýraznenie obalu pri prechode cez lifeline	Gabriela Brndiarova	L
TRIDUML-92	TRIDUML-61 LOOP a OPT fragmenty	Andrej Valko	L

TRIDUML-91	TRIDUML-67 Vymazanie aktuálneho modelu	Andrej Stajer	M
TRIDUML-90	TRIDUML-67 Vytváranie objektov z XML	Erik Suta	L
TRIDUML-88	TRIDUML-61 Uloženie, načítavanie	Andrej Stajer	S
TRIDUML-87	Ukladanie interakcií v objekte lifeline	Andrej Valko	XS
TRIDUML-83	TRIDUML-63 Spustenie mazania	Ivan Martos	S
TRIDUML-82	TRIDUML-63 Vytvorenie deštruktorov pre asynchrónne správy	Ivan Martos	S
TRIDUML-81	TRIDUML-63 Vytvorenie deštruktorov pre lifeline	Peter Kysel	M
TRIDUML-67	Serializacia	*Unassigned*	
TRIDUML-66	Control interface	*Unassigned*	
TRIDUML-63	Delete object	*Unassigned*	
TRIDUML-61	Create Fragment	*Unassigned*	
TRIDUML-93	Oprava chýb vo 4. šprinte	*Unassigned*	L
TRIDUML-89	TRIDUML-44 Dokumentácia ku šprintu č.4	*Unassigned*	

A.8 Stretnutie č. 08 (28.11.2013)

Dátum: 28.11.2013
Miestnosť: Jobsovo softvérové štúdio (FIIT STU)
Vyhotovil: Bc. Matej Štetiar

Prítomní:

Pedagóg:	Ing. Ivan Polášek, PhD.		
Členovia tímu:	Bc. Brndiarová Gabriela,	Bc. Kyseľ Peter,	Bc. Martoš Ivan,
	Bc. Štajer Andrej,	Bc. Štetiar Matej,	Bc. Šuta Erik,
	Bc. Valko Andrej		

Téma stretnutia

Oboznámenie sa s aktuálnym stavom úloh v šprinte číslo 4. Poukázanie na problémy, na ktoré sme narazili. Diskusia k možným zlepšeniam prototypu.

Vyhodnotenie úloh z predchádzajúceho stretnutia

[nadradená úloha] popis úlohy	Zodpovedná osoba	Stav
Označenie lifeline ťahom	Matej Štetiar	Čaká na revíziu
Vytvorenie stavov a tlačidla	Peter Kyseľ	Rozpracovaná
Dokumentácia	Erik Šuta	Rozpracovaná
Zvýraznenie obalu pri prechode cez lifeline	Gabriela Brndiarová	Rozpracovaná
LOOP a OPT fragmenty	Andrej Valko	Neriešená
Vymazanie aktuálneho modelu	Andrej Štajer	Neriešená
Vytváranie objektov z XML	Erik Šuta	Neriešená
Uloženie, načítavanie	Andrej Štajer	Rozpracovaná
Ukladanie interakcií v objekte lifeline	Andrej Valko	Uzatvorená
Spustenie mazania	Ivan Martoš	Neriešená
Vytvorenie deštruktorov pre asynchrónne správy	Ivan Martoš	Rozpracovaná
Vytvorenie deštruktorov pre lifeline	Peter Kyseľ	Rozpracovaná
Oprava chýb vo 4. šprinte	*Unassigned*	Neriešená
Dokumentácia ku šprintu č.4	*Unassigned*	Neriešená

Priebeh stretnutia

1. Ivan a Erik oznámili, že naša webová stránka nie je aktuálna. Dôvodom neaktuálnosti je problém s naštartovaním servera. Tento problém Ivan intenzívne rieši.
2. Vedúci tímu nám ukázal video s aktuálnym stavom fragmentu, na ktorom pracuje jeden jeho diplomant. Náš fragment a jeho fragment by sa mali spojiť. Preto je potrebné sa zamyslieť, či implementáciu zjednotíme teraz alebo pôjdeme dočasne vlastnou cestou.
3. Zhrnuli sme aktuálny stav úloh v tomto šprinte. Všetky úlohy sú rozpracované a problémy, na ktoré riešitelia narazili, sme si spolu vydiskutovali.
4. Úlohu uloženia a načítania scény Erik konzultoval aj s bakalármi, ktorí sa s podobnými úlohami stretli vo svojich bakalárskych projektoch.

5. Jeden z bakalárov má naprogramované mazanie prvkov scény. Porozprávali sme sa s ním o jeho riešení, čím nám pomohol pri riešení našej úlohy týkajúcej sa mazania.
6. Andrej Štajer poukázal na potrebu pridania logovania do kódu. Zhodli sme sa, že vytvorenie logovania bude zaradené do ďalšieho šprintu.

Úlohy do budúceho stretnutia

Dokončiť úlohy naplánovaná v šprinte. Pribudli tieto úlohy:

#	**Popis úlohy**	**Zodpovedné osoby**	**Dátum dokončenia**
1.	Formálne spísanie rizík a procesov ich manažmentu	Andrej Š.	05.12.2013
2.	Podrobný popis plánu	Gabriela	05.12.2013
3.	Návrh obdobia a riešenia pre spojenie implementácie fragmentu	Matej	05.12.2013

A.9 Stretnutie č. 09 (05.12.2013)

Dátum: 05.12.2013
Miestnosť: Jobsovo softvérové štúdio (FIIT STU)
Vyhotovil: Bc. Erik Šuta

Prítomní:

Pedagóg: Ing. Ivan Polášek, PhD.
Členovia tímu: Bc. Brndiarová Gabriela, Bc. Kyseľ Peter, Bc. Martoš Ivan, Bc. Štajer Andrej, Bc. Štetiar Matej, Bc. Šuta Erik, Bc. Valko Andrej

Téma stretnutia

Odovzdanie šprintu č. 4. Príprava na šprint č. 5.

Vyhodnotenie úloh z predchádzajúceho stretnutia

#	Popis úlohy	Zodpovedné osoby	Stav
1.	Označenie lifeline ťahom	Matej Štetiar	95%
2.	Vytvorenie stavov a tlačidla	Peter Kyseľ	90%
4.	Zvýraznenie obalu pri prechode cez lifeline	Gabriela Brndiarová	90%
5.	LOOP a OPT fragmenty	Andrej Valko	-
6.	Vymazanie aktuálneho modelu	Andrej Štajer	0%
7.	Vytváranie objektov z XML	Erik Šuta	85%
8.	Uloženie, načítavanie	Andrej Štajer	95%
9.	Ukladanie interakcií v objekte lifeline	Andrej Valko	100%
10.	Spustenie mazania	Ivan Martoš	95%
11.	Vytvorenie deštruktorov pre asynchrónne správy	Ivan Martoš	95%
12.	Vytvorenie deštruktorov pre lifeline	Peter Kyseľ	0%
13.	Dokumentácia ku šprintu č. 4	Erik Šuta	100%
14.	Oprava chýb v 4. šprinte – Zmiznutie fragmentu	Andrej Valko	100%
	Oprava chýb v 4. šprinte – fragment a jeho pozícia na zadných vrstvách	Andrej Valko	95%
15.	Formálne spísanie rizík a procesov ich manažmentu	Andrej Štajer	100%
16.	Podrobný opis plánu	Gabriela Brndiarová	0%
17.	Návrh obdobia a riešenia pre spojenie implementácie fragmentu	Matej Štetiar, Andrej Valko	100%

*95% chýba review

**90% chýba dokumentácia a review

Priebeh stretnutia

1. Pred príchodom vedúceho tímu sa diskutovalo o nasledujúcich témach:
    a. Všeobecné problémy predchádzajúceho šprintu a aktuálnej verzie prototypu
    b. Problém s chýbajúcou webstránkou – padlo niekoľko návrhov na riešenie, ako najschodnejšiu cestu sme vybrali riešenie pomocou jednoduchej boostrap webstránky podľa pripravenej šablóny
    c. Prebehlo krátke interné zhrnutie stavu úloh

2. Po príchode vedúceho tímu bol zhrnutý návrh profesorky Bielikovej o vzájomnej prezentácii tímových projektov, pravdepodobne začiatkom letného semestra. Prebrali sme aj ďalšie detaily, ako je hodnotenie, diskusia atď.
3. Vedúci tímu zdôraznil vhodnosť práce na tímovom projektu aj počas prestávky medzi zimným a letným semestrom
4. Prebehlo odovzdanie šprintu scrum masterovi (vedúci tímu):
    a. Prebehla debata o aktuálnom stave fragmentu a o možnosti spojenia s implementáciou fragmentu v diagrame aktivít v kontexte architektúry a konzistencie riešenia
    b. Bola navrhnutá možnosť užšej spolupráce s bakalármi pracujúcimi na funkcionalite exportu a importu diagramov do/z XML
    c. Prebehla debata o možnosti mazania diagramu zo scény (ponechanie v dátovom modely) a o kompletnom zmazaní (hard delete) a krátka rozprava o implementácii tejto funkcionality
5. Navrhli sme posledný šprint, jeho hlavnou náplňou má byť refactoring implementovanej funkcionality, finalizácia nedokončených úloh, oprava vzniknutých chýb, finalizácia dokumentácie v zimnom semestri a vytvorenie webstránky
6. Navrhli sme čas a termín prezentácie práce na tímovom projekte v kontexte predmetov MSS/MIS. Predbežný dátum sme určili na 17.12.2013. Neskôr v priebehu stretnutia bol tento dátum potvrdený aj s časom stretnutia (14:00-14:45)
7. V krátkosti sme navrhli ako by mohol vyzerať výstup tímového projektu v letnom semestri
8. Bola vykonaná retrospektíva šprintu
    a. Start
        i. XP programming sessions – v predchádzajúcom šprinte, dňa 29.11.2013 prebehla prvá XP session a chceme v tomto trende aj naďalej pokračovať (v kontexte retrospektívy sme však prebrali túto udalosť až na tomto stretnutí, preto ju uvádzame do „Start“)
        ii. Častejšie vykonávanie rebase pri funkcionálnych vetvách vývoja za účelom zníženia množstva vzniknutých konfliktov pri zlučovaní vetiev vývoja
    b. Keep
        i. Planning poker cards
        ii. Burn-down chart
        iii. Udržiavať pre tím prospešnú komunikáciu s Matejom Škodom
        iv. Používanie integrácie nástrojov Jira a Bitbucket
        v. Zahrnutie časov na dokumentovanie a code review do časových plánov na šprint
        vi. Používanie makra na automatický export dát o úlohách z nástroja Jira
        vii. Veľkosti tričiek ako metrika ohodnocovania úloh
        viii. Neformálne stretnutia a komunikácia tímu
        ix. Dokumentovanie úloh v každom šprinte
        x. Používanie funkcionálnych vývojových vetiev na implementáciu novej funkcionality
    c. Stop
        i. -
9. Prebehla detailnejšia rozprava o plánoch na nasledujúci šprint. Táto časť zahŕňa formálnu špecifikáciu úloh, následné ohodnotenie úloh pomocou planning poker cards a pridelenie úloh jednotlivým členom tímu.

Úlohy do budúceho stretnutia

kód úlohy	[nadradená úloha] popis úlohy	Zodpovedná osoba	veľkosť
TRIDUML-111	[TRIDUML-108]: Naplnenie Webovej stránky informáciami	Matej Štetiar	S
TRIDUML-115	[TRIDUML-113]: Spojenie prezentácie do jedného súboru	Unassigned	XS
TRIDUML-112	[TRIDUML-108]: Nahodenie stránky na server	Matej Štetiar	S
TRIDUML-114	[TRIDUML-113]: Vytvoriť šablónu prezentácie	Andrej Štajer	S
TRIDUML-110	[TRIDUML-108]: Vytvorenie webovej stránky	Ivan Martoš	L
TRIDUML-109	[TRIDUML-107]: Inštalácia server	Andrej Valko	L
TRIDUML-28	[TRIDUML-67]: Ukladanie diagram do XML	Erik Šuta	S
TRIDUML-91	[TRIDUML-67]: Vymazanie aktuálneho modelu	Andrej Štajer	M
TRIDUML-90	[TRIDUML-67]: Vytváranie objektov z XML	Erik Šuta	M
TRIDUML-118	[TRIDUML-116]: Pripojenie “InteractiveCover” vetvy	Gabriela Brndiarová	M
TRIDUML-117	[TRIDUML-116]: Pripojenie “delete” vetvy	Peter Kyseľ	M
TRIDUML-104	[TRIDUML-44]: Dokumentácia ku šprintu č. 5	Erik Šuta	XS
TRIDUML-96	[TRIDUML-105]: AddLifelineWindow sa nevypne po stlačení “create” tlačidla	Gabriela Brndiarová	-

B Preberací protokol

PREBERACÍ PROTOKOL ZA ZIMNÝ SEMESTER

Tímový projekt 2013/2014

Tím 02 – GAMATEPI

Predmet odovzdávania:

- Dokumentácia k riadeniu projektu – verzia za zimný semester
- Dokumentáciu k inžinierskemu dielu – verzia za zimný semester

Vedúci projektu: Ing. Ivan Polášek, PhD
Podpisom potvrdzuje prebratie vyššie uvedených častí projektu.

V Bratislave

....

Dátum Podpis