Slovenská technická univerzita

Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava 4

Odhaľovanie emocionálneho stavu používateľa

Team EmLog

Dokumentácia k riadeniu

Vedúca tímu:
Doc. Mgr. Daniela Chudá, PhD.

Členovia tímu:
Bc. Jozef Gajdoš
Bc. Martin Geier
Bc. Peter Greguš
Bc. Miroslav Hudák
Bc. Peter Sivák
Bc. Peter Šinský

Kontakt: team.10.tp@gmail.com

1 Úvod 1-1
2 Predstavenie členov tímu 2-1
3 Motívácia 3-1
4 Koncepcia riešenia 4-1
5 Zápisy zo stretnutí 5-1
5.1 Zápis č.1 5-2
5.2 Zápis č.2 5-4
5.3 Zápis č.4 5-7
5.4 Zápis č.4 5-11
5.5 Zápis č.5 5-14
5.6 Zápis č.6 5-18
6 Manažment rozvrhu a plánovania 6-1
6.1 Plán projektu 6-1
6.1.1 Šprint 0 6-1
6.1.2 Šprint 1 6-2
6.1.3 Šprint 2 6-3
7 Manažment rizík 7-1
8 Manažment kvality 8-1
9 Manažment monitorovania projektu 9-1
9.1 Šprint 0 9-1
9.2 Šprint 1 9-2
10 Manažment podpory vývoja 10-1
11 Manažment komunikácie 11-1
11.1 Metóda vývoja SCRUM 11-1
11.1.1 Backlog 11-1
11.1.2 Vytváranie podúloh a plánovanie šprintov 11-1
12 Metodiky 12-1
12.1 Metodika pre manažment chýb a tvorba záznamu o chybe v systéme Redmine 12-1
12.1.1 Úvod 12-1
12.1.2 Skratky a pojmy 12-1
12.1.3 Procesy manažmentu chýb 12-1

12.1.4 Roly a zodpovednosti účastníkov 12-3
12.1.5 Procesy v manažmente chýb 12-3
12.1.6 Súvisiace metodiky 12-5
12.1.7 Detailný opis procesu Vytvorenie záznamu o chybe 12-6
12.2 Metodika pre manažment údržby softvéru pomocou metódy rafaktorovania 12-10
12.2.1 Úvod 12-10
12.2.2 Zoznam pojmov 12-10
12.2.3 Role a zodpovednosti 12-10
12.2.4 Proces údržby softvéru 12-11
12.2.5 Proces aplikovania refaktorovania v *MS Visual Studio* 12-14
12.3 Vloženie používateľského príbehu do nástroja Redmine 12-20
12.3.1 Úvod 12-20
12.3.2 Definovanie pojmov 12-20
12.3.3 Role a zodpovednosti 12-20
12.3.4 Proces zberu požiadaviek 12-23
12.3.5 Aktualizácia požiadaviek 12-25
12.3.6 Vytvorenie akceptačných testov 12-25
12.3.7 Vloženie používateľského príbehu do nástroja Redmine 12-26
12.4 Plánovanie šprintu (scrum) A jeho evidencia v Nástroji Redmine. 12-30
12.4.1 Úvod 12-30
12.4.2 Pojmy 12-30
12.4.3 Role a zodpovednosti 12-30
12.4.4 Proces plánovania šprintu 12-31
12.4.5 Evidencia úloh v systéme redmine (nižšia úroveň) 12-35
12.5 Manažment zberu požiadaviek vloženie používateľského príbehu do nástroja Redmine 12-39
12.5.1 Úvod 12-39
12.5.2 Definovanie pojmov 12-39
12.5.3 Role a zodpovednosti 12-39
12.5.4 Grafická reprezentácia procesu zberu požiadaviek 12-41
12.5.5 Proces zberu požiadaviek 12-42

12.5.6 Vloženie používateľského príbehu do nástroja Redmine 12-45
12.6 Zmena funkcionálnych upresnení alebo nefunkcionálnych požiadaviek 12-48
12.7 Metodika manažmentu úloh 12-49
12.7.1 Úvod 12-49
12.7.2 Slovník pojmov 12-49
12.7.3 Úloha 12-49
12.7.4 Roly a zodpovednosť 12-51
12.7.5 Procesy 12-52
12.7.6 Aplikovanie metodiky v systéme Redmine 12-57
12.8 Manažment verzií a odovzdanie verzie softvérového artefaktu v nástroji *Visual Studio* do verziovacieho systému platformy *Team Foundation Server* 12-60
12.8.1 Úvod 12-60
12.8.2 Pojmy 12-60
12.8.3 Roly, zodpovednosti a procesy 12-60
12.8.4 Procesy na vyššej úrovni 12-60
12.8.5 Odovzdanie verzie softvérového artefaktu v nástroji *Visual Studio* do verziovacieho systému platformy *Team Foundation Server* 12-64
12.10 Metodika pre písanie zdrojových kódov v C# 12-67
12.10.1 Konvencia pre pomenovávanie namespacov, dátových typov, metód a fieldov 12-67
12.11 Konvencie lexikálneho štruktúrovania a úpravy zdrojového kódu 12-67
12.11.1 Konvencia komentovania zdrojových kódov 12-68
12.11.2 Konvencia štruktúrovania zdrojových kódov 12-68
12.11.3 Poznámka 12-68

1 ÚVOD

Účelom tohto dokumentu je zdokumentovanie riadenia projektu Odhaľovanie emocionálneho stavu používateľa. Dokument obsahuje vymedzené postupy a úlohy pre jednotlivých členov tímu, ktoré je potrebné vykonávať a dodržovať, aby na konci vznikol funkčný produkt podľa požiadaviek vedúceho projektu doc. Mgr. Daniela Chudá, PhD. Projekt je riešený v rámci predmetu Tímový projekt I a Tímový projekt II, ktoré absolvujeme v akademickom roku 2012/2013 v inžinierskom štúdiu.

2 PREDSTAVENIE ČLENOV TÍMU

Bc. Jozej Gajdoš - Softvérové inžinierstvo
Ukončené vzdelanie: 1. stupeň vysokoškolského vzdelania na STU FIIT, odbor INFO
Predmety inžinierskeho štúdia: OOANS, AASS
Pracovné skúsenosti: Vývoj webových aplikácií (sociálna sieť)
Zručnosti: C#, PHP, C++, Java, Haskell, Lua, Javascript, UML, MySQL, Cassandra, Qt
Iné: návrh architektúry aplikácií, algoritmy na spracovávanie textov, evolučné algoritmy

Bc. Martin Geier - Softvérové inžinierstvo
Ukončené vzdelanie: 1. stupeň vysokoškolského vzdelania na STU FIIT, odbor INFO
Predmety inžinierskeho štúdia: OOANS, DPRS, AASS
Zručnosti: C++, Java, UML, SQL, Gtkmm
Iné: funkcionálne a logické programovanie

Bc. Peter Greguš - Softvérové inžinierstvo
Ukončené vzdelanie: 1. stupeň vysokoškolského vzdelania na STU FIIT, odbor INFO
Predmety inžinierskeho štúdia: OOANS, DPRS, OZNAL
Zručnosti: PHP, C, Java, UML, MySQL, XML
Iné: tvorba webových aplikácií, funkcionálne a logické programovanie

Bc. Miroslav Hudák - Softvérové inžinierstvo
Ukončené vzdelanie: 1. stupeň vysokoškolského vzdelania na STU FIIT, odbor INFO
Predmety inžinierskeho štúdia: OOANS, AASS, KSS, NGNSSP
Pracovné skúsenosti: Vývoj grafických komponentov pre Windows Mobile a Windows CE
Zručnosti: C#, C++, UML, Qt
Iné: Prepínanie a smerovanie v TCP/IP sieťach, konfigurácia sieťových prvkov, Kinect

Bc. Peter Sivák - Softvérové inžinierstvo
Ukončené vzdelanie: 1. stupeň vysokoškolského vzdelania na STU FIIT, odbor INFO
Predmety inžinierskeho štúdia: STROJUC, NS, EA
Pracovné skúsenosti: tvorba webu, vytváranie a zdokonaľovanie bankového systému
Zručnosti: C++, Java, Qt, UML, MySQL, Latex, GIT
Iné: tvorba desktopových aplikácií

Bc. Peter Šinský - Informačné systémy
Ukončené vzdelanie: 1. stupeň vysokoškolského vzdelania na STU FIIT, odbor INFO
Predmety inžinierskeho štúdia: PDT, OZNAL
Zručnosti: C#, C++, Java, UML, Qt, Qt Mobility, XML
Iné: tvorba softvérov pre platformu Symbian

3 MOTÍVÁCIA

Vo všeobecnosti sú city spojené s potrebami človeka ako takého. Únava, nechutenstvo, radosť a ostatné emócie ovplyvňujú produktivitu človeka a jeho postoj k práci.

Emočný stav a celková psychická nálada ovplyvňuje našu výkonnosť a aj faktory, ktoré je možné merať pomocou bežne dostupných prostriedkov počítača, ako klávesnica, myš a kamera. Človek, ktorý je v dobrej nálade má inú frekvenciu písania ako keď ju má zlú a je apatický. Únava a psychická vyčerpanosť ovplyvňuje aj žmurkanie, ale aj napríklad to, či oči sledujú pozíciu kurzora. Takéto meranie emočného stavu nijako neobťažuje používateľa na rozdiel od klasických biometrických údajov ako tepová frekvencia či EEG.

Sledovanie, vyhodnocovanie a zaznamenávanie citov, nám môže pomôcť pri mnohých veciach. Od výskumu použiteľnosti aplikácii, kde môžeme sledovať aký má daná aplikácia vplyv na používateľa. Vytvorenie spätnej väzby pre aplikácie a vývojové prostredia. Inou možnosťou je reportovanie spätnej väzby na prácu iných ľudí, napríklad frustráciu pri čítaní cudzích zdrojových kódov, prideľovanie krátkodobých úloh v práci na základe aktuálneho emočného stavu.

Dosť zaujímavé využitie vyhodnocovania emócii vidíme v hernom priemysle, kde by bolo možné pomocou neho, upravovať obtiažnosť či dej počítačovej hry.

Využitie sledovania emócii vidíme aj vo vytvorenie spätnej väzby používateľovi , napríklad zobrazovať upozornenia, že je čas spraviť si prestávku, zmeniť typ práce a podobne. Takisto môže byť zaujímavé sledovať vplyv iných faktorov na emócie človeka, napríklad počúvanie hudby počas práce.

Tento projekt nás zaujal, najmä vďaka možnostiam jeho využitia a aj tým, že osoba, ktorej zisťujeme emočný stav to nevníma, respektíve o tom nemusí vedieť.

4 KONCEPCIA RIEŠENIA

Prvým krokom pre zistenie emočného stavu používateľa je potrebné sledovať jeho správanie na počítači. Najjednoduchšie je sledovať dynamiku úderov na klávesnicu a jej zmeny v priebehu času, časové intervaly medzi stlačeniami klávesnice, aký čas je podržaný kláves a celková rýchlosť písania textu. Ďalším periférnym zariadením, ktorý sa ľahko sleduje je myš, kde sa dá sledovať plynulosť pohybov, ich rýchlosť a náhle zmeny smeru. Pri myši je situácia zložitejšia ako pri klávesnici, pretože treba sledovať aktuálnu pozíciu, stav tlačidiel a čas. Tu môžeme vyhodnocovať smer, rýchlosť pohybu kurzora, rýchlosť dvojkliku, relatívne zrýchlenie pohybu kurzora a iné. Pri využití kamery je nutné rozpoznať na snímanom obraze oči. A sledovať kam sa používateľ pozerá, zisťovať rozdiel medzi pozíciu na ktorú sa pozerá a pozíciu kurzora.

Model používateľa by mal reprezentovať akýsi priemer (normálne duševné rozpoloženie) jeho emočných stavov . Je v ňom potrebné uchovávať jeho normálnu a priemernú dynamiku stláčania kláves, ako sa mýli (stlačenia backspace), dynamiku myši a údaje zozbierané kamerou. Model by mal v sebe uchovávať aj zosumarizované údaje, ktoré by slúžili na rýchle porovnanie aktuálneho stavu používateľa a tým vyhodnotiť aktuálny emocionálny stav a k nemu priradiť akcie aplikácie.

Do modelu je nutné zaznamenať aj aké periférne zariadenia daný používateľ používa, pretože keď je navyknutý na klávesnicu QWERTY na QWERTZ bude mať viacej chýb a pravdepodobne aj pomalšie písanie. Obdobne pri inej váhe/typu myši.

Údaje na porovnanie z modelom sa musia zbierať istý čas, pomocou odchytávania udalostí zariadení a následne vyfiltrovať od artefaktov alebo informácii, ktoré sa pri porovnávaní z modelom nepoužijú a následne kategorizovať udalosti. Pravdepodobne bude nutné experimentálne overiť aký čas sa majú zhromažďovať potrebné údaje aby bolo možné relevantne vyhodnotiť aktuálny emočný stav.

Na samotné určenie emočného stavu používateľa by bolo možné použiť niektoré z ďalej uvedených postupov:

Naivný algoritmus – keď používateľ píše pomalšie a viac sa míli je unavený, ….

Neurónové siete – pre daného používateľa sa neurónová sieť naučí rozpoznať v akom emočnom stave sa nachádza

Skryté Markovove modeli – pomocou údajov ako dynamika stláčania textu dokážu identifikovať v akom emočnom stave sa nachádza používateľ

5 ZÁPISY ZO STRETNUTÍ

V tejto kapitole sa nachádzajú všetky zápisy zo stretnutia, ktoré boli zaznamenávané počas celého semestra pri každom oficiálnom stretnutí tímu. Zápisy majú formálny tvar a obsahujú informácie: dátum, čas, miesto stretnutia, prítomný členovia, vedúci stretnutia a zapisovateľ. Ďalej je v zápise opísaný priebeh stretnutia. Tieto informácie sú zapisované po bodoch. Medzi tieto informácie patrí: téma stretnutia, vyhodnotenie úloh z predchádzajúceho obdobia, záznam stretnutia, úlohy do nasledujúceho obdobia, poznámky a prílohy.

5.1 ZÁPIS Č.1

Dátum:	3.10.2012
Čas stretnutia:	14:00 – 15:15
Miesto stretnutia:	softvérové štúdio, FIIT STU
Prítomní:	
Pedagóg:	doc. Mgr. Daniela Chudá, PhD.
Členovia tímu:	Bc. Jozef Gajdoš Bc. Martin Geier Bc. Miroslav Hudák Bc. Peter Sivák Bc. Peter Šinský
Vedúci stretnutia:	doc. Mgr. Daniela Chudá, PhD.
Zapisovateľ:	Bc. Peter Sivák

TÉMA STRETNUTIA

Úvodné priblíženie predmetu

VYHODNOTENIE ÚLOH Z PREDCHÁDZAJÚCEHO OBDOBIA

-

ZÁZNAM STRETNUTIA

- Bude sa pracovať metodológiou Scrum (2-týždňové intervaly).
- Kartičkami sa budú ohodnocovať úlohy.
- Jediný spôsob, ako rozpoznať emocionálny stav používateľa, je zo vstupných dát.
- Fázy vývoja
    - Modelovanie
    - Rozpoznávanie podľa modelov
- Do Novembra podať prihlášku na TP Cup:
    - Poster
    - 2-stranový abstrakt v Apríli – predstavenie na IITSRC
- Dostaneme dátový priestor, kde umiestnime webovú stránku tímu.
- Podporné prostriedky vývoja:
    - Manažovací nástroj – Redmine

    - Verziovací systém – http://gitbus.fiit.stuba.sk/
- Každý člen tímu dostane vlastnú známku za jeho prácu.
- Zistiť bližšie informácie o Keystroke Dynamics.

ÚLOHY DO NASLEDUJÚCEHO OBDOBIA

ID	Popis úlohy	Zodpovedná osoba	Termín	Stav	Hotovo [%]
	Vytvorenie šablóny zápisov	Hudák	10.10.2012		

POZNÁMKY

-

PRÍLOHY

-

doc. Mgr. Daniela Chudá, PhD.	Bc. Peter Sivák
vedúci stretnutia	zapisovateľ

5.2 ZÁPIS Č.2

Dátum:	10.10.2012
Čas stretnutia:	14:00 – 15:20
Miesto stretnutia:	softvérové štúdio, FIIT STU
Prítomní:	
Pedagóg:	doc. Mgr. Daniela Chudá, PhD.
Členovia tímu:	Bc. Jozef Gajdoš Bc. Martin Geier Bc. Peter Greguš Bc. Miroslav Hudák Bc. Peter Sivák
Vedúci stretnutia:	doc. Mgr. Daniela Chudá, PhD.
Zapisovateľ:	Bc. Peter Sivák

TÉMA STRETNUTIA

Analýza problému, hrubý návrh riešenia

VYHODNOTENIE ÚLOH Z PREDCHÁDZAJÚCEHO OBDOBIA

ID	Popis úlohy	Zodpovedná osoba	Termín	Stav	Hotovo [%]
	Vytvorenie šablóny zápisov	Hudák	10.10.2012	Hotovo	100

ZÁZNAM STRETNUTIA

- Spomínalo sa využitie trigramov v projekte Perkonik, čo by sme mohli využiť aj v našom projekte.
- Prvá časť aplikácie nášho projektu je logovanie, ktorá sa skladá z nasledovných vecí krokov:
    - Značkovanie emocionálneho stavu.
    - Získanie modelu používateľa (môže to byť vektor čísel).
    - Modelov je zrazu veľa – môj, kamarátov, od iných ľudí – následne systém používateľovi odporučí nejakú akciu a pokračuje sa odznova.

- Logger loguje klávesnicu, myš, webstránky (je to dotnetová aplikácia bežiaca na pozadí).
- Vybrali sme si rozšírenie existujúceho riešenia (aplikácie) oproti programovaniu úplne novej aplikácie od začiatku.
- Spomínalo sa využitie manažovacieho systému Team Foundation Server namiesto verziovacieho systému Git.
- Vybrali sme si manažovací systém Redmine.
- Peter Sivák bude komunikovať s Karolom Rástočným ohľadne výberu technológií.
- Možnosti modelu aplikácie
    - Gratexlog + thin log + vlastné modifikácie
    - Gratexlog + všetko (vybrali sme si túto možnosť)
- Každý, kto bude niečo analyzovať, to aj odprezentuje ostatným členom tímu, aby z toho aj tí niečo mali.
- Product Backlog (časť)
    - Ja ako používateľ som sledovaný.
        - Ja ako používateľ som sledovaný kamerou.
        - Ja ako používateľ som sledovaný klávesnicou a myšou.
    - Ja ako používateľ som modelovaný (môj profil).
    - Modely sú rozpoznávané.
    - Ja ako používateľ dostanem odporúčanie.
- V redmine budú tri roly – manager, developer a reporter.

ÚLOHY DO NASLEDUJÚCEHO OBDOBIA

ID	Popis úlohy	Zodpovedná osoba	Termín	Stav	Hotovo [%]
	Rozbehať Redmine – vytvoriť projekt, nahádzať ľudí, pridať user stories.	Sivák	17.10.2012		
	Urobiť webstránku tímu.	Gajdoš	17.10.2012		
	Analyzovať existujúce logovacie softvéry.	Geier	17.10.2012		
	Aké odporúčania sú vhodné na ponúknutie používateľovi?	Greguš	17.10.2012		
	Rozbehať logger.	Hudák	17.10.2012		
	Techniky logovania, aké rôzne veci sa môžu logovať.	Gajdoš	17.10.2012		

	Prejsť články na emocionálne stavy (8 stavov a menej) a zistiť čo je to model používateľa.	Šinský	17.10.2012		
	(Rozpoznávanie) analýza modelov	Sivák	17.10.2012		

POZNÁMKY

-

PRÍLOHY

-

doc. Mgr. Daniela Chudá, PhD.	Bc. Peter Sivák
vedúci stretnutia	zapisovateľ

5.3 ZÁPIS Č.4

Dátum:	17.10.2012
Čas stretnutia:	14:00 – 16:50
Miesto stretnutia:	softvérové štúdio, FIIT STU
Prítomní:	
Pedagóg:	doc. Mgr. Daniela Chudá, PhD.
Členovia tímu:	Bc. Jozef Gajdoš Bc. Martin Geier Bc. Peter Greguš Bc. Miroslav Hudák Bc. Peter Sivák Bc. Peter Šinský
Vedúci stretnutia:	Bc. Peter Sivák
Zapisovateľ:	Bc. Peter Greguš

TÉMA STRETNUTIA

Začatie prvého šprintu, dopĺňanie backlogu.

VYHODNOTENIE ÚLOH Z PREDCHÁDZAJÚCEHO OBDOBIA

ID	Popis úlohy	Zodpovedná osoba	Termín	Stav	Hotovo [%]
	Rozbehať Redmine – vytvoriť projekt, nahádzať ľudí, pridať user stories.	Sivák	17.10.2012	Hotovo	100%
	Urobiť webstránku tímu.	Gajdoš	17.10.2012	Hotovo	100%
	Analyzovať existujúce logovacie softvéry.	Geier	17.10.2012	Hotovo	100%
	Aké odporúčania sú vhodné na ponúknutie používateľovi?	Greguš	17.10.2012	Hotovo	100%
	Rozbehať logger.	Hudák	17.10.2012	Rieši sa	60%
	Techniky logovania, aké rôzne veci sa môžu logovať.	Gajdoš	17.10.2012	Hotovo	100%
	Prejsť články na emocionálne stavy (8 stavov a menej) a zistiť čo je to model používateľa.	Šinský	17.10.2012	Hotovo	100%

	(Rozpoznávanie) analýza modelov	Sivák	17.10.2012	Hotovo	100%

ZÁZNAM STRETNUTIA

- Popis existujúcich riešení loggerov (možnosť využiť *KidLogger*, OsdHotkey, PyKeyLogger – open source)
- Čo logovať:
  - klávesnica: doba stlačenia klávesy (únava), rýchlosť písania, frekvencia chýb, frekvencie vybraných trigramov, backspace (chyby)
  - myš: rýchlosť klikania, frekvencia klikov, zrýchlenie kurzora, smer, doubleclick
- redmine: vytvorený projekt, pridaný členovia tímu, pridané user-stories
- backlog:
  - **ja ako používateľ som sledovaný**
    - Sledovanie klávesnice s myšou
    - Odosielanie údajov na server + uloženie
    - Zadanie emočného stavu
  - **ja ako používateľ som modelovaný**
    - CRUD modelu
    - Prispôsobovanie modelu
  - **modely sú rozpoznávané**
    - Porovnanie aktuálneho stavu s modelom
    - Vybranie modelu používateľa
    - Detekcia aktuálnej emócie + uloženie
    - narušiteľ
  - **ja ako používateľ dostanem odporúčanie**
    - Výber odporúčania na základe emočného stavu
    - Zobrazenie odporúčania
- Model: ID + model emócií
- Základné emócie: strach, hnev, znechutenie, pohŕdanie, radosť, smútok, prekvapenie
- Odporúčanie: fyzická relaxácia, hra, socializovanie, jedlo, šport, hnev-prestávka, káva, farby backgroundu, odporúčanie dovolenky, zmeniť úlohu
- Model: dáta charakterizujúce používateľa
- Rozpoznávanie: rozhodovacie stromy, neurónové siete – výber metódy na základe modelu, nazbieraných dát
- Zvolené emócie (predbežne): radosť/nadšenie, neutrálny, stres, únava, hnev

- Model: trigramy, keystroke/minute, chyby (del,bsp)/minute
- Preposielať na perconik

ÚLOHY DO NASLEDUJÚCEHO OBDOBIA

ID	Popis úlohy	Zodpovedná osoba	Termín	Stav	Hotovo [%]
	Zistiť čo trackuje perkonik	Hudák	24.10.2012		
	Fotky členov tímu, odkazy na podporné nástroje na stránku	Gajdoš	24.10.2012		
	Pridať prezentácie na redmine	Sivák	24.10.2012		
	Zistiť ako hádzať logy na server	Hudák	24.10.2012		
	Nainštalovať logger	všetci	24.10.2012		
	Analýza datbázy	Gajdoš	24.10.2012		
	Analýza stesu + čo odporučiť	Geier	24.10.2012		
	Analýza hnevu + čo odporučiť	Šinský	24.10.2012		
	Analýza únava + čo odporučiť	Šinský	24.10.2012		
	Analýza radosť + čo odporučiť	Greguš	24.10.2012		
	Rozbehať win server + web	Sivák	24.10.2012		

POZNÁMKY

-

PRÍLOHY

-

Bc. Peter Sivák vedúci stretnutia	Bc. Peter Greguš zapisovateľ

5.4 ZÁPIS Č.4

Dátum:	24.10.2012
Čas stretnutia:	14:00 – 15:30
Miesto stretnutia:	softvérové štúdio, FIIT STU
Prítomní:	
Pedagóg:	doc. Mgr. Daniela Chudá, PhD.
Členovia tímu:	Bc. Jozef Gajdoš Bc. Martin Geier Bc. Peter Greguš Bc. Miroslav Hudák Bc. Peter Sivák Bc. Peter Šinský
Vedúci stretnutia:	Bc. Peter Greguš
Zapisovateľ:	Bc. Jozef Gajdoš

TÉMA STRETNUTIA

Pokračovanie prvého šprintu.

VYHODNOTENIE ÚLOH Z PREDCHÁDZAJÚCEHO OBDOBIA

ID	Popis úlohy	Zodpovedná osoba	Termín	Stav	Hotovo [%]
	Zistiť čo trackuje perkonik	Hudák	24.10.2012	vyriešená	100
	Fotky členov tímu, odkazy na podporné nástroje na stránku	Gajdoš	24.10.2012		90
	Pridať prezentácie na redmine	Sivák	24.10.2012	vyriešené	100
	Zistiť ako hádzať logy na server	Hudák	24.10.2012		
	Nainštalovať logger	všetci	24.10.2012		83
	Analýza databázy	Gajdoš	24.10.2012	vyriešené	100
	Analýza stresu + čo odporučiť	Geier	24.10.2012	vyriešené	100
	Analýza hnevu + čo odporučiť	Šinský	24.10.2012	vyriešené	100
	Analýza únavy + čo	Šinský	24.10.2012	vyriešené	100

	odporučiť				
	Analýza radosti + čo odporučiť	Greguš	24.10.2012	vyriešené	100
	Rozbehať win server + web	Sivák	24.10.2012	vyriešené	100

ZÁZNAM STRETNUTIA

- Prezentácia loggera PerConIk (M. Hudák).
  - Nevieme čo je milestone.
- Číslovanie dokumentácie (rímska číslica – a,b,c,d ,...)(analýza, jej kapitoly) a treba robiť dokumentáciu.
- Hrubý plán, nakreslili sme si 4 časti celého softvéru, plán je v ktorom šprinte sa bude čo rozpracovávať.
  - zima – implementácia
  - leto – testovanie, vylepšovanie, malé zlepšenia
- Prezentácia dátového modelu PerConIk-a (J. Gajdoš).
- Diskusia k termínom odovzdávania a webu.
- Prezentácia analýz emócií :
  - Únava a hnev (P. Šinský)
    - Treba sa opýtať psychologičky, či nám nevie odporučiť literatúru.
  - Stres (M. Geier)
  - Radosť (P. Greguš)
- Čo budeme kódiť do lokálneho PerConIk-a
  - Zdávanie emócie (GUI, odosielanie,)
  - Pridanie milestone – zmena emócie (v GUI).
- Čo treba dokódiť na serveri:
  - Do DB pridať:
    - Entita State: Zadaná emócia
    - Entita State: Detegovaná emócia
- Vysvetľovanie aký je rozdiel medzi SqLite a MS Sql.

ÚLOHY DO NASLEDUJÚCEHO OBDOBIA

ID	Popis úlohy	Zodpovedná osoba	Termín	Stav	Hotovo [%]
	Zistiť čo je to milestone. A ako pridať vlastné.	Hudák			
	Nainštalovať Windows 7.	Gajdoš			
	Zistiť ako sa vytvára XML s dátového objektu, zistiť ako sa používateľ autorizuje.	Greguš			

	Zistiť ako sa z XML vytvára dátový objekt na serveri.	Greguš			
	Dokumentácia	Všetci			
	Pridať zadanú emóciu do XML.	Greguš			
	Pridať zadanú emóciu do SqLite.	Geier			
	Spísať konvencie ohľadom zdrojového kódu.	Gajdoš			
	Zistiť ako sa vkladá dátový objekt do MS Sql.	Geier			
	Zistiť ako získať dátový objekt s SqLite.	Geier			
	Vypracovať metodiku pre spôsob komitovania.	Sivák			
	Rozbehať FS. A nahodiť PerConIk.	Sivák			
	Vypracovať metodiku písania zdrojového kódu.	Gajdoš			
	Zlúčenie dokumentácie.	Šinský			
	Vytvoriť šablónu dokumentácie.	Šinský			

POZNÁMKY

- Ďalší šprint bude asi model používateľa.

PRÍLOHY

-

Bc. Peter Greguš	Bc. Jozef Gajdoš
vedúci stretnutia	zapisovateľ

5.5 Zápis č.5

Dátum:	31.10.2012
Čas stretnutia:	14:00 – 16:50
Miesto stretnutia:	softvérové štúdio, FIIT STU
Prítomní:	
Pedagóg:	doc. Mgr. Daniela Chudá, PhD.
Členovia tímu:	Bc. Jozef Gajdoš Bc. Martin Geier Bc. Peter Greguš Bc. Miroslav Hudák Bc. Peter Sivák Bc. Peter Šinský
Vedúci stretnutia:	Bc. Jozef Gajdoš
Zapisovateľ:	Bc. Peter Šinský

Téma stretnutia

Začatie druhého šprintu

Vyhodnotenie úloh z predchádzajúceho obdobia

ID	Popis úlohy	Zodpovedná osoba	SP	Termín	Hotovo [%]
4201	Zistiť, čo je milestone a ako pridať vlastné	Hudák		31.10.2012	100
	Nainštalovať Windows 7	Gajdoš		31.10.2012	100
4203	Zistiť, ako sa vytvára XML z dátového objektu a zistiť, ako sa používateľ autorizuje	Greguš		31.10.2012	100
4204	Zistiť, ako sa z XML vytvára dátový objekt na serveri	Greguš		31.10.2012	100
4205 4206 4207 4208 4209 4210	Dokumentácia	všetci		31.10.2012	100
4212	Pridať zadanú emóciu do XML	Greguš		31.10.2012	100
4214	Pridať zadanú emóciu do SQLite	Geier		31.10.2012	100

4215	Spísať konvencie ohľadom zdrojového kódu	Gajdoš		31.10.2012	100
4216	Zistiť, ako sa vkladá dátový objekt do MS SQL	Geier		31.10.2012	100
4217	Zistiť, ako získať dátový objekt z SQLite	Geier		31.10.2012	100
4218	Vypracovať metodiku pre spôsob komitovania.	Sivák		31.10.2012	20
4219	Rozbehať TFS a nahodiť PerConIK	Sivák		31.10.2012	100
4389	Vypracovať metodiku písania zdrojového kódu	Hudák		31.10.2012	100
4221	Zlúčenie dokumentácie	Šinský		31.10.2012	100
4222	Vytvoriť šablónu dokumentácie	Šinský		31.10.2012	100

ZÁZNAM STRETNUTIA

- Riešenie úloh, ktoré už mali byť.
- Vysvetľovanie fungovania ukladania dát na lokáli a serveri (Geier).
    - Objasnenie potreby riešenia zodpovedajúcich úloh.
- Vysvetlenie, čo je milestone.
- Pravdepodobne bude klient zisťovať vypočítanú emóciu pomocou vzdialeného volania metódy.
- Riešenie pridania emócie k logu.
- Ako sa bude počítať emócia z nalogovaných dát.
- Ukladanie emócií do databázy, treba trackovať viacej emócií, aby sme vedeli, ktoré sa dajú rozpoznať.
    - Návrh na viac emócií, aspoň 8.
- Návrh modelu používateľa:
    - Metriky pre spracovanie logov na klávesnici.
        - Priemerné stlačenie kláves.
        - Počet del a backspace/čas a počet kláves.
        - Počet kláves/čas.
        - Priemerné časy bi a trigramov.
    - Metriky pre spracovanie logov na myši.
        - Priemerné zrýchlenie.
        - Priemerné klikanie všetkých troch tlačidiel.
        - Skrolovanie, rýchlosť a vzdialenosť.
        - Prejdená vzdialenosť s myšou.
        - Počet zmien smeru.
    - Emotion vector
        - Id:int
        - User_id:int

- Emotion:int
- Data:blob

ÚLOHY DO NASLEDUJÚCEHO OBDOBIA

ID	Popis úlohy	Zodpovedná osoba	SP	Termín	Hotovo [%]
4414 4415 4416 4417 4418 4419	Nainštalovať Visual Studio SDK (na lokáli)	všetci	1	7.11.2012	0
4420	Vytvoriť účty na Windows Serveri a nastaviť TFS	Sivák	3	7.11.2012	0
4421	Dorobiť zisťovanie nálady pri zapnutí	Gajdoš	1	7.11.2012	0
4422	Upraviť dump DB pridaním stĺpca (Zadaná emócia, Rozpoznaná emócia)	Gajdoš	5	7.11.2012	0
4423	Vytvorenie DB schémy na serveri a spustiť skript	Šinský	2	7.11.2012	0
4424	Opraviť autoload watchera (zdrojáky)	Gajdoš	8	7.11.2012	0
4425	Pridať konvertor pre emotion state DTO	Geier	13	7.11.2012	0
4426	Zistiť prístupné porty a pozrieť dokumentáciu k virtuálnemu serveru	Geier	5	7.11.2012	0
4427	Pridať milestone zmena emočného stavu	Hudák	5	7.11.2012	0
4428	Pridať zvyšných členov do email forwardingu	Šinský	1	7.11.2012	0
4429	Analýza modelu emócie a ako urobiť model z logov klávesnice a myši	Greguš	20	7.11.2012	0
4430	Analýza klávesnice	Sivák	13	7.11.2012	0
4431	Analýza myši	Šinský	13	7.11.2012	0
4432	Vytvorenie tabuľky v DB (emotion vector)	Gajdoš	5	7.11.2012	0
4433	Vytvoriť ORM k emotion vector	Hudák	20	7.11.2012	0
4434	Vybaviť knihu	Šinský	5	7.11.2012	0
4435	Vytvoriť initial commit	Geier	5	7.11.2012	0
4218	Vypracovať metodiku pre spôsob komitovania.	Sivák	5	7.11.2012	20

POZNÁMKY

-

PRÍLOHY

-

Bc. Jozef Gajdoš vedúci stretnutia	Bc. Peter Šinský zapisovateľ

5.6 ZÁPIS Č.6

Dátum:	7.11.2012
Čas stretnutia:	14:00 – 16:40
Miesto stretnutia:	softvérové štúdio, FIIT STU
Prítomní:	
Pedagóg:	doc. Mgr. Daniela Chudá, PhD.
Členovia tímu:	Bc. Jozef Gajdoš Bc. Martin Geier Bc. Peter Greguš Bc. Miroslav Hudák Bc. Peter Sivák Bc. Peter Šinský
Vedúci stretnutia:	Bc. Peter Šinský
Zapisovateľ:	Bc. Martin Geier

TÉMA STRETNUTIA

Druhé stretnutie ku šprintu Henrieta

VYHODNOTENIE ÚLOH Z PREDCHÁDZAJÚCEHO OBDOBIA

ID	Popis úlohy	Zodpovedná osoba	SP	Termín	Hotovo [%]
4414 **4415** **4416** **4417** **4418** **4419**	Nainštalovať Visual Studio SDK (na lokáli)	všetci	1	7.11.2012	84
4420	Vytvoriť účty na Windows Serveri a nastaviť TFS	Sivák	3	7.11.2012	100
4421	Dorobiť zisťovanie nálady pri zapnutí	Gajdoš	1	7.11.2012	100
4422	Upraviť dump DB pridaním stĺpca (Zadaná emócia, Rozpoznaná emócia)	Gajdoš	5	7.11.2012	100
4423	Vytvorenie DB schémy na serveri a spustiť skript	Šinský	2	7.11.2012	100
4424	Opraviť autoload watchera	Gajdoš	8	7.11.2012	100

	(zdrojáky)				
4425	Pridať konvertor pre emotion state DTO	Geier	13	7.11.2012	100
4426	Zistiť prístupné porty a pozrieť dokumentáciu k virtuálnemu serveru	Geier	5	7.11.2012	100
4427	Pridať milestone zmena emočného stavu	Hudák	5	7.11.2012	80
4428	Pridať zvyšných členov do email forwardingu	Šinský	1	7.11.2012	100
4429	Analýza modelu emócie a ako urobiť model z logov klávesnice a myši	Greguš	20	7.11.2012	50
4430	Analýza klávesnice	Sivák	13	7.11.2012	0
4431	Analýza myši	Šinský	13	7.11.2012	0
4432	Vytvorenie tabuľky v DB (emotion vector)	Gajdoš	5	7.11.2012	100
4433	Vytvoriť ORM k emotion vector	Hudák	20	7.11.2012	100
4434	Vybaviť knihu	Šinský	5	7.11.2012	0
4435	Vytvoriť initial commit	Geier	5	7.11.2012	100
4218	Vypracovať metodiku pre spôsob komitovania.	Sivák	5	31.10.2012	100

ZÁZNAM STRETNUTIA

- Diskusia s Chudou k priebehu práce:
  - Rozbíjanie veľkých taskov.
  - Kontrola projektového denníka – či si vedieme zápisky.
  - Zjemňovanie plánovania počas šprintov.
  - Diskusia o rizikách počas stretnutí (najmenej 2x za semester).
  - Dokumentácia – na konci sumarizačná kapitola:
    - Zima – sumarizácia prototypu.
    - Leto – sumarizácia produktu.
  - Problémy s nevyriešením úlohy – ak nevie riešiť tím – obrátiť sa na učiteľa.
  - Debata o komunikácii – pridanie chudej do mailu.
  - Písanie automatizovaných testov.
  - Diskusia ohľadne TP-Cupu:
    - Žiadosť o účasť.
    - Čo nás to bude stáť – plagát, dokument (správa).
    - Čo nám to prinesie – nové vedomosti, prezentácia.
  - Prezentácia výsledku práce:

        - Manažérska prezentácia – rozdávanie lístočkov, zaujať ľudí.
        - Grafická prezentácia.
    - Grafická prezentácia nášho projektu:
        - Grafické znázornenie „user stories“ – tabuľa.
        - Priebeh fungovania programu.
- Návod inštalácie lokálneho servera – Greguš.
- Implementácia milestone – Hudák.
- Rozbíjanie features na tasky.
- Transformácia prichádzajúcich dát na emotion vector:
    - vložiť obrázok na model používateľa .
- Typy na metriky:
    - Scenár určitej aplikácie – zapnem mail, zapnem sme.sk, atď.
    - Aké aplikácie sú spustené.
    - Aké klávesové skratky kto používa.
- Uchovávanie Backlog-u v Exceli.
- Odporúčanie od vedúcej – ručná analýza dát.
- Nové zadelenie rolí:
    - Gajdoš – vedúci vývoja.
    - Greguš – manažér technickej dokumentácie.
    - Šinský – manažér dokumentácie riadenia.

Návrh úloh do Backlog-u.

Používateľ je sledovaný:

- Sledovanie myši.
- Sledovanie klávesnice.
- Sledovanie aplikácie.

Používateľ je rozpoznávaný:

- Neurónová sieť.
- Komparátor.
- Pravdepodobnostné funkcie.

Odporúčania:

- Aplikácie.
- Hudba.
- Jedlo.

Úlohy do nasledujúceho obdobia

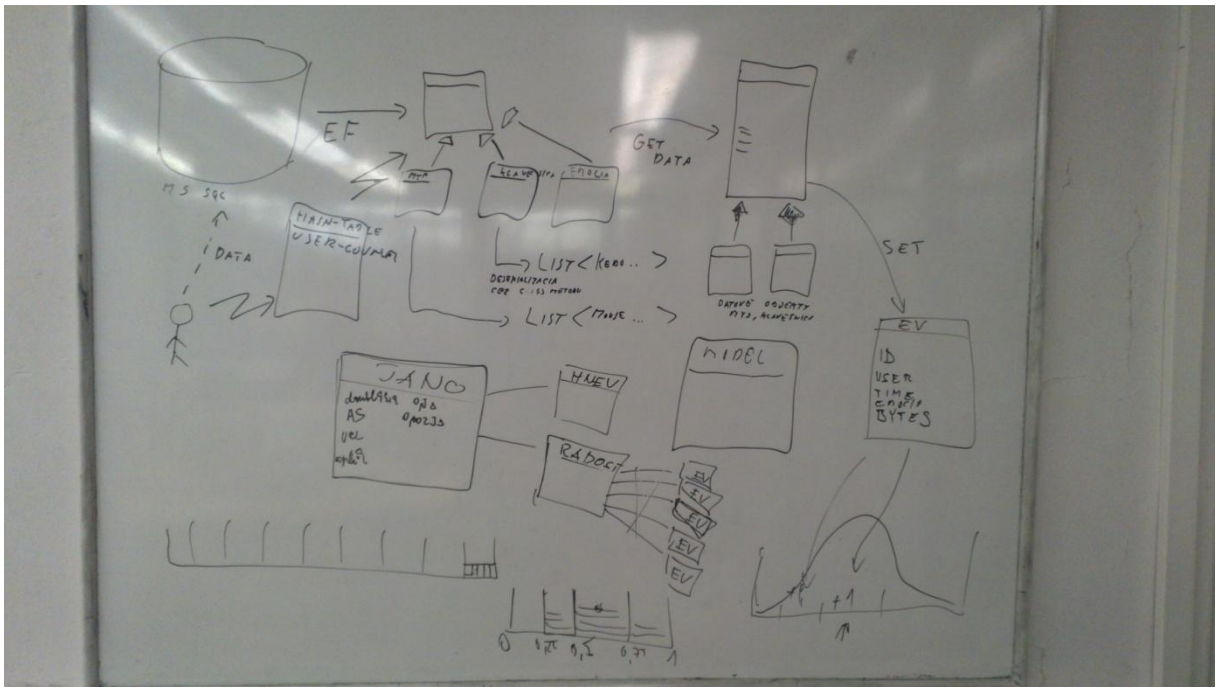
ID	Popis úlohy	Zodpovedná osoba	SP	Termín	Hotovo [%]
4561	Analyzovať mouse data blob	Gajdoš		14.11.2012	0
4562	Analyzovať keyboard data blob	Geier		14.11.2012	0
4563	Aktualizovať šablónu zápisnice	Sivák		14.11.2012	0
4564 **4565** **4566** **4567** **4568** **4569**	Napísať svoju časť dokumentácie z druhého šprintu	všetci		14.11.2012	0
4570	Zlúčiť dokumentácie riadenia	Šinský		14.11.2012	0
4571	Vytlačiť a zviazať dokumentácie	Hudák		14.11.2012	0
4572	Zlúčiť technické dokumentácie	Greguš		14.11.2012	0
4573	Spísať backlog	Sivák		14.11.2012	0
4574	Pridať description k taskom	Sivák		14.11.2012	0
4575	Urobiť export do csv	Gajdoš		14.11.2012	0
4576	Spísať konvencie do metodiky	Hudák		14.11.2012	0

Poznámky

Peter Sivák bude udržiavať Backlog a Releaselog.

Zvážiť analýzu metrík aplikácií.

Prílohy



OBRÁZOK 1 NÁVRH UKLADANIA VÝSLEDKOV METRÍK

Bc. Peter Šinský vedúci stretnutia	Bc. Martin Geier zapisovateľ

6 MANAŽMENT ROZVRHU A PLÁNOVANIA

Kapitola sa venuje procesu vytvárania plánu pre náš projekt. Proces vytvárania plánu sa vykonáva priebežne počas vývoja projektu. Na začiatku je vytvorený hrubý plán, ktorý sa však postupne zjemňuje. Zjemňovanie hrubého plánu sa vykonáva na začiatku každého šprintu. Rovnako na začiatku šprintu je vytvorený podrobný plán, ktorý odzrkadľuje prácu, ktorú je potrebné vykonať počas jedného šprintu.

6.1 PLÁN PROJEKTU

Na začiatku projektu vznikol hrubý plán, ktorý odzrkadľoval prácu potrebnú vykonať v zimnom semestri. Hrubý plán je rozdelený do štyroch dvojtýždňových šprintov. Termíny šprintov boli definované vopred. Ako bolo vyššie spomenuté, plán bol postupne zjemňovaný. Popis plánovania šprintu je podrobnejšie opísaný v metodike plánovania šprintu 12.4.

Šprint	Týždeň	Úloha
Šprint 0	3 – 4 týždeň	• Tvorba webovej prezentácie • Analýza zdrojov • Analýza logovača PerConIK

Šprint 1	4 – 6 týždeň	• Vyriešenie logovania používateľa
		  ○ Analyza logovania dat v PerConIK-ovi
		  ○ Analyza posielania dat na lokal a server

Šprint 2	6 – 8 týždeň	• Vytvorenie modelu používateľa • Analýza a definícia metrík • zachytávanie logov • Analýza zachytených dát
Šprint 3	8 – 10 týždeň	• Rozpoznanie modelu používateľa • Analýza dostupných metód pre rozpoznávanie (neurónové siete, komparátor ...) • Návrh metódy • Implementácia navrhnutej metódy
Šprint 4	10 – 12 týždeň	Poskytnutie odporúčania na základe emočného stavu

6.1.1 ŠPRINT 0

Podrobný plán úloh pred začatím šprintov.

Popis úlohy	Vypracováva	Termín
Rozbehať Redmine – vytvoriť projekt, nahádzať ľudí, pridať user stories	Peter Sivák	17.10.2012
Urobiť webstránku tímu	Jozef Gajdoš	17.10.2012
Analyzovať existujúce logovacie softvéry	Martin Geier	17.10.2012
Aké odporúčania sú vhodné na ponúknutie používateľovi?	Peter Greguš	17.10.2012
Rozbehať logger	Miroslav Hudák	17.10.2012

Techniky logovania, aké rôzne veci sa môžu logovať.	Jozef Gajdoš	17.10.2012
Prejsť články na emocionálne stavy (8 stavov a menej) a zistiť čo je model používateľa.	Peter Šinský	17.10.2012
(Rozpoznávanie) analýza modelov	Peter Sivák	17.10.2012

6.1.2 Šprint 1

Podrobný plán úloh pre šprint 1.

Popis úlohy	Vypracováva	Termín
Zistiť čo trackuje perkonik	Miroslav Hudák	24.10.2012
Fotky členov tímu, odkazy na podporné nástroje na stránku	Jozef Gajdoš	24.10.2012
Pridať prezentácie na redmine	Peter Sivák	24.10.2012
Zistiť ako hádzať logy na server	Miroslav hudák	24.10.2012
Nainštalovať logger	Všetci	24.10.2012
Analýza databázy	Jozef Gajdoš	24.10.2012
Analýza stresu + odporúčania	Martin Geier	24.10.2012
Analýza hnevu + odporúčania	Peter Šinský	24.10.2012
Analýza únavy + odporúčania	Peter Šinský	24.10.2012
Analýza radosť + odporúčania	Peter Greguš	24.10.2012
Rozbehať win server + web	Peter Sivák	24.10.2012
Zistiť, čo je milestone a ako pridať vlastné	Miroslav Hudák	24.10.2012
Nainštalovať Windows 7	Jozef Gajdoš	24.10.2012
Zistiť, ako sa vytvára XML z dátového objektu a zistiť, ako sa používateľ autorizuje	Peter Greguš	24.10.2012
Zistiť, ako sa z XML vytvára dátový objekt na serveri	Peter Greguš	24.10.2012
Dokumentácia	všetci	24.10.2012
Pridať zadanú emóciu do XML	Peter Greguš	24.10.2012
Pridať zadanú emóciu do SQLite	Martin Geier	24.10.2012
Spísať konvencie ohľadom zdrojového kódu	Jozef Gajdoš	24.10.2012
Zistiť, ako sa vkladá dátový objekt do MS SQL	Martin Geier	24.10.2012
Zistiť, ako získať dátový objekt z SQLite	Marint Geier	24.10.2012
Vypracovať metodiku pre spôsob komitovania.	Peter Sivák	24.10.2012
Rozbehať TFS a nahodiť PerConIK	Peter Sivák	24.10.2012
Vypracovať metodiku písania zdrojového kódu	Miroslav Hudák	24.10.2012
Zlúčenie dokumentácie	Peter Šinský	24.10.2012
Vytvoriť šablónu dokumentácie	Peter Šinský	24.10.2012

6.1.3 Šprint 2

Podrobný plán úloh pre šprint 2.

Popis úlohy	Vypracováva	Termín
Nainštalovať Visual Studio SDK (na lokáli)	všetci	31.10.2012
Vytvoriť účty na Windows Serveri a nastaviť TFS	Peter Sivák	31.10.2012
Dorobiť zisťovanie nálady pri zapnutí	Jozef Gajdoš	31.10.2012
Upraviť dump DB pridaním stĺpca (Zadaná emócia, Rozpoznaná emócia)	Jozef Gajdoš	31.10.2012
Vytvorenie DB schémy na serveri a spustiť skript	Peter Šinský	31.10.2012
Opraviť autoload watchera (zdrojáky)	Jozef Gajdoš	31.10.2012
Pridať konvertor pre emotion state DTO	Martin Geier	31.10.2012
Zistiť prístupné porty a pozrieť dokumentáciu k virtuálnemu serveru	Martin Geier	31.10.2012
Pridať milestone zmena emočného stavu	Miroslav Hudák	31.10.2012
Pridať zvyšných členov do email forwardingu	Peter Šinský	31.10.2012
Analýza modelu emócie a ako urobiť model z logov klávesnice a myši	Peter Greguš	31.10.2012
Analýza klávesnice	Peter Sivák	31.10.2012
Analýza myši	Peter Šinský	31.10.2012
Vytvorenie tabuľky v DB (emotion vector)	Jozef Gajdoš	31.10.2012
Vytvoriť ORM k emotion vector	Miroslav Hudák	31.10.2012
Vybaviť knihu	Peter Šinský	31.10.2012
Vytvoriť initial commit	Martin Geier	31.10.2012
Vypracovať metodiku pre spôsob komitovania.	Peter Sivák	31.10.2012
Analyzovať mouse data blob	Jozef Gajdoš	31.10.2012
Analyzovať keyboard data blob	Martin Geier	31.10.2012
Aktualizovať šablónu zápisnice	Peter Sivák	31.10.2012
Napísať svoju časť dokumentácie z druhého šprintu	všetci	31.10.2012
Zlúčiť dokumentácie riadenia	Peter Šinský	31.10.2012
Vytlačiť a zviazať dokumentácie	Miroslav Hudák	31.10.2012
Zlúčiť technické dokumentácie	Peter Greguš	31.10.2012
Spísať backlog	Peter Sivák	31.10.2012
Pridať description k taskom	Peter Sivák	31.10.2012
Urobiť export do csv	Jozef Gajdoš	31.10.2012
Spísať konvencie do metodiky	Miroslav Hudák	31.10.2012

7 MANAŽMENT RIZÍK

Manažment rizík je v tímovom projekte veľmi dôležitý. Samotný proces manažmentu rizík vo všeobecnosti pozostáva s týchto základných činností:

- identifikácia,
- analýza,
- plánovanie,
- sledovanie,
- riadenie,
- komunikácia

Pravdepodobnosť vzniku nežiaducej udalosti a veľkosť škody sa rozdeľujeme do troch úrovní:

- *malá*,
- *stredná*,
- *vysoká*

Kedy sa bude riešiť potenciálny alebo vzniknutý problém rozdeľujeme na:

- *nerieši sa*,
- *kontinuálne*,
- *pri udalosti*

Uvádzame tabuľku s identifikovanými rizikami:

Číslo	Riziko	Pravdepodobnosť vzniku	Veľkosť škody	Kedy vyhodnocovať riziko	Spôsob riešenia rizika
1	Dočasná práceneschopnosť člena tímu	stredná	stredná	pri udalosti	Presunúť aktuálne úlohy práceneschopného člena na iných členov tímu
2	Ukončenie štúdia člena tímu	malá	vysoká	nerieši sa	
3	Nerealistické ciele	stredná	vysoká	kontinuálne	Prehodnotiť ciele a presunúť úlohy v tíme
4	Vývoj nevhodnej funkcionality	stredná	stredná	kontinuálne	Vedúci tímu musí dohliadnuť na to aby sa vyvíjalo, len to čo je potrebné
5	Nevhodné navrhnutie používateľského	malá	malá	pri udalosti	Testovanie a hodnotenie intuitívnosti

	rozhrania				používateľského rozhrania
6	Pozlacovanie kódu	vysoká	malá	kontinuálne	Manažér vývoja a kvality dohliadnu na vývoj systému
7	Spojité zmeny požiadaviek	malá	malá	nerieši sa	
8	Podcenenie technológie	vysoká	vysoká	kontinuálne	Včasné hľadanie podobných riešení
9	Precenenie technológie	vysoká	vysoká	kontinuálne	Včasné hľadanie podobných riešení
10	Nová technológia	stredná	malá	kontinuálne	Predbežne zistiť, aké technológie sa budú používať a v prípade potreby, vyhradiť si na ne čas a ľudí (spravenie prezentácie, predstavenie technológie a jej záludností ostatným členom tímu)
11	Problémy s integráciou kódu	stredná	vysoká	pri udalosti	Pri tejto situácii sa treba obrátiť na manažéra podpory vývoja
12	Porucha servera	malá	vysoká	kontinuálne & pri udalosti	Lokálne zálohy zdrojových kódov a dát
13	Riziko straty dát	malá	vysoká	kontinuálne	Priebežne zálohovať databázy
14	Nedostatočná vzorka trénovacích a testovacích dát	vysoká	vysoká	kontinuálne	Priebežne sa snažiť získavať trénovanie a testovacie dáta.
15	Nevhodné zadeľovanie zreťazených úloh v tíme	vysoká	stredná	kontinuálne	Presunúť úlohu, alebo poveriť člena tímu, aby pomohol z danou úlohou

Tabulka 1 - tabulka rizík

8 MANAŽMENT KVALITY

9 MANAŽMENT MONITOROVANIA PROJEKTU

Táto kapitola sa venuje procesu monitorovania projektu. Monitorovanie prebieha počas celého životného cyklu projektu. Na konci každého šprintu je vytvorená krátka správa, ktorá zobrazuje vyhodnotenie splnenia úloh oproti plánu pre uplynulý šprint.

Za monitorovanie zodpovedá: Miroslav Hudák

9.1 ŠPRINT 0

Táto časť zobrazuje vyhodnotenie plnenia úloh zadaných pre úvodný šprint.

Vyhodnotenie úloh:

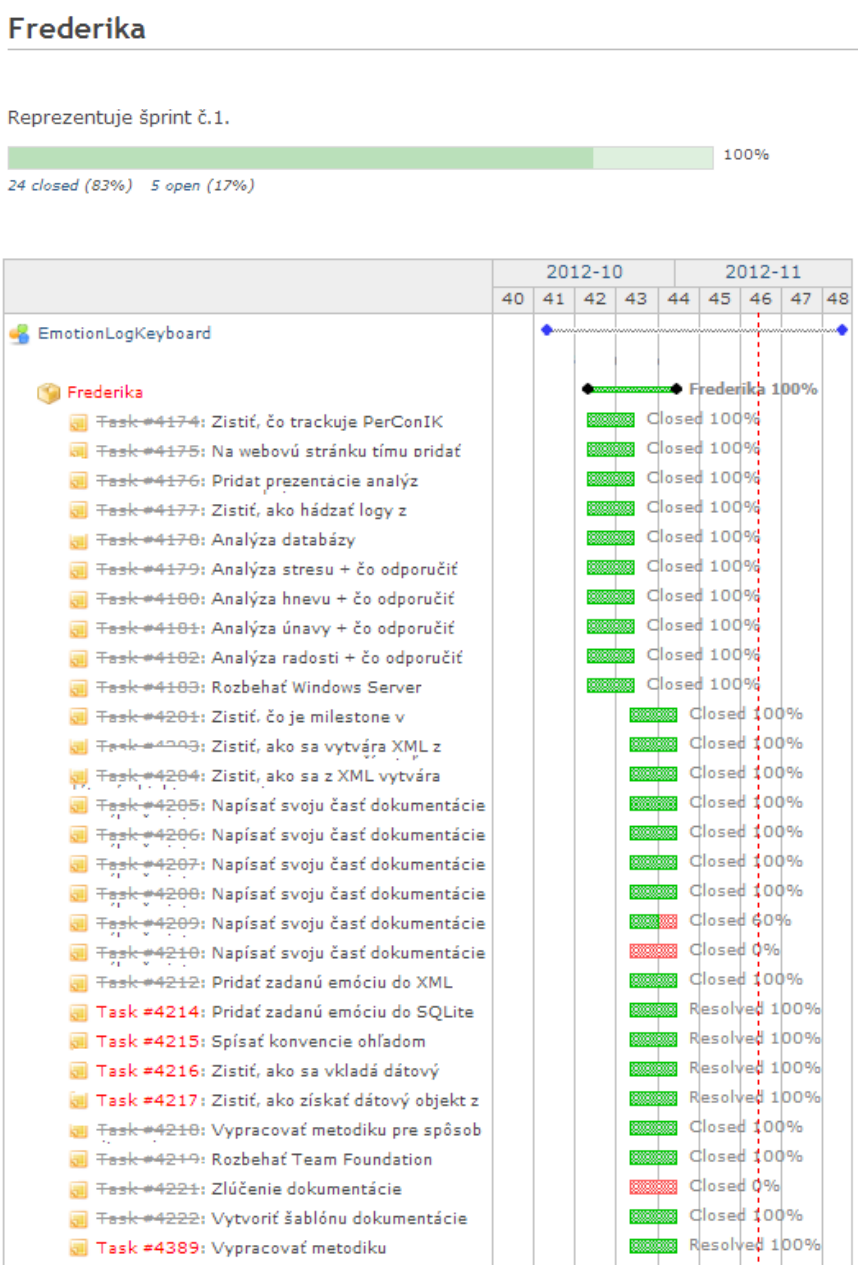
Vytvorenie šablóny zápisov	100 %
Rozbehať Redmine – vytvoriť projekt, nahádzať ľudí, pridať user stories.	100 %
Urobiť webstránku tímu	100 %
Analyzovať existujúce logovacie softvéry	100 %
Aké odporúčania sú vhodné na ponúknutie používateľovi?	100 %
Rozbehať logger	60 %
Techniky logovania, aké rôzne veci sa môžu logovať	100 %
Prejsť články na emocionálne stavy	100 %
(Rozpoznávanie) analýza modelov	100 %

Záver:

V tomto úvodnom šprinte bolo zadaných 9 úloh. V danom období sa ešte nepoužíval manažovací systém, preto tieto úlohy v ňom nie sú definované. Úlohy boli splnené na 96 %. Nesplnené úlohy boli presunuté do nasledujúceho šprintu s najvyššou prioritou plnenia.

9.2 Sprint 1

Táto časť zobrazuje vyhodnotenie plnenia úloh zadaných pre prvý šprint pomenovaný *Frederika*.



Záver:

V tomto šprinte bolo zadaných 29 úloh a 1 prenesená úloha z predchádzajúceho šprintu. Splnenie úloh trvalo 74 hodín, čo predstavuje v priemere približne 12 hodín na jedného člena tímu. Manažovací systém zobrazuje úroveň plnenia úloh na 83 %. Avšak toto percento nevypovedá o reálnom splnení úloh, pretože na konci šprintu sa nestihol vykonať proces

vyhodnotenia a uzatvorenia úloh testermi pre každú zadanú úlohu. Manažér monitorovania upovedomil vedúceho tímu o tejto skutočnosti.

10 Manažment podpory vývoja

11 MANAŽMENT KOMUNIKÁCIE

Pozíciu manažéra komunikácie zastáva člen Bc. Martin Geier. Táto pozícia v sebe zahŕňa aj pozíciu vedúceho týmu a pretože vyvíjame metódou *SCRUM* aj pozíciu *SCRUM Master-a*.

11.1 METÓDA VÝVOJA SCRUM

Vývoj metódou SCRUM prebieha v malých tímoch, obvykle o 6 - 7 členoch. V tíme z pohľadu metódy SCRUM máme členov:

- produktový vlastník *(produkt owner)* – člen zastávajúci stakeholderov - Doc. Mgr. Daniela Chudá, PhD.
- *SCRUM master,*
- tím.

Metóda vývoja SCRUM pozostáva zo šprintov ktoré sú vždy rovnako dlhé, najčastejšie 30 dní. Náš tím skrátil dobu trvania šprintov na 14 dní. Po 7 dňoch od začiatku šprintu sa uskutočňuje naplánované stretnutie tímu aj s produktovým vlastníkom. Počas doby trvania šprintu dochádza aj ku viacerým neplánovaným stretnutiam členov tímu.

11.1.1 BACKLOG

Na úvodnom stretnutí sme si ako tím spolu s produktovým vlastníkom definovali používateľské príbehy, ktoré majú pokryť úplnú funkcionalitu vyvíjaného softvéru. Pretože sme na semester naplánovali 4 šprinty, používateľské príbehy sú taktiež 4 a každý šprint by mal pokrývať jeden používateľský príbeh.

Každý používateľský príbeh sme rozdelili na úlohy, ktoré jemnejšie opisujú funkcionalitu systému.

Súčasťou *backlogu* sú aj všetky nápady a myšlienky, ktoré by systém mohol alebo mal spĺňať. Nie všetky časti je však možné implementovať a preto úlohy, ktoré budeme implementovať vyberá produktový vlastník, podľa jeho požiadaviek a priority.

11.1.2 VYTVÁRANIE PODÚLOH A PLÁNOVANIE ŠPRINTOV

Šprinty sa plánujú v dvojtýždňových intervaloch. Pred začatím šprintu je potrebné vytvoriť podúlohy, ktoré je možné prideliť členom tímu. Podúlohy sú vytvárané z *backlogu* a aktuálneho stavu projektu. Počas stretnutia členovia tímu diskutujú o podúlohách pričom sú monitorovaný produktovým vlastníkom, ktorý môže do priebehu zasiahnuť a vyjadriť názor k diskutovanej časti. Z diskusie priamo vyplývajú podúlohy ktoré je potrebné

vyriešiť. Takýmto spôsobom súčasne vzniká aj opis podúloh. Vytvorené podúloh sú následne ohodnotené na základe časovej zložitosti.

Ak sú vytvorené podúlohy a pridelené hodnotenia, SCRUM master pridelí podúlohy jednotlivým členom tímu s prihliadnutím na zložitosť podúlohy a predchádzajúcej práce člena, ktorá môže poskytovať potrebné vedomosti.

Po ukončení šprintu je úlohou SCRUM mastera ohodnotenie členov tímu a informovanie učiteľského dozoru o výkonoch a výsledkoch tímu.

12 METODIKY

Táto kapitola obsahuje metodiky všetkých členov tímu.

12.1 METODIKA PRE MANAŽMENT CHÝB A TVORBA ZÁZNAMU O CHYBE V SYSTÉME REDMINE

12.1.1 ÚVOD

Cieľom tejto metodiky je zadefinovanie manažmentu chýb v životnom cykle vývoja softvéru. Metodika podrobne rozoberá procesy definované v rámci manažmentu chyby. V druhej časti metodiky je detailne rozobraný proces Vytvorenie záznamu o chybe, konkrétne v systéme *Redmine*.

Táto metodika je určená pre malé tímy.

12.1.2 SKRATKY A POJMY

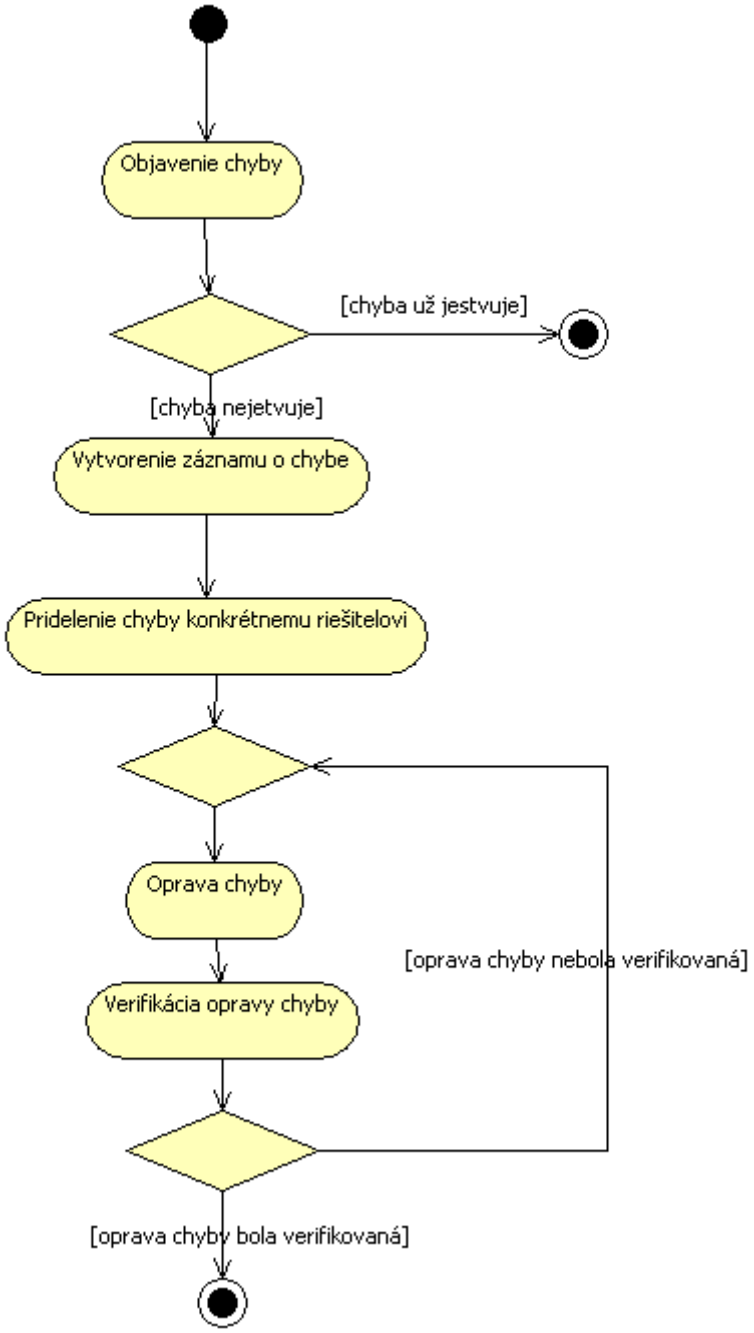
Redmine	Nástroj na podporu vývoja softvéru.
BTS	Bug Tracking System - Systém na sledovanie chýb.
Stakeholder	Osoba, ktorá má relevantný vzťah ku softvérovému systému.
Bug	Chyba v systéme.
url	Doslovne jednotný vyhľadávač zdrojov, je to univerzálny formát používaný na označenie zdroja na internete.

12.1.3 Procesy manažmentu chýb

Metodika manažmentu chýb zahŕňa tieto základné procesy:

1. Objavenie chyby
2. Vytvorenie záznamu o chybe
3. Priradenie chyby ku konkrétnemu riešiteľovi
4. Oprava chyby
5. Verifikovanie opravenia chyby

Nasledujúci diagram činností zobrazuje procesy manažmentu chýb, ich detailný opis sa nachádza v kapitole 5.


Objavenie chyby
[chyba už jestvuje]
[chyba nejetvuje]
Vytvorenie záznamu o chybe
Pridelenie chyby konkrétnemu riešitelovi
Oprava chyby
[oprava chyby nebola verifikovaná]
Verifikácia opravy chyby
[oprava chyby bola verifikovaná]

12.1.4 ROLY A ZODPOVEDNOSTI ÚČASTNÍKOV

Metodika pre manažment chýb zahrňuje nasledujúce roly v softvérovom tíme:

Rola	**Zodpovednosť**	**Proces**
Zadávateľ chyby do systému	Nájdenie chyby	Objavenie chyby
	Vytvorenie reportu o chybe	Vytvorenie záznamu o chybe
Vedúci tímu	Priradenie chyby zodpovednej osobe	Priradenie chyby ku konkrétnemu riešiteľovi
Vývojár	Opravuje chybu	Oprava chyby
Tester	Opravuje chybu v testoch	Oprava chyby
	Nájdenie chyby testami	Objavenie chyby
	Vytvorenie reportu o chybe	Vytvorenie záznamu o chybe
	Kontrola opravenia chyby	Verifikovanie opravenia chyby

Vývojár v rámci tímu môže byť analytik, návrhár alebo programátor, pričom každý z nich opravuje časť projektu, za ktorú zodpovedá.

12.1.5 PROCESY V MANAŽMENTE CHÝB

12.1.5.1 OBJAVENIE CHYBY

Vstup: Chyba sa nachádza v softvérovom systéme.

Výstup: Informácie o chybe a jej opis.

Identifikované role: Zadávateľ chyby do systému, Tester.

Proces objavenia chyby opisuje spôsob, akým sa chyba identifikuje.

Objavenie chyby sa môže udiať niekoľkými spôsobmi:

1. Objavenie chyby statickou prehliadkou zdrojového kódu.
2. Objavenie chyby počas behu spustenej aplikácie.
3. Objavenie chyby prostredníctvom notifikačného systému, systém pri detekcii chyby zašle notifikáciu zodpovednej osobe.
4. Objavenie chyby v analýze.
5. Objavenie chyby v návrhu softvéru.
6. Objavenie chyby v logoch.
7. Iné.

Chybu v systéme môže odhaliť akýkoľvek *stakeholder*, ktorý príde so systémom do kontaktu, ale nie vždy má právo vkladať opis chyby do systému na sledovanie chýb. V takom prípade ju oznámi zodpovednej osobe.

12.1.5.2 VYTVORENIE ZÁZNAMU O CHYBE

Vstup: Informácie o chybe a jej opis.

Výstup: Chyba je zapísaná do systému na sledovanie chýb.

Identifikované role: Zadávateľ chyba do systému, Tester.

Proces Vytvorenie záznamu o chybe popisuje vloženie záznamu do systému na sledovanie chýb.

Základné kroky procesu:

1. Ak zadávateľ chyby do systému nie je tá istá osoba, ktorá identifikovala chybu, tak najskôr overí, či oznámená chyba skutočne existuje (táto časť procesu sa nerozoberá v kapitole Detailný opis procesu Vytvorenie záznamu o chybe).
2. Zadávateľ chyby do systému najskôr vyhľadá v systéme na sledovanie chýb chybu s rovnakým alebo podobným opisom.
    a. Ak taká chyba už jestvuje, doplní údaje do reportu chyby.
    b. Ak taká chyba nejestvuje, vytvorí nový záznam o chybe. Táto chyba sa označí ako nová.

Proces Vytvorenie záznamu o chybe je detailne popísaný v kapitole Detailný opis procesu Vytvorenie záznamu o chybe.

12.1.5.3 PRIRADENIE CHYBY KU KONKRÉTNEMU RIEŠITEĽOVI

Vstup: Chyba je zapísaná do systému na sledovanie chýb.

Výstup: Chyba je pridelená konkrétnemu riešiteľovi.

Identifikované role: Vedúci tímu.

V tomto procese vedúci tímu rozhodne, ktorému vývojárovi je najvhodnejšie prideliť chybu na opravu.

Základné kroky procesu:

1. Vedúci tímu si prečíta hlásenie o chybe.
2. Vedúci tímu analyzuje chybu.
3. Vedúci tímu posúdi relevantnosť chyby
    a. Ak sa nejedná o chybu alebo jej opravenie nie je potrebné (napríklad modul, na ktorom vznikla chyba, sa nebude vyvíjať), tak sa chyba v systéme na sledovanie chýb zruší a označí sa ako zrušená.
    b. Ak sa chyba už v systéme nachádza alebo sa v ňom nachádza podobná chyba, je otvorená a účastne priradená vývojárovi, tak do nej doplní informácie.
    c. Ak sa chyba už v systéme nachádza a je označená ako uzavretá, tak ju vedúci tímu označí ako znovuotvorenú, a ak je potrebné, doplní do nej ďalšie informácie.

d. Ak ide o relevantnú chybu, ktorá v systéme na sledovanie chýb nemá priradeného konkrétneho riešiteľa, tak ju priradí vývojárovi, ktorý má danú časť softvérového systému na starosti, podieľal sa na jeho vývoji alebo je to expert v danej oblasti (Tento proces priraďovania opisuje metodika *Metodika prideľovania úloh* .)

12.1.5.4 OPRAVA CHYBY

Vstup: Chyba je pridelená konkrétnemu riešiteľovi.

Výstup: Chyba je označená ako opravená.

Identifikované role: Vývojár,Tester.

Proces Oprava chyby opisuje spôsob opravenia chyby. Zahŕňa v sebe aj reverzné testovanie.

1. Vývojár si prečíta záznam o chybe v systéme na sledovanie chýb.
2. Vývojár preskúma podmienky, za ktorých chyba vznikla.
3. Vývojár opraví chybu.
4. Vývojár spustí reverzné testovanie, ak prebehlo úspešne, pokračuje krokom 5, inak musí znovu opraviť chybu.
5. Vývojár označí chybu ako vyriešenú.

12.1.5.5 VERIFIKOVANIE OPRAVENIA CHYBY

Vstup: Chyba je označená ako opravená.

Výstup: Chyba je uzatvorená./Chyba je priradená.

Identifikované role: Tester.

Proces Verifikovanie opravenia chyby opisuje spôsob vyhodnotenia chyby, ktorá bola označená ako opravená, či je naozaj opravená.

1. Ak je možné vytvoriť test na verifikáciu opravenia chyby, tak ho tester vytvorí. (Tento proces využíva metodiku Metodika tvorby reverzných testov.)
2. Ak jestvuje test na verifikáciu chyby, tester ju použije.
   a. Ak opravená časť softvérového systému prejde testami, tester označí chybu ako uzavretú.
   b. Ak opravená časť softvérového systému neprejde testami, tester vráti chybu znovu na opravenie a označí ju ako priradenú.

12.1.6 SÚVISIACE METODIKY

- Metodika prideľovania úloh
- Metodika tvorby reverzných testov

12.1.7 DETAILNÝ OPIS PROCESU VYTVORENIE ZÁZNAMU O CHYBE

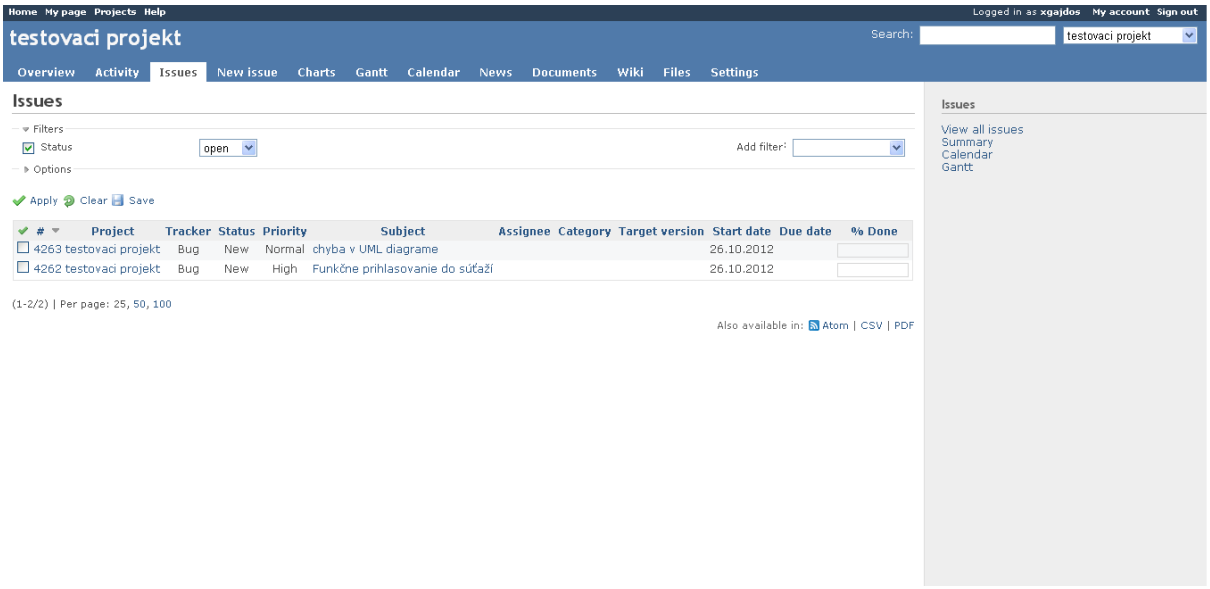
Táto kapitola opisuje proces Vytvorenia záznamu o chybe v systéme na sledovanie chýb *Redmine* a poskytuje návod a popis pre použitie tohto systému.

Autorom záznamu o chybe môžu byť role Zadávateľ chyby do systému a Tester, Zadávateľ chyby do systému je osoba, ktorá ma nato oprávnenie.

12.1.7.1 ZISTENIE, ČI ROVNAKÁ ALEBO PODOBNÁ CHYBA UŽ V SYSTÉME NA SLEDOVANIE CHÝB JESTVUJE

Cieľom tohto kroku je zistiť, či už jestvuje rovnaká alebo podobná chyba zapísaná v systéme na sledovanie chýb *Redmine*.

1. Zadávateľ chyby do systému sa prihlási do systému *Redmine* svojim prihlasovacím menom a heslom.
2. V ľavom hornom rohu klikne myšou na *Projects* .
3. Zobrazí sa mu stránka s aktuálnymi projektami, z ktorých si vyberie kliknutím ten, v ktorom chce overiť existenciu hlásenia chyby. Táto obrazovka je na obrázku **Obrázok 2 - zoznam zapísaných chýb**.
4. Následne sa mu zobrazí stránka *Overview* projektu, kde má základné informácie o projekte, kliknutím na položku *Issuses* sa dostane do zoznamu chýb, vylepšení v projekte a podobne .
5. Zvolí filter typu *All* a kline na *Appli*. Tým sa zobrazia všetky bugy v projekte.
6. Zadávateľ chyby v zozname všetkých chýb pohľadá, či sa v ňom nachádza podobná alebo rovnaká chyba a pokračuje krokom **Zapísanie chyby do systému na sledovanie chýb**. Ak sa v systéme takáto chyba nenachádza, tak pokračuje krokom **Doplnenie informácií o chybe**.



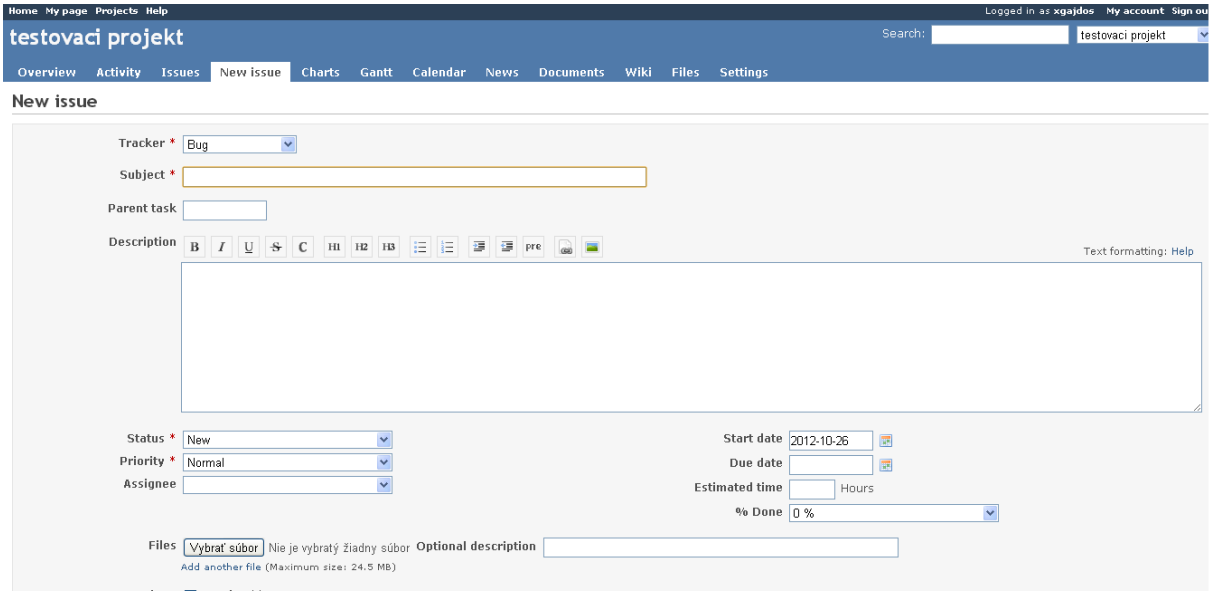
OBRÁZOK 2 - ZOZNAM ZAPÍSANÝCH CHÝB

12.1.7.2 ZAPÍSANIE CHYBY DO SYSTÉMU NA SLEDOVANIE CHÝB

Cieľom tohto kroku je zapísať identifikovanú chybu do systému *Remine*.

1. Zadávateľ chyby do systému *Redmine* klikne na *New issuse*. Následne sa mu zobrazí formulár pre pridávanie chýb. Tento formulár je na **Obrázok 3 - formulár na zapísanie chyby**.
2. *Tracker* sa nastaví na *Bug*.
3. Vyplní *Subject*, teda predmet chyby, tento popis nesmie obsahovať slovesá, iba vecné pomenovania. Napríklad : „Nefunkčné prihlasovanie administrátora“ a nie „Administrátor sa nemôže prihlásiť“ .
4. Ak oprava práve zadávanej chyby závisí od opravy inej chyby alebo vytvorenia či úpravy časti softvérového systému, tak zadávateľ vyplní aj položku *Parent task*, kde zadá názov chyby, ktorá ma byť opravená pred práve zadávanou chybou.
5. Zadávateľ chyby musí vyplniť *Description* nasledovne:
    a. Na úvod v krátkosti opíše stručný opis chyby niekoľkými vetami. Do úvodu patria aj url linky, ktoré priamo ukazujú na miesto, kde sa chyba nachádza, keď sa jedná o webový systém.
    b. V nasledujúcom odstavci podrobne opíše, za akých podmienok chyba nastala, ako sa prejavuje, prípadne uvedie, ako by sa mal systém správať po oprave chyby.
    c. Ak sa k opisu chyby prikladajú prílohy, tak sa do opisu chyby vloží ďalší odstavec označený ako „Opis príloh“, v ktorom bude opis príloh, ktorý je dlhší ako jeden riadok. Tento opis je na zvážení zadávateľa do systému.
6. Položku status nastaví na New.

7. Zadávateľ chyby zvolí položku Priority podľa urgentnosti opravy chyby, čím je oprava chyby urgentnejšia, tým má vyššiu prioritu.
8. Ak má zadávateľ chyby do systému aj súbory, ktoré súvisia s chybou alebo jej opisom, tak sa priložia ako prílohy v sekcii *Files*. Ku každému súboru musí napísať krátky opis v kolónke *Optimal description*. Do príloh patria súbory ako *screenshot*-y obrazoviek, *stack-trace výpisy*, maily od notifikačných systémov a iné.
9. Nakoniec stlačí tlačidlo *Submit* a tým uloží nové hlásenie o chybe.



OBRÁZOK 3 - FORMULÁR NA ZAPÍSANIE CHYBY

12.1.7.3 DOPLNENIE INFORMÁCIÍ O CHYBE

Cieľom tohto kroku procesu je doplnenie informácii do už jestvujúceho zápisu o chybe v systéme *Redmine*. Zadávateľ chyby do systému sa nachádza v zozname všetkých chýb a našiel duplicitnú chybu.

1. Zadávateľ chyby do systému klikne na *Subject* chyby, ktorú chce doplniť.
2. Následne sa mu zobrazia podrobnosti.
3. Klikne dole vpravo na *Update* a *Redmine* zobrazí formulár na doplnenie informácií.
4. Ak ide o uzavretú alebo vyriešenú chybu, tak sa jej *Status* zmení na *Rejected*.

5. Zadávateľ chyby do systému vyplní ďalšie informácie o chybe do textarey *Node*, ktoré musia byť štruktúrované podobne ako pri zadávaní novej chyby do systému.
6. Ak je k hláseniu o chybe ešte potrebné pridať prílohy, nahrajú sa na server pomocou položky *Files* a pre každú prílohu musí zadávateľ chyby do systému pridať krátky opis do položky *Optimal description*.
7. Nakoniec formulár odošle kliknutím na tlačidlo *Submit*.

12.2 METODIKA PRE MANAŽMENT ÚDRŽBY SOFTVÉRU POMOCOU METÓDY RAFAKTOROVANIA

12.2.1 ÚVOD

Účelom tejto metodiky je stanoviť postup pre údržbu softvéru a údržbu zdrojového kódu pomocou metódy refaktorovania vo vývojovom prostredí *Microsoft Visual Studio 2010* za podpory zásuvný modul *R# ReSharper 7*.

12.2.2 ZOZNAM POJMOV

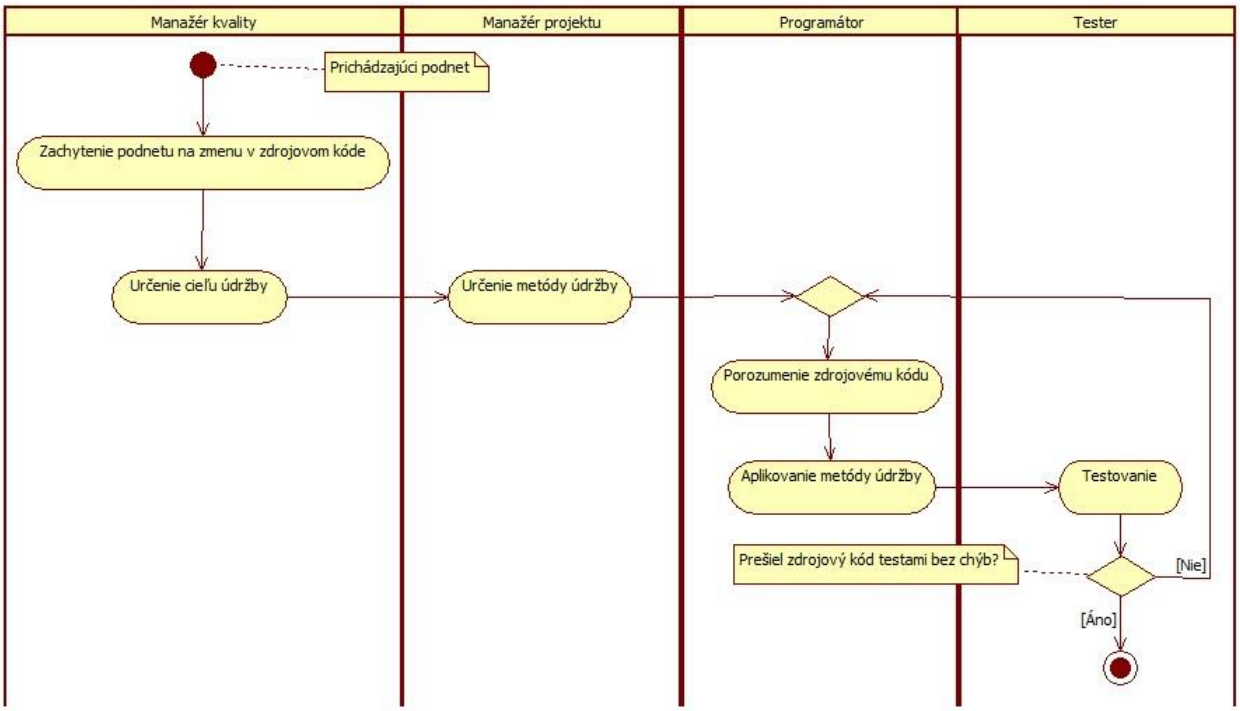
- *Refactoring* – proces, pri ktorom sa softvér mení tak, že sa vylepšuje vnútorná štruktúra kódu, pričom sa neovplyvňuje jeho vonkajšie správanie
- *Visual Studio 2010* – vývojové prostredie do ktorého sa pridá a bude využívať plug-in pre údžbu zdrojového kódu
- *R# ReSharper 7* – zásuvný modul (*plug-in*) na údržbu zdrojového kódu pre *Microsoft Visual Studio*
- *Zásuvný modul* – softvérový komponent, ktorý rozširuje špecifickú vlastnosť väčšej softvérovej aplikácie
- *Prepravka* – návrhový vzor, ktorý je určený na prenos dát – implementačne trieda bez metód
- *Továrenská metóda* – návrhový vzor, ktorý zapuzdruje konštruktor do metódy

12.2.3 ROLE A ZODPOVEDNOSTI

	Rola	Zodpovednosť	Proces
1.	Manažér kvality	Zachytenie podnetov	Zachytenie podnetu na zmenu v zdrojovom kóde
		Určenie cieľu a jeho špecifikácia	Určenie cieľu údržby
2.	Manažér projektu	Výber metódy údržby	Určenie metódy údržby
		Pridelenie úlohy programátorovi	Určenie metódy údržby
3.	Programátor	Naštudovanie zdrojového kódu	Porozumenie zdrojovému kódu
		Implementácia metódy údržby	Aplikovanie metódy údržby
4.	Tester	Testovanie upraveného kódu	Testovanie

12.2.4 PROCES ÚDRŽBY SOFTVÉRU

V tejto kapitole je popísaný postup údržby softvéru.



	Krok	Kapitola
1.	Zachytenie podnetu na zmenu v zdrojovom kóde	12.2.4.1
2.	Určenie cieľu údržby	12.2.4.2
3.	Určenie metódy údržby	12.2.4.3
4.	Porozumenie zdrojovému kódu	12.2.4.4
5.	Aplikovanie metódy údržby	12.2.4.5
6.	Testovanie	12.2.4.6

12.2.4.1 ZACHYTENIE PODNETU NA ZMENU V ZDROJOVOM KÓDE

Vstup: Podnet na zmenu v zdrojovom kóde.

Výstup: Požiadavka po zmene v zdrojovom kóde.

Zodpovedný: Manažér kvality.

Manažér kvality zachytáva požiadavky na zmenu zdrojového kódu, ktoré môžu prichádzať z rôznych zdrojov:

- Zákazník
- manažér projektu
- programátor
- tester
- softvér na zisťovanie metrík kódu
- iné

12.2.4.2 Určenie cieľu údržby

Vstup: Požiadavka po zmene v zdrojovom kóde.

Výstup: Zvolený cieľ údržby softvéru.

Zodpovedný: Manažér kvality.

Manažér kvality určí cieľ údržby kódu na základe zisteného nedostatku v zdrojovom kóde a tento cieľ konkrétne špecifikuje.

	Zistenie nedostatku v zdrojovom kóde	Cieľ
1.	Nahlásená chyba	Oprava chyby
2.	Dlhá doba behu programu	Optimalizácia
3.	Požiadavka od zákazníka	Pridanie/Odobranie/Zmena funkcionality
4.	Zlé metriky kódu	Zmena štruktúry kódu
5.	Iné	Určí manažér kvality podľa potreby

12.2.4.3 Určenie metódy údržby

Vstup: Zvolený cieľ údržby softvéru.

Výstup: Zvolená metóda údržby softvéru a pridelenie tejto úlohy programátorovi.

Zodpovedný: Manažér projektu.

Manažér projektu určí metódu údržby na základe cieľa, ktorý je potrebné dosiahnuť a poverí programátora pre vykonanie tejto metódy.

	Cieľ	Metóda
1.	Oprava chýb	Prepísanie časti zdrojového kódu, tak aby program prešiel testami
2.	Optimalizácia	Prepísanie časti zdrojového kódu, tak aby sa skrátila doba jeho vykonania
3.	Pridanie funkcionality	Pripísanie časti zdrojového kódu, tak aby program prešiel cez akceptačné testy požadovanej funkcionality
4.	Odobratie funkcionality	Vymazanie nadbytočnej časti zdrojového kódu
5.	Zmena štruktúry kódu	Refaktorovanie.
6.	Iné	Postup určí manažér projektu podľa potreby

12.2.4.4 POROZUMENIE ZDROJOVÉMU KÓDU

Vstup: Zdrojový kód, ktorý je potrebné preštudovať, aby mohla byť vykonaná jeho údržba.

Výstup: Naštudovaný zdrojový kód určený na údržbu programátorom.

Zodpovedný: Programátor.

Pre vykonanie implementačnej časti údržby si musí programátor naštudovať zdrojový kód do takej miery aby sa v ňom vedel pohybovať a efektívne vykonať potrebné zmeny.

12.2.4.5 APLIKOVANIE METÓDY ÚDRŽBY

Vstup: Zdrojový kód a metóda údržby, ktorá je nad ním potrebná vykonať.

Výstup: Upravený zdrojový kód.

Zodpovedný: Programátor.

Programátor pomocou postupu určeného v prvej časti procesu údržby upraví zdrojový kód, tak aby splnil cieľ určený v tejto časti.

12.2.4.6 TESTOVANIE

Vstup: Upravený zdrojový kód.

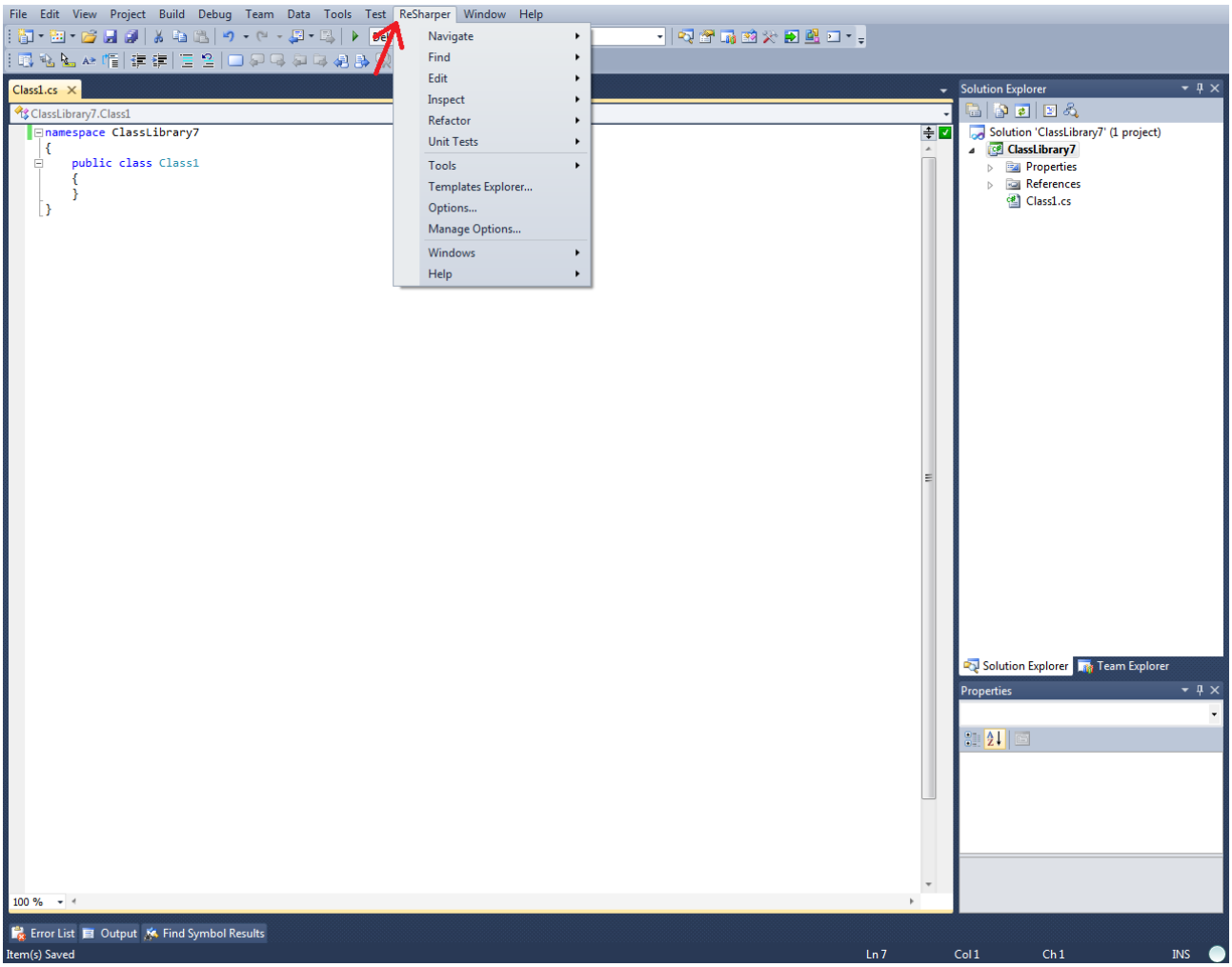
Výstup: Výsledok testov.

Zodpovedný: Tester.

Tester zistí stav testov po úprave zdrojového kódu. Ak program prešiel všetkými testami, aktuálny proces údržby sa ukončí. Ak niektoré testy oznámia chybu, tester vráti zdrojový kód manažérovi s požiadavkou na odstránenie chýb – vyvolá sa proces údržby s cieľom odstránenia chýb.

12.2.5 Proces aplikovania refaktorovania v *MS Visual Studio*

V nasledovnej časti je popísaný proces aplikovania refaktorovania ako metódy údržby vo vývojovom prostredí *Microsoft Visual Studio 2010* pomocou prostriedkov nainštalovaného zásuvného modulu *R# ReSharper 7*. Na nasledovnom obrázku je znázornený pridaný modul a jeho ponuka, kde sú



umiestnené viaceré nástroje pre údržbu systému.

Podľa metodiky pre určovanie pachov v zdrojovom kóde a ich následnom postupe odstránenia sú v nasledujúcej časti popísané najpoužívanejšie formy refaktorovania pomocou modulu *ReSharper 7*.

	Typ refaktorovania	Kapitola
1.	Odstraňovanie redundancií v zdrojovom kóde	12.2.5.1
2.	Extrakcia triedy podľa vzoru prepravka z parametrov metódy	12.2.5.2
3.	Extrakcia rozhrania z metód triedy	12.2.5.3
4.	Nahradenie konštruktora pomocou továrenskej metódy	12.2.5.4

12.2.5.1 ODSTRAŇOVANIE REDUNDANCIÍ V ZDROJOVOM KÓDE

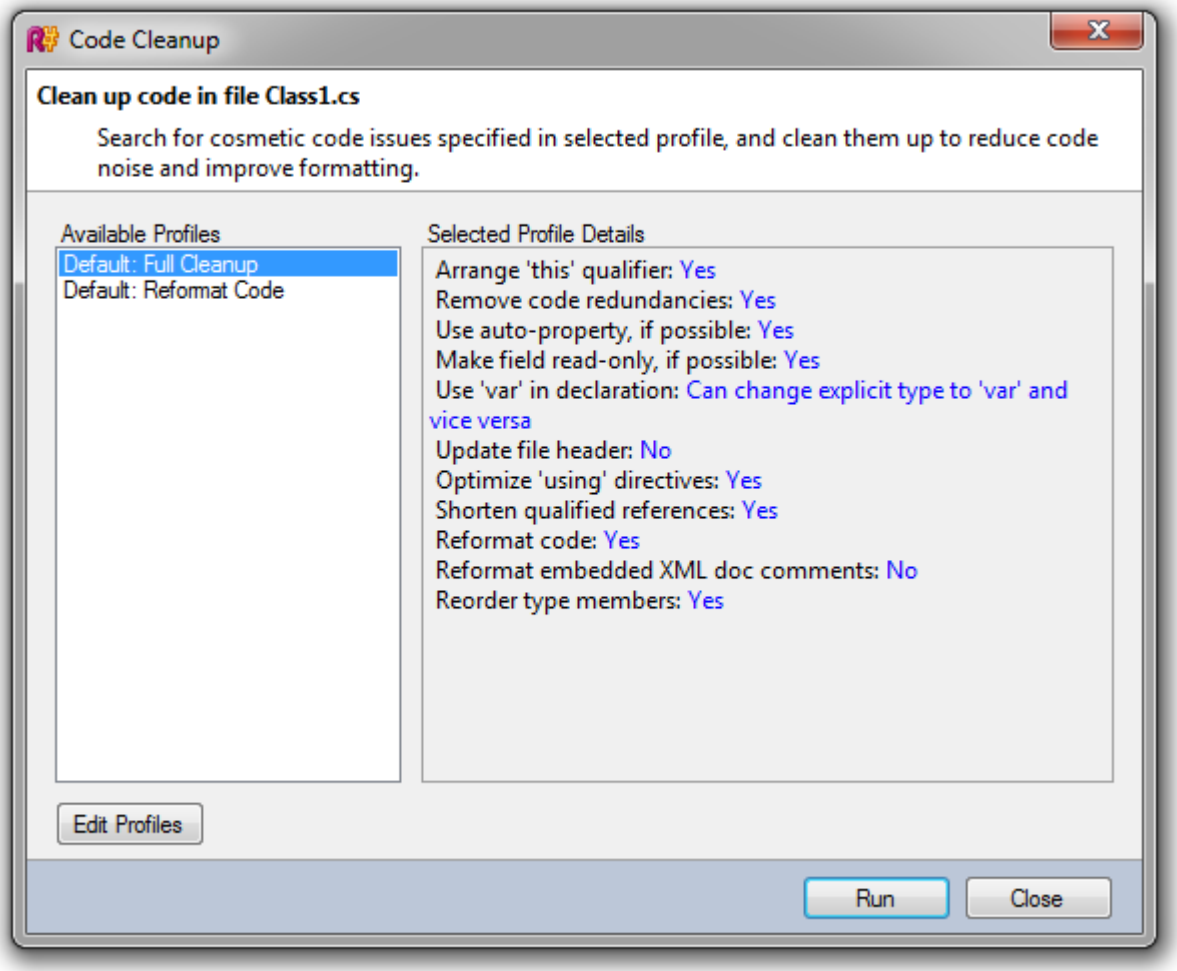
Nástroj *ReSharper* umožňuje analyzovať zdrojový kód a odstrániť v ňom redundantné časti (prázdne konštruktory, redundantné vytváranie delegátov a pod.).

Postup údržby:

V menu *MS Visual Studio* sa vyberie možnosť *ReSharper*.

V ponúknutom menu sa vyberie možnosť *Tools → Cleanup Code …*

Otvorí sa okno *Code Cleanup*, kde sú nastavené dva profily (*Full Cleanup* a *Reformat Code*), pre odstránenie redundancií sa vyberie profil *Full Cleanup* a potvrdí sa stlačením tlačidla Run.

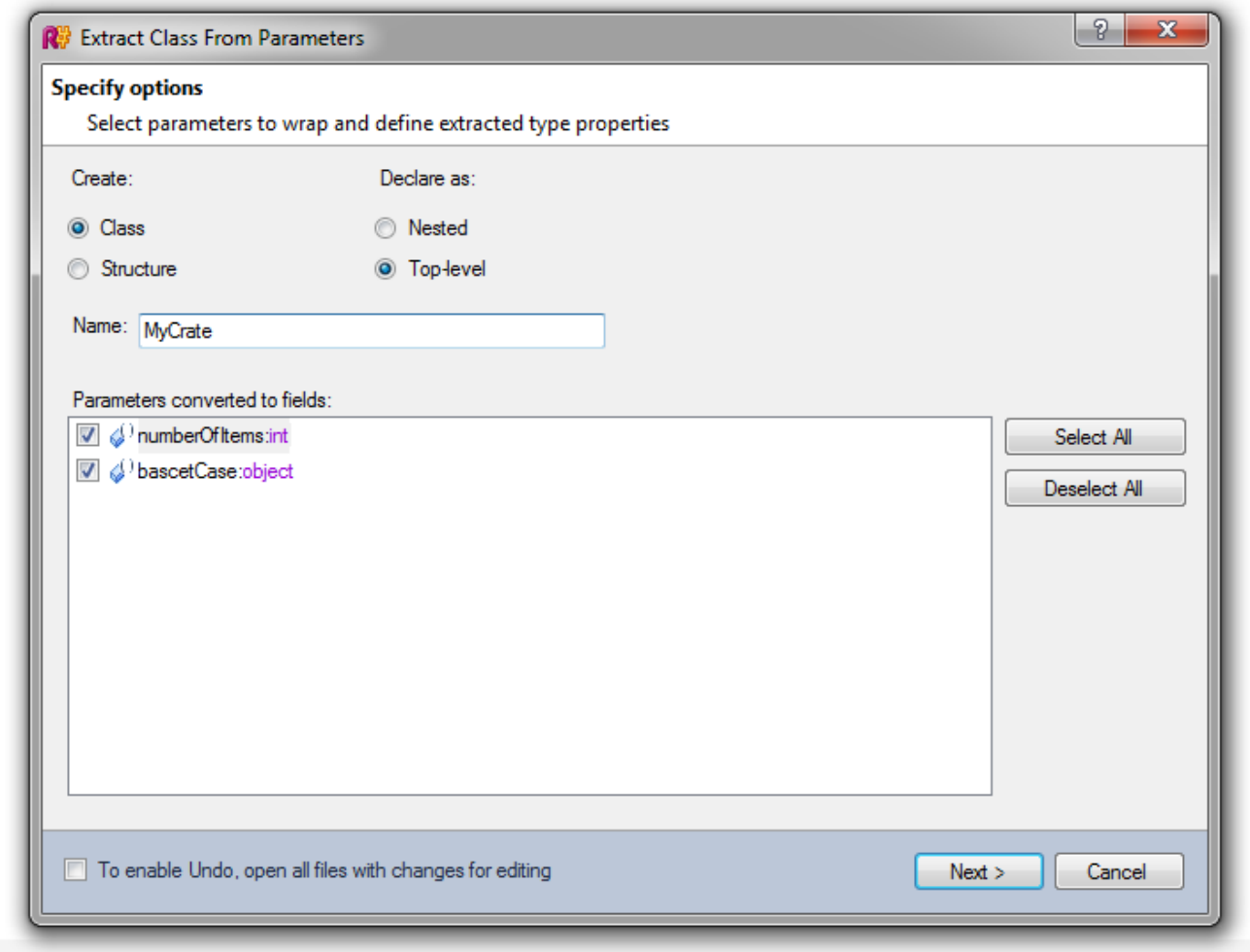


Poznámka: pre odstraňovanie redundancií v zdrojovom kóde je možné zvoliť aj možnosť vlastného profilu a upraviť si ho podľa potreby avšak je nutné aby položka *Remove code redundances* bola zvolená ako *Yes*.

12.2.5.2 EXTRAKCIA TRIEDY PODĽA VZORU PREPRAVKA Z PARAMETROV METÓDY

Postup údržby:

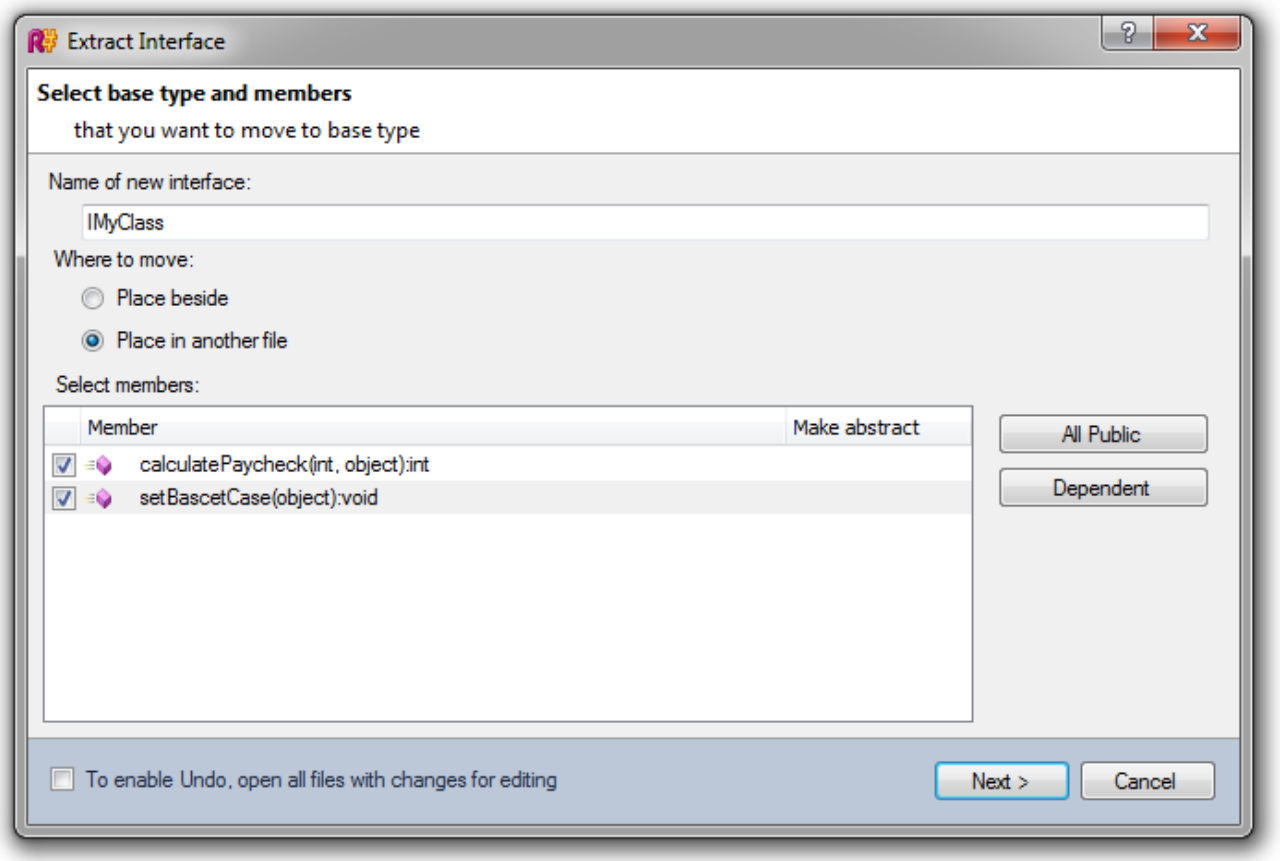
1. Pravý klik na metódu, z ktorej chceme vyextrahovať triedu.
2. V ponúknutom menu sa vyberie možnosť *Refactor → Extract Class From Parametres …*
3. V časti *Create* sa vyberie možnosť *Class*, ak nie je určené inak.
4. V časti *Declare* as sa vyberie možnosť *Top-level*, ak sa prepravka bude využívať vo viacerých triedach, alebo možnosť *Nested*, ak sa prepravka bude využívať len v aktuálnej triede.
5. Do políčka *Name* sa napíše názov vytváranej triedy (prepravky).
6. V časti *Parameters converted to fields* sa vyberú parametre, ktoré je potrebné vyextrahovať.
7. Vytvorenie novej triedy sa potvrdí stlačením tlačidla *Next*.



12.2.5.3 EXTRAKCIA ROZHRANIA Z METÓD TRIEDY

Postup údržby:

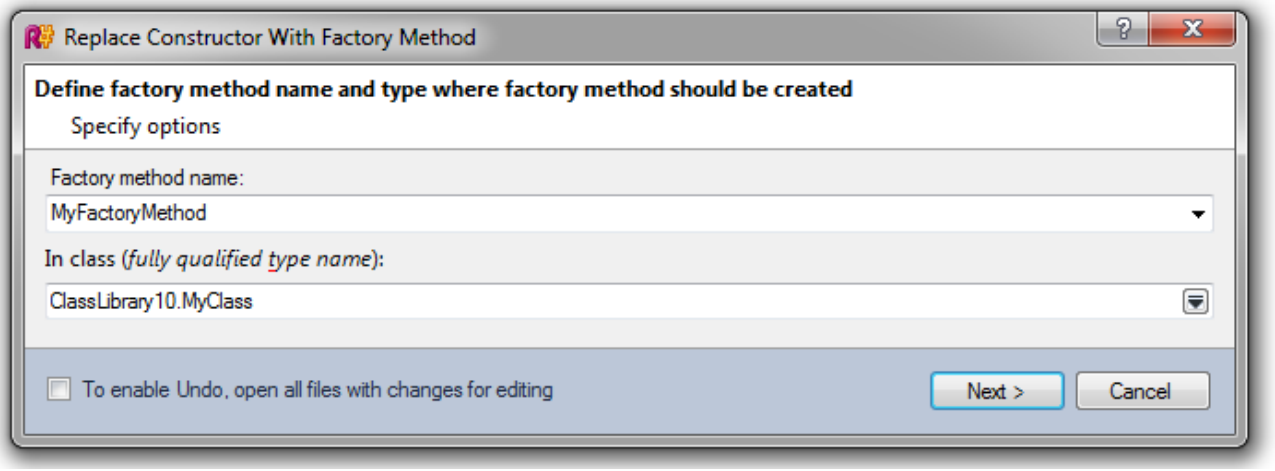
1. Pravý klik na triedu, z ktorej chceme vyextrahovať rozhranie.
2. V ponúknutom menu sa vyberie možnosť *Refactor → Extract Interface …*
3. Do políčka *Name of new interface* sa napíše názov vytváraného rozhrania.
4. V časti *Where to move* sa vyberie možnosť *Place in another file*, ak bude toto rozhranie dediť viacero tried, alebo možnosť *Place beside*, ak bude toto rozhranie dediť iba aktuálna trieda.
5. V časti *Select members* sa vyberú metódy ktoré sa vyextrahujú do rozhrania.
6. Vytvorenie rozhrania sa potvrdí stlačením tlačidla *Next*.



12.2.5.4 NAHRADENIE KONŠTRUKTORA POMOCOU TOVÁRENSKEJ METÓDY

Postup údržby:

1. Pravý klik na konštruktor, ktorý chceme zmeniť na továrenskú metódu.
2. V ponúknutom menu sa vyberie možnosť *Refactor* → *Replace Constructor with Factory Method …*
3. Do políčka *Factory method name* sa napíše názov vytváranej továrenskej metódy.
4. V časti *In class* sa vyberie trieda do ktorej chceme umiestniť továrenskú metódu.
5. Vytvorenie továrenskej metódy sa potvrdí stlačením tlačidla *Next*.


Replace Constructor With Factory Method
Define factory method name and type where factory method should be created
Specify options
Factory method name:
MyFactoryMethod
In class (fully qualified type name):
ClassLibrary10.MyClass
To enable Undo, open all files with changes for editing
Next >
Cancel

12.3 Vloženie používateľského príbehu do nástroja Redmine

12.3.1 Úvod

Metodika má za úlohu informovať čitateľa o postupoch pri zbere požiadaviek a definovať úlohy pre jednotlivých členov týmu, ktoré sú spojené s procesom zberu požiadaviek v agilnom spôsobe vývoja SCRUM.

Druhá časť dokumentu popisuje spôsob zadávanie požiadaviek vo forme používateľských príbehov do online nástroja Redmine. Súčasťou tejto časti je aj aktualizácia a zmena stavu používateľského príbehu.

12.3.2 Definovanie pojmov

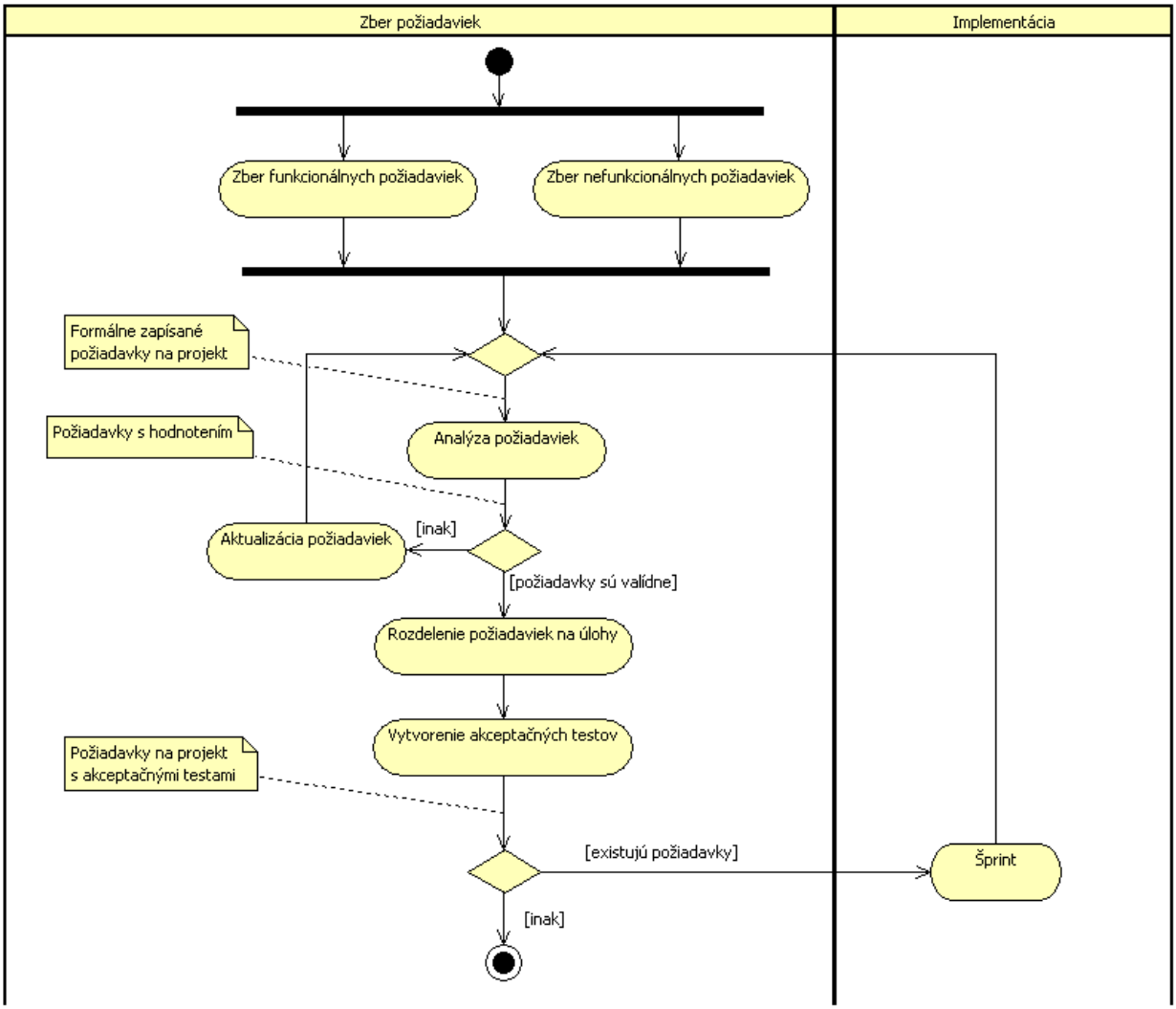
- Produktový vlastník
  - Zastupuje stakeholderov. Blízko spolupracuje so zákazníkom a vývojovým tímom. Musí rozumieť problémovej oblasti.
- *Scrum master*
  - Člen tímu, odstraňuje problémy spôsobené vonkajšími vplyvmi, motivuje členov tímu k lepším výkonom, upokojuje potýčky medzi členmi tímu.
- Používateľský príbeh (User story)
  - Úloha zapísané v tvare „Ako <rola>, chcem <cieľ> aby som mohol <dôvod>“. Udáva nielen čo sa má spraviť, ale aj kto to bude používať a na čo je to dobré.
- Redmine
  - *Open source* webový nástroj slúžiaci na manažment projektov. Zahŕňa Grantove diagramy, kalendár, wiki, fórum, role, emailové notifikácie a ďalšie.

12.3.3 Role a zodpovednosti

Rola	Proces	Zodpovednosť
Produktový vlastník	Zber funckionálnych a nefunkcionálnych požiadaviek	Vytvorenie používateľských príbehov
Produktový vlastník	Analýza požiadaviek	Preskúmanie aktuálneho stavu požiadaviek
Produktový vlastník	Aktualizácia požiadaviek	Úprava požiadaviek na projekt

Radový používateľ	Zber funckionálnych a nefunkcionálnych požiadaviek	Pomoc s vytvorením používateľských príbehov
Scrum master	Analýza požiadaviek	Oboznámenie sa s problémovými časťami dokumentu
Scrum master	Aktualizácia požiadaviek	Pomoc s opravou požiadaviek
Analytik	Analýza požiadaviek	Napísanie hodnotenia požiadaviek
Výkonný manažment	Vytvorenie akceptačných testov	Vytvorenie scenárov testov
Tester	Vytvorenie akceptačných testov	Vytvorenie scenárov testov
Tím	Rozdelenie požiadaviek na úlohy	Rozdelenie používateľských príbehov na úlohy
Manažér podpory vývoja	Rozdelenie požiadaviek na úlohy	Vloženie alebo aktualizácia požiadaviek

Tabuľka č. 1: role a zodpovednosti



Obrázok 1. Aktivity diagram zberu požiadaviek

12.3.4 PROCES ZBERU POŽIADAVIEK

	Krok	Kapitola
1.	Zber funkcionálnych požiadaviek	5.1
2.	Zber nefunkcionálnych požiadaviek	5.2
3.	Analýza požiadaviek	5.3
4.	Rozdelenie požiadaviek na úlohy	5.4
5.	Aktualizácia požiadaviek	5.5
6.	Vytvorenie akceptačných testov	5.6

Tabuľka č. 2: Proces zberu požiadaviek

12.3.4.1 ZBER FUNKCIONÁLNYCH POŽIADAVIEK

Vstup: Ústny opis požiadaviek na projekt

Výstup: Formálne zapísané požiadavky na projekt

Zodpovedný: Produktový vlastník, radový používatelia

Požiadavky, ktoré vysloví produktový vlastník budú zapísané vo forme používateľských príbehov. Ak produktový vlastník neuvedie časť „dôvod“, táto časť bude vynechaná zo zápisu. Používateľské príbehy budú zapísané v dokumente a budú zvýraznené štýlom bold. Zákazník zadá dôležitosť používateľského príbehu v rozsahu hodnotenia:

- nízka,
- normálna,
- vysoká,
- urgentná,
- bezprostredná

a následne uvedie dátum kedy je vyžadované ukončenie používateľského príbehu. Dátum a hodnotenie bude zapísané v prvom riadku za názvom používateľského príbehu. Za týmito informáciami budú nasledovať funkcionálne upresnenie, ak takéto upresnenia existujú, a bude im predchádza riadok s textom „Funkcionálne upresnenia:“. Za týmto riadkom

budú zapísané upresnenia v neočíslovanom zozname, každé na novom riadku. Pred každým ďalším používateľským príbehom bude voľný jeden riadok.

12.3.4.2 ZBER NEFUNKCIONÁLNYCH POŽIADAVIEK

Vstup: Ústny opis nefunkcionálnych požiadaviek na projekt

Výstup: Formálne zapísané požiadavky na projekt

Zodpovedný: Produktový vlastník, rádový používatelia

Každý používateľský príbeh môže byť doplnený o nefunkcionálne požiadavky. O potrebe doplnenia nefunkcionálnych požiadaviek rozhoduje produktový vlastník. Nefunkcionálne požiadavky budú doplnené za funkcionálne upresnenia a bude im predchádzať riadok s textom „Nefunkcionálne požiadavky“. Požiadavky budú zapísané v neočíslovanom zozname, každá na novom riadku.

12.3.4.3 ANALÝZA POŽIADAVIEK

Vstup: Formálne zapísané požiadavky na projekt

Výstup: Formálne zapísané požiadavky na projekt s hodnotením požiadaviek

Zodpovedný: Produktový vlastník, *Scrum master*, Analytik

Scrum master spolu s analytikom zhodnotia správnosť zadaných používateľských príbehov, funkcionálnych upresnení a nefunkcionálnych požiadaviek. V dokumente bude pod nefunkcionálne požiadavky doplnený nový riadok s textom „Zhodnotenie: “, za ktorým na nových riadkoch budú štruktúrovane zapísané nedostatky požiadaviek. Ak žiadne nedostatky neexistujú, pod textom „Zhodnotenie:“ bude text „Schválené“ zvýraznené štýlom bold.

12.3.4.4 ROZDELENIE POŽIADAVIEK NA ÚLOHY

Vstup: Formálne zapísané požiadavky na projekt s hodnotením požiadaviek

Výstup: Zapísané požiadavky a úlohy na projekt

Zodpovedný: Tím, Manažér podpory vývoja

Ak boli požiadavky schválené a vznikli nové požiadavky, alebo pôvodné požiadavky boli zmenené, tím rozdelí tieto požiadavky zapísané v tvare používateľských príbehov na úlohy. Súčasťou procesu je aj zanesenie nových používateľských príbehov, alebo zmenených používateľských príbehov do

nástroja na manažment projektov podľa kapitoly 6 *Vloženie používateľských príbehov do nástroja Redmine*.

12.3.5 AKTUALIZÁCIA POŽIADAVIEK

Vstup: Formálne zapísané požiadavky na projekt s hodnotením požiadaviek

Výstup: Formálne zapísané požiadavky na projekt

Zodpovedný: Produktový vlastník, *Scrum master*

Ak je potrebné zmeniť zadanie niektorých funkcionálnych alebo nefunkcionálnych požiadaviek v projekte, *scrum master* pomáha produktovému vlastníkovi s vysvetlením a opravením problémových častí. Pôvodný dokument sa archivuje a časť ktorá obsahovala zhodnotenie používateľského príbehu bude vymazaná. Spôsob zápisu opravených požiadaviek je rovnaký ako v bodoch 5.1 a 5.2

12.3.6 VYTVORENIE AKCEPTAČNÝCH TESTOV

Vstup: Formálne zapísané požiadavky na projekt

Výstup: Formálne zapísané akceptačné testy

Zodpovedný: Výkonný manažment, Tester

Výkonný manažment spolu s členom týmu zodpovedného za testovanie vytvoria scenáre testovania nových používateľských príbehov. Testy budú zapísané v dokumente. Štruktúra dokumentu bude nasledovná:

Používateľské príbehy zvýraznené štýlom bold, nasledované scenárom testovania, vstupmi a očakávanými výstupmi na nových riadkoch.

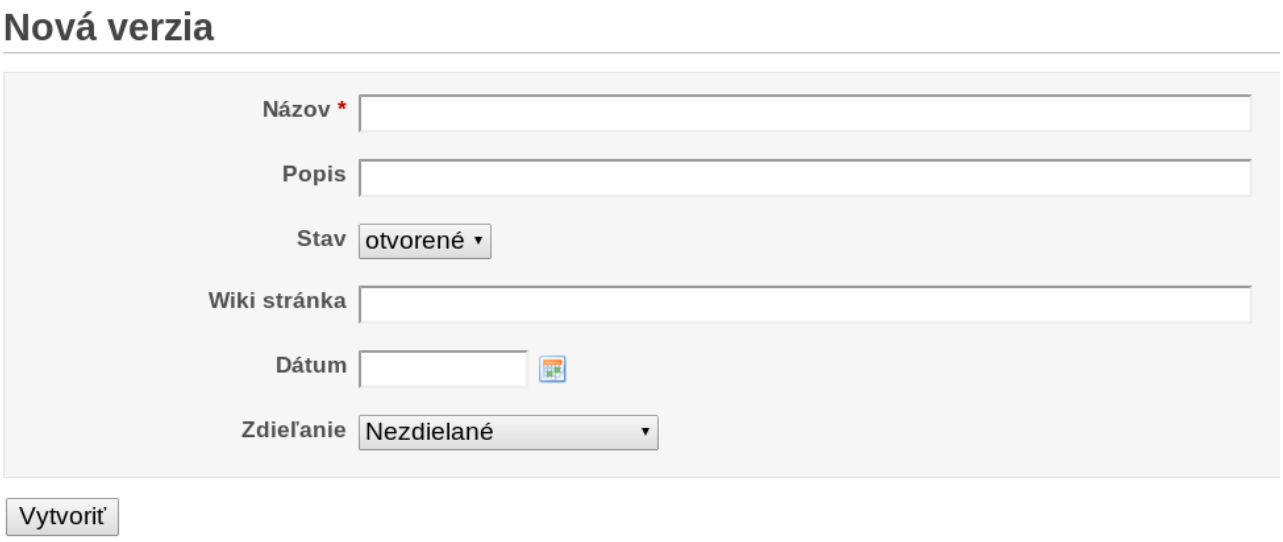
12.3.7 VLOŽENIE POUŽÍVATEĽSKÉHO PRÍBEHU DO NÁSTROJA REDMINE

Online nástroj na manažovanie projektov Redmine používame cez webové rozhranie, preto aj nasledujúci návod popisuje prácu s webovým rozhraním pri pridávaní nových používateľských príbehov a ich aktualizácii.

12.3.7.1 VYTVORENIE VERZIE PRE POUŽÍVATEĽSKÉ PRÍBEHY

Potrebné vykonať práve raz.

1. Otvorenie karty „Projekty“
2. Výber projektu
3. Otvorenie karty „Nastavenia“
4. Otvorenie karty „Verzie“
5. Kliknúť na „Nová verzia“
6. Vyplnenie formulára – Obrázok 2:
7. Pole „Názov“ vyplniť na „*Backlog*“
8. Pole „Popis“ vyplniť na „Reprezentuje *product backlog* (*user stories, features a task*s).“
9. Pole „Stav“ nastaviť na „Otvorené“
10. Pole „Wiki stránka“ nevypĺňať
11. Pole „Dátum“ vyplniť na aktuálny dátum
12. Pole „Zdieľanie“ nastaviť na „Nezdieľané“
13. Vytvorenie verzie tlačidlom „Vytvoriť“

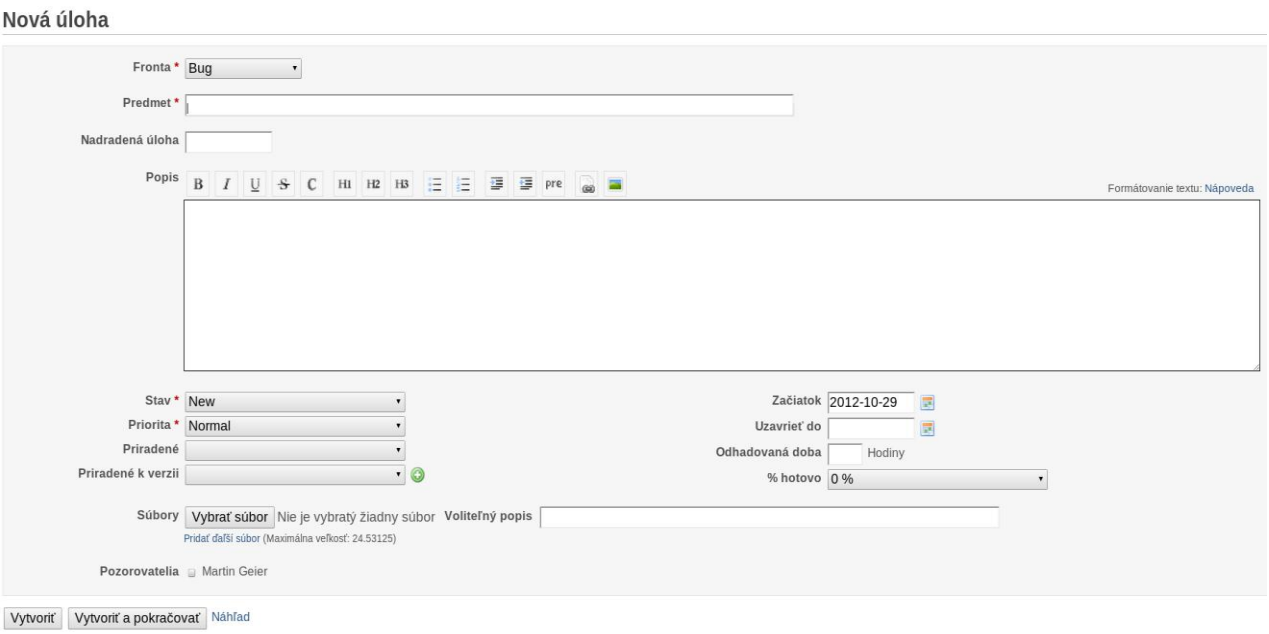


Obrázok 4: Formulár vytvorenia novej verzie

12.3.7.2 VLOŽENIE NOVÉHO POUŽÍVATEĽSKÉHO PRÍBEHU

Opis vloženia nového používateľského príbehu:

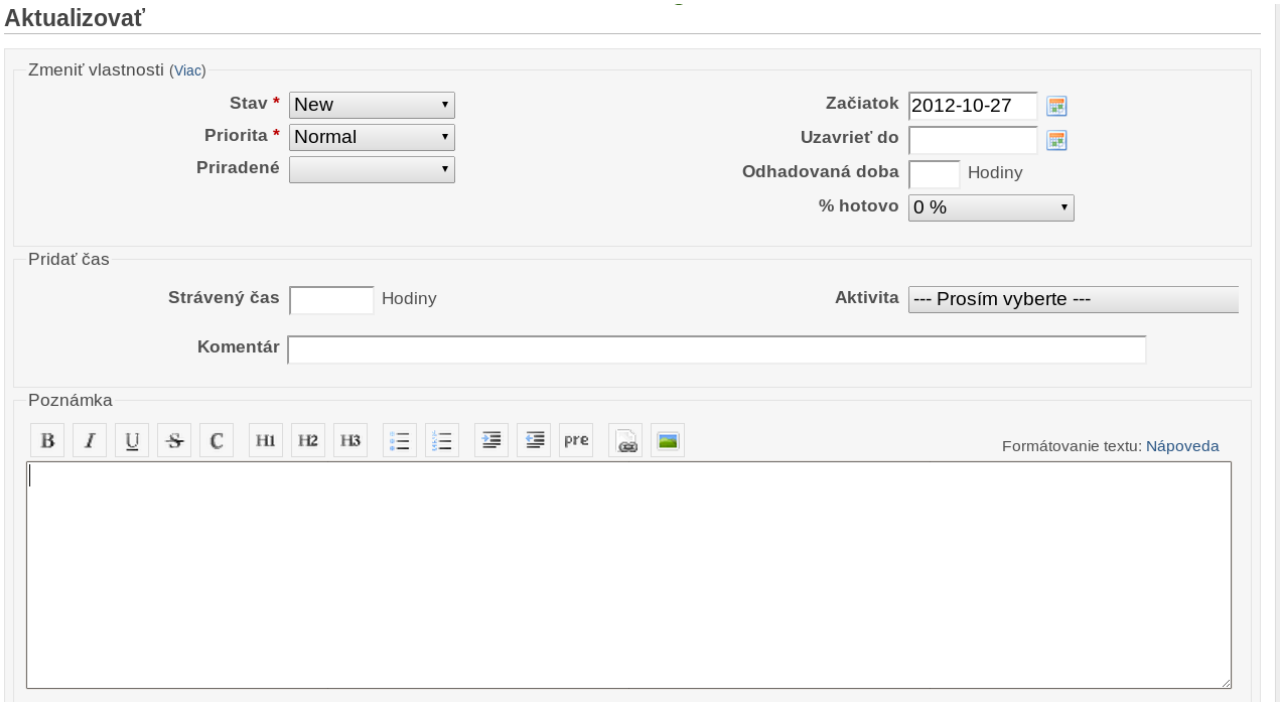
1. Otvorenie karty „Projekty“
2. Výber projektu
3. Otvorenie karty „Nová úloha“
4. Vyplnenie formulára – Obrázok 3:
   - Pole „Fronta“ zmeniť na „Idea“
   - Pole „Predmet“ vyplniť na názov používateľského príbehu vo formáte „Ja ako <roľa>, chcem <cieľ>, aby som mohol <dôvod>“, časť <dôvod> nie je povinná
   - Pole „Popis“ vyplniť na:
     - Ak existujú funkcionálne upresnenie, do riadku zapísať výraz „Funkcionálne upresnenie: “, štýlom „bold“, na nasledujúce riadky jednotlivé body, každý na nový riadok štýlom „Zoznam“
     - Ak existujú nefunkcionálne požiadavky, do riadku zapísať „Nefunkcionálne požiadavky: “, štýlom „bold“, na nasledujúce riadky jednotlivé body, každý na nový riadok štýlom „Zoznam“
   - Pole „Stav“ zmeniť na „New“
   - Pole „Priorita“ vyplniť podľa priority z dokumentu ktorú zadal produktový vlastník pri udávaní funkcionálnych požiadaviek
   - Pole „Priradené“ nechať nevyplnené
   - Pole „Priradené k verzii“ zmeniť na „Backlog“
   - Pole „Začiatok“ vyplniť na aktuálny dátum
   - Pole „Uzavrieť do“ nevypĺňať
   - Pole „Odhadovaná doba“ zmeniť podľa dátumu vyžadovaného produktovým vlastníkom
   - Pole „Hotovo“ nastaviť na „0 %“
5. Potvrdiť vytvorenie úlohy tlačidlom „Vytvoriť“



Obrázok 5: Formulár vytvorenia novej úlohy

12.3.7.3 ZMENA STAVU POUŽÍVATEĽSKÉHO PRÍBEHU

1. Otvorenie karty „Projekty“
2. Výber projektu
3. Otvorenie karty „Úlohy“
4. Výber úlohy ktorej stav ideme meniť
5. Vyvolanie aktualizácie úlohy kliknutím na tlačidlo „Aktualizovať“
6. Zmena stavu úlohy – Obrázok 4:
   - Úloha bola vyriešená
     - Pole „Stav“ nastaviť na „Resolved“
     - Pole „% hotovo“ nastaviť na „100 %“
     - Pole Strávený čas vyplniť podľa času práce stráveného na úlohe od posledného zmenu stavu
     - Pole „Poznámka“ vyplniť podla informácií ktoré mohli vzniknúť pri riešení úlohy a je predpoklad ich riešenia v budúcnosti
   - V úlohe bola vyriešená konkrétna časť
     - Pole „% hotovo“ nastaviť na percentuálnu časť podľa množstva vyriešených podúloh zo všetkých úloh prislúchajúcich danému používateľskému príbehu
     - Pole „Strávený čas“ vyplniť podľa času stráveného na úlohe od poslednej zmene stavu
     - Pole „Stav“ zmeniť na „In progress“
7. Uložiť zmenu stavu tlačidlom „Potvrdiť“



Obrázok 6: Formulár aktualizovania úlohy

12.3.7.4 ZMENA FUNKCIONÁLNYCH UPRESNENÍ ALEBO NEFUNKCIONÁLNYCH POŽIADAVIEK

Postup zmeny obsahu používateľského príbehu.

1. Otvorenie karty „Projekty“
2. Výber projektu
3. Otvorenie karty „Úlohy“
4. Výber úlohy ktorú chceme upraviť
5. Stlačiť tlačidlo Duplikovať
6. Vyplniť zmenené polia podla predchádzajúcich pravidiel
7. Uložiť upravenú úlohu tlačidlom „Vytvoriť“
8. Vybrať pôvodnú úlohu
9. Pole „Stav“ zmeniť na „*Rejected*“
10. Uložiť zmenu stavu tlačidlom „Potvrdiť“

12.4 Plánovanie šprintu (SCRUM) a jeho evidencia v Nástroji Redmine

12.4.1 Úvod

Cieľom metodiky je definovať postup plánovania šprintu, ktorý je súčasťou agilnej metódy vývoja softvéru nazývanej Scrum. Metodika podrobne opisuje pridanie nových úloh v nástroji Redmine, ktorý slúži pre manažment projektu a jednotlivých úloh. Táto metodika je určená pre tímy, ktoré vyvíjajú prostredníctvom agilnej metódy Srcum.

12.4.2 Pojmy

Pojem	**Vysvetlenie**
SCRUM	Agilná metóda vývoja softvéru.
Backlog projektu	Zoznam požiadaviek zoradený podľa priorít vlastníka produktu.
Backlog šprintu	Zoznam požiadaviek vybraných z backlogu projektu. Tieto požiadavky tím rieši v aktuálnom šprinte.
Šprint	Základná jednotka metódy SCRUM; Časová perióda, počas ktorej musí byť dokončená stanovená práca a pripravená na zhodnotenie.
Redmine	Webová aplikácia pre flexibilný projektový manažment.
Plánovacie pokrové karty (Planning poker cards)	Karty, na základe ktorých tím odhaduje čas potrebný na dokončenie úloh v šprinte.

12.4.3 Role a zodpovednosti

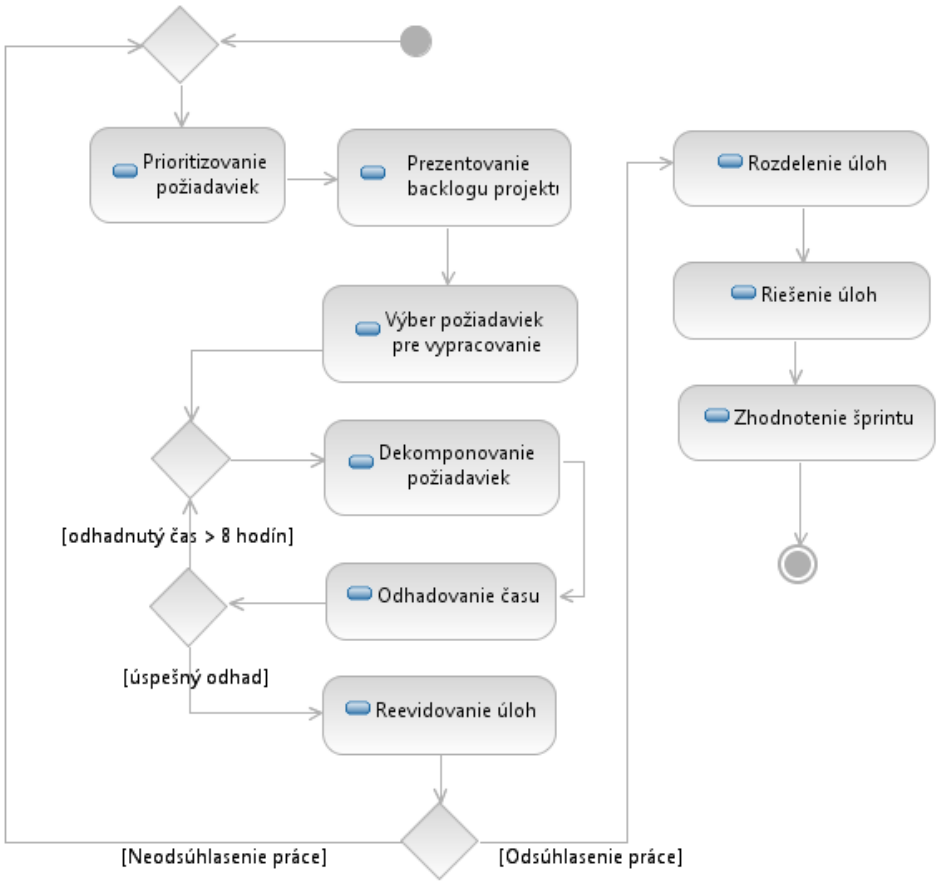
Rola	**Proces**	**Zodpovednosť**
Vlastník produktu (Product owner)	Prioritizácia požiadaviek	Stanovenie priorít práce na základe biznis hodnoty
	Prezentácia backlogu	Odprezentovanie backlogu pre pochopenie
	Výber úloh pre vypracovanie	Odsúhlasenie výberu požiadaviek Schopnosť robiť aj ťažké rozhodnutia
	Reevidácia úloh	Posúdenie časového harmonogramu z hľadiska biznis hodnoty
Vedúci tímu (Scrum master)	Výber úloh pre vypracovanie	Poradca vo výbere
	Dekompozícia požiadaviek Odhadovanie času	Prideľovanie úloh Vedenie tímu Moderovanie stretnutia

	Rozdelenie úloh	
	Zhodnotenie šprintu	Evidencia priebehu šprintu
Tím	Dekompozícia požiadaviek Odhadovanie času	Správne usudzovanie Zmysel pre detail
	Rozdelenie úloh	Výber úlohy podľa svojej schopnosti
	Zhodnotenie šprintu	Zhodnotenie priebehu, identifikácia možných prekážok, zlepšenie do budúcna
Manažér podpory vývoja	Rozdelenie úloh	Manažovanie projektových úloh
Jednotlivec tímu	Riešenie úloh	Kvalita práce Priebežné informovanie o postupe

12.4.4 PROCES PLÁNOVANIA ŠPRINTU

V tejto kapitole je opísaný proces plánovania šprintu, ktorý sa vykonáva vždy pred začiatkom každého šprintu.

č.	**Krok**	**Časť**
1	Prioritizácia požiadaviek	4.1
2	Prezentácia backlogu projektu	4.2
3	Výber úloh pre vypracovanie	4.3
4	Dekompozícia požiadaviek	4.4
5	Odhadovanie času	4.5
6	Reevidácia úloh	4.6
7	Rozdelenie úloh	4.7
8	Riešenie úloh	4.8
9	Zhodnotenie šprintu	4.9



Obr. 1: Proces plánovania šprintu

12.4.4.1 PRIORITIZÁCIA POŽIADAVIEK

Vstup: Backlog projektu
Výstup: Prioritizovaný zoznam požiadaviek
Zodpovedný: Vlastník produktu

Vlastník produktu si prejde backlog a usporiada požiadavky podľa biznis hodnoty. Požiadavky budú usporiadané tak, že požiadavky s najväčšou prioritou budú na vrchu zoznamu a postupne bude priorita klesať.

12.4.4.2 PREZENTÁCIA BACKLOGU PROJEKTU

Vstup: Backlog projektu
Výstup: Oboznámenie tímu s dôležitými požiadavkami
Zodpovedný: Vlastník produktu

Na začiatku šprint mítingu vlastník produktu prezentuje jednotlivé požiadavky s najväčšou prioritou a tie, ktoré chce aby boli riešené na najbližšom šprinte a zároveň vysvetľuje, ako vidí, že pracujú z funkcionálneho hľadiska. Tím si urobí prehľad a porozmýšľa o ich zložitosti, prípadných prekážkach a prichystá si otázky, ktoré lepšie identifikujú ich zámer.

12.4.4.3 VÝBER ÚLOH PRE VYPRACOVANIE

Vstup: Prioritizovaný zoznam požiadaviek
Výstup: Backlog šprintu
Zodpovedný: Vlastník produktu, Vedúci tímu

Po prezentácií sa vlastník produktu spolu s vedúcim tímu dohodne a navrhne požiadavky, ktoré musí tím vypracovať v nasledujúcom šprinte. Začne diskusia, kde tím na základe pripravených otázok identifikuje bližší zámer a možné prekážky jednotlivých požiadaviek a urobí predbežný odhad času, na základe ktorého vyberie toľko požiadaviek, aby ich stihol počas šprintu dokončiť. Požiadavky, ktoré tím vyberie, presunie na backlog šprintu. Následne na základe diskusie produktový vlastník odsúhlasí vybrané úlohy a tím začne pracovať na dekompozícií.

12.4.4.4 DEKOMPOZÍCIA POŽIADAVIEK

Vstup: Backlog šprintu
Výstup: Jednotlivé úlohy na vypracovanie
Zodpovedný: Vedúci tímu, Tím

Vedúci tímu spolu s tímom začnú diskutovať o jednotlivých požiadavkách z backlogu šprintu. Tím začne pracovať z poznámkami, ktoré získal už počas procesu prezentácie a výberu požiadaviek pre šprint backlog. Počas diskusie vedúci tímu spisuje všetky dôležité veci na tabuľu pre lepší prehľad nad diskutovaným problémom. Identifikované úlohy vedúci tímu spíše pod seba.

12.4.4.5 ODHADOVANIE ČAS

Vstup: Jednotlivé úlohy
Výstup: Časové odhady pre jednotlivé úlohy
Zodpovedný: Vedúci tímu, Tím

Vedúci tímu rozdá každému členovi karty plánovacieho pokru. Postupne prečíta úlohy, ktoré vznikli na základe dekompozície požiadaviek. Tím pre každú úlohu odhaduje čas potrebný na vypracovanie. Po každom odhade je tím povinný diskutovať odhadnutý čas a najmä ak sa odhady príliš líšia. V tomto prípade po diskusii prebehne druhé kolo odhadu. Ak je po úspešnom odhadovaní odhad väčší ako 8 hodín na úlohu, je potrebné ju dekomponovať na menšie časti. Po úspešnom časovom odhade vedúci tímu priradí čas jednotlivým úlohám na tabuli. Následne vedúci tímu spolu s tímom určí prioritu úloh, ktoré je potrebné v nasledujúcom období vypracovať.

12.4.4.6 REEVIDÁCIA ÚLOH

Vstup: Jednotlivé úlohy
Výstup: Odsúhlasenie práce pre šprint
Zodpovedný: Vedúci tímu, Vlastník produktu

Vedúci tímu prichystá pre vlastníka produktu stručnú správu, v ktorej ho informuje časovom harmonograme nadchádzajúcej práce a úlohách, ktoré sa budú riešiť. Následne vedúci tímu vlastníkovi produktu túto správu odprezentuje a vlastník produktu odsúhlasí prácu alebo zmení požiadavky pre vypracovanie.

12.4.4.7 ROZDELENIE ÚLOH

Vstup: Odsúhlasené úlohy a ich časové odhady
Výstup: Pridelené úlohy
Zodpovedný: Vedúci tímu, tím, Manažér podpory vývoja

Po odsúhlasení vedúci tímu rozbehne diskusiu o rozdelení úloh. Dohodne sa s každým členom tímu, ktorú úlohu mu pridelí. Po odsúhlasení všetkých členov s pridelenou/ pridelenými úlohami poverí manažéra podpory vývoja, aby zaevidoval pridelené úlohy do systému. Tento krok je podrobne opísaný v kapitole 5.1.

12.4.4.8 RIEŠENIE ÚLOH

Vstup: Pridelené úlohy
Výstup: Postup pri vypracovávaní úloh
Zodpovedný: Jednotlivý člen tímu

Každý člen tímu je povinný pracovať na pridelenej úlohe a zároveň je povinný priebežne zaznamenávať stav riešenia úlohy do systému. Tento krok je bližšie opísaný v kapitole 5.2.

12.4.4.9 ZHODNOTENIE ŠPRINTU

Vstup: Priebeh šprintu
Výstup: Zaznamenané zhodnotenie šprintu
Zodpovedný: Vedúci tímu, tím

Po skončení šprintu sa zvolá míting, kde vedúci tímu a tím diskutujú o priebehu šprintu, dokončených prípadne nedokončených úlohách, prekážkach, problémoch. Počas diskusie si vedúci tímu robí poznámky a zaznamenáva tak priebeh každého šprintu.

12.4.5 EVIDENCIA ÚLOH V SYSTÉME REDMINE (NIŽŠIA ÚROVEŇ)

V tejto kapitole je popísaný presný postup, ako pridať úlohu a zmeniť riešený stav v systéme Redmine.

č.	Krok	Časť
1	Pridelenie úlohy	5.1
2	Zmena stavu riešenej úlohy	5.2

12.4.5.1 PRIDELENIE ÚLOHY

Vstup: Úloha z backlog šprintu

Výstup: pridelená úloha v systéme Redmine

Zodpovedný: manažér podpory vývoja

Úlohy je povinný pridať do systému manažér podpory vývoja ihneď po skončení šprint mítingu. Pridelenie úlohy prebieha podľa nasledujúcich krokov:

1. Prihlásiť sa do systému Redmine
2. Kliknúť na kolónku *Project*
3. Vybrať projekt kliknutím na jeho názov
4. Kliknúť na záložku *New issue*
5. Vyplniť formulár pre zadanie novej úlohy

5.1. V kolónke *Tracker* vybrať možnosť *Task*, keďže ide o pridanie novej úlohy, ktorú bude riešiť daný člen tímu

5.2. Do zadávacieho poľa *Subjet* zadať názov úlohy

5.3. Ak zadávaná úloha je podúlohou inej úlohy, vyplniť položku *Parent task.* Do kolónky sa zadá *ID* nadúlohy.

5.4. V kolónke *description* vyplniť popis úlohy v nasledujúcom tvare:

Vstup:
Výstup:
Popis:
Poznámka:

Položky vstup a výstup sa vyplnia len v prípade, že úloha má taký charakter. Popis sa vyplní vždy a poznámka v prípade, že na mítingu boli ohľadom úlohy spomenuté nejaké poznámky, prípadne pripomienky.

5.5. V kolónke *status* sa nechať hodnotu *new,* keďže ide o novú úlohu

5.6. V kolónke *Priority* nastaviť prioritu úlohy. Priorita sa nastaví podľa toho, ako bola zvolená pri odhadovaní času. Priorita sa volí na základe faktorov ako sú urgentnosť úlohy, náväznosť na inú úlohu a obtiažnosť

5.7. V kolónke *Assignee* vybrať člena tímu, pre ktorého je úloha určená

5.8. V kolónke *Target* vybrať šprint, do ktorého úloha patrí

5.9. V kolónke *Start date* vyplniť dátum začatia plnenia úlohy

5.10. V kolónke *Due date* vyplniť dátum konca plnenia úlohy

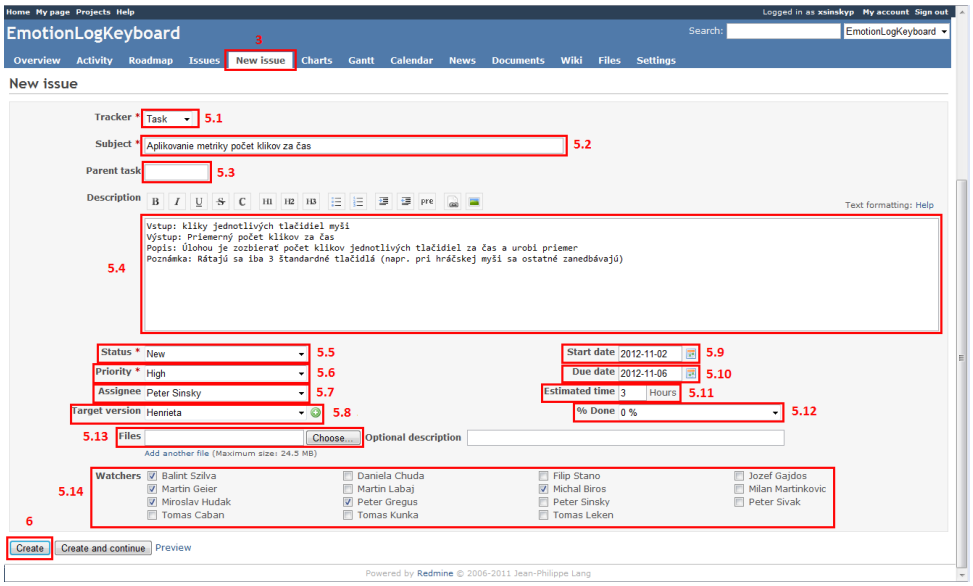
5.11. V kolónke *Estimated timevyplniť* predpokladaný čas. Tento čas bol odhadnutý počas procesu odhadovanie času.

5.12. V kolónke *% Done* nechať nastavenú hodnotu 0%, keďže sa na úlohe ešte nepracovalo

5.13. Pridať prílohy v kolónke *Files*, ak sú nejaké k dispozícií

5.14. Nastaviť osoby, ktoré môžu sledovať postup úlohy. Pomocou označovacieho políčka vybrať osoby.

6. Vytvoriť úlohu kliknutím na tlačítko *Create*



12.4.5.2 ZMENA STAVU RIEŠENEJ ÚLOHY

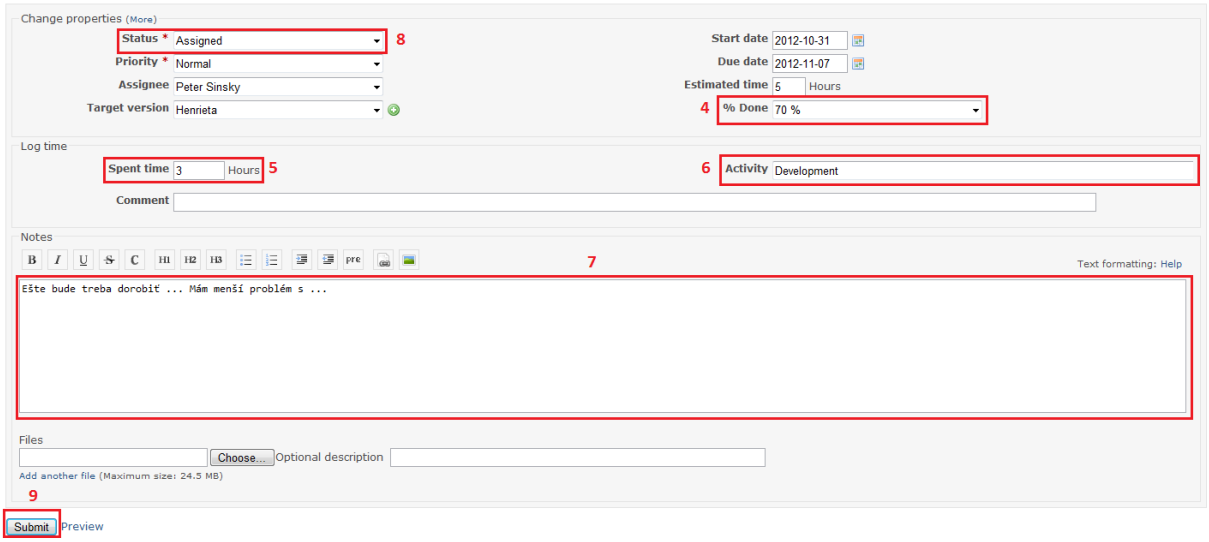
Vstup: Riešená úloha

Výstup: Upravený stav riešenia

Zodpovedný: Člen tímu

Každý člen tímu je povinný pravidelne aktualizovať stav riešenej úlohy vždy, ak zaznamenal pokrok. Tento stav aktualizuje nasledovným postupom:

1. Prihlásiť sa do systému Redmine
2. Vybrať úlohu kliknutím na jej názov
3. Kliknúť na položku *Update*
4. V kolónke *%Done* zmeniť percentuálny odhad vyjadrujúci postup práce. Odhad sa robí podľa vlastného uváženia.
5. V kolónke *Spent time* zadať počet hodín odpracovaných na plnenej úlohe
6. V kolónke *Activity* vybrať aktivitu, ktorá bola/je vykonávaná na úlohe
7. Do kolónky *Notes* je každý člen tímu povinný vždy napísať krátke zhodnotenie aktuálne riešenej úlohy.
8. V prípade, že je úloha vyriešená, v kolónke *Status* nastaviť hodnotu *Resolved*
9. Uložiť zmenu kliknutím na tlačítko *Submit*


Change properties (More)
Status * Assigned
8
Priority * Normal
Assignee Peter Sinsky
Target version Henrieta
Start date 2012-10-31
Due date 2012-11-07
Estimated time 5 Hours
4
% Done 70 %
Log time
Spent time 3 Hours
5
6
Activity Development
Comment
Notes
7
Text formatting: Help
Ešte bude treba dorobiť ... Mám menší problém s ...
Files
Choose...
Optional description
Add another file (Maximum size: 24.5 MB)
9
Submit
Preview

12.5 MANAŽMENT ZBERU POŽIADAVIEK VLOŽENIE POUŽÍVATEĽSKÉHO PRÍBEHU DO NÁSTROJA REDMINE

12.5.1 ÚVOD

Metodika má za úlohu informovať čitateľa o postupoch pri zbere požiadaviek a definovať úlohy pre jednotlivých členov týmu, ktoré sú spojené s procesom zberu požiadaviek v agilnom spôsobe vývoja SCRUM.

Druhá časť dokumentu popisuje spôsob zadávanie požiadaviek vo forme používateľských príbehov do online nástroja Redmine. Súčasťou tejto časti je aj aktualizácia a zmena stavu používateľského príbehu.

12.5.2 DEFINOVANIE POJMOV

- Produktový vlastník
  Zastupuje stakeholderov. Blízko spolupracuje so zákazníkom a vývojovým tímom. Musí rozumieť problémovej oblasti.
- *Scrum master*
  Člen tímu, odstraňuje problémy spôsobené vonkajšími vplyvmi, motivuje členov tímu k lepším výkonom, upokojuje potýčky medzi členmi tímu.
- Používateľský príbeh (User story)
  Úloha zapísané v tvare „Ako <rola>, chcem <cieľ> aby som mohol <dôvod>“. Udáva nielen čo sa má spraviť, ale aj kto to bude používať a na čo je to dobré.
- Redmine
  Open source webový nástroj slúžiaci na manažment projektov. Zahŕňa Grantove diagramy, kalendár, wiki, fórum, role, emailové notifikácie a ďalšie.

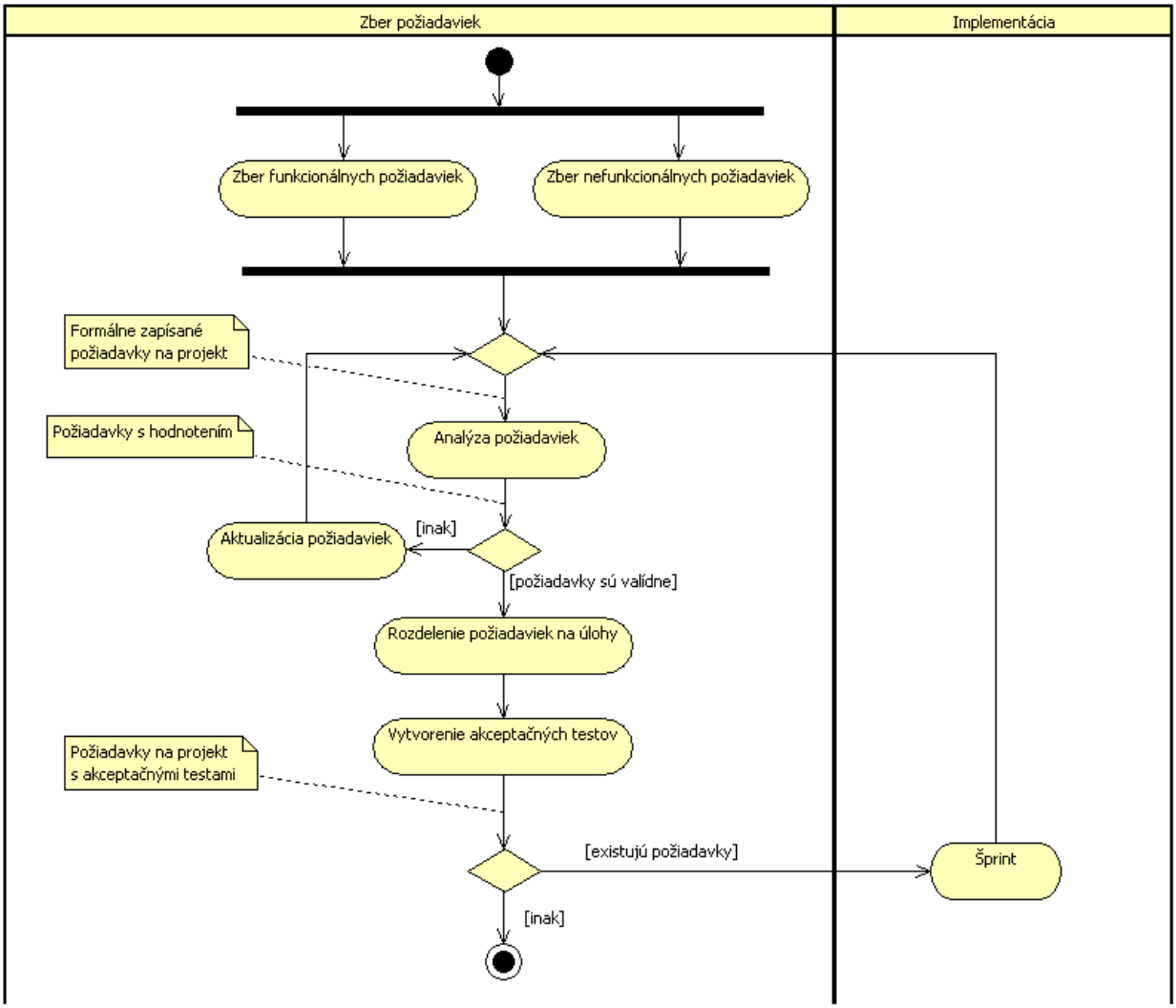
12.5.3 ROLE A ZODPOVEDNOSTI

Rola	Proces	Zodpovednosť
Produktový vlastník	Zber funckionálnych a nefunkcionálnych požiadaviek	Vytvorenie používateľských príbehov
Produktový vlastník	Analýza požiadaviek	Preskúmanie aktuálneho stavu požiadaviek

Produktový vlastník	Aktualizácia požiadaviek	Úprava požiadaviek na projekt
Radový používateľ	Zber funckionálnych a nefunkcionálnych požiadaviek	Pomoc s vytvorením používateľských príbehov
Scrum master	Analýza požiadaviek	Oboznámenie sa s problémovými časťami dokumentu
Scrum master	Aktualizácia požiadaviek	Pomoc s opravou požiadaviek
Analytik	Analýza požiadaviek	Napísanie hodnotenia požiadaviek
Výkonný manažment	Vytvorenie akceptačných testov	Vytvorenie scenárov testov
Tester	Vytvorenie akceptačných testov	Vytvorenie scenárov testov
Tím	Rozdelenie požiadaviek na úlohy	Rozdelenie používateľských príbehov na úlohy
Manažér podpory vývoja	Rozdelenie požiadaviek na úlohy	Vloženie alebo aktualizácia požiadaviek

Tabuľka č. 1: role a zodpovednosti

12.5.4 GRAFICKÁ REPREZENTÁCIA PROCESU ZBERU POŽIADAVIEK



Obrázok 7: Aktivity diagram zberu požiadaviek

12.5.5 PROCES ZBERU POŽIADAVIEK

	Krok	Kapitola
1.	Zber funkcionálnych požiadaviek	5.1
2.	Zber nefunkcionálnych požiadaviek	5.2
3.	Analýza požiadaviek	5.3
4.	Rozdelenie požiadaviek na úlohy	5.4
5.	Aktualizácia požiadaviek	5.5
6.	Vytvorenie akceptačných testov	5.6

Tabuľka č. 2: Proces zberu požiadaviek

12.5.5.1 ZBER FUNKCIONÁLNYCH POŽIADAVIEK

Vstup: Ústny opis požiadaviek na projekt

Výstup: Formálne zapísané požiadavky na projekt

Zodpovedný: Produktový vlastník, radový používatelia

Požiadavky, ktoré vysloví produktový vlastník budú zapísané vo forme používateľských príbehov. Ak produktový vlastník neuvedie časť „dôvod“, táto časť bude vynechaná zo zápisu. Používateľské príbehy budú zapísané v dokumente a budú zvýraznené štýlom bold. Zákazník zadá dôležitosť používateľského príbehu v rozsahu hodnotenia:

- nízka,
- normálna,
- vysoká,
- urgentná,
- bezprostredná

a následne uvedie dátum kedy je vyžadované ukončenie používateľského príbehu. Dátum a hodnotenie bude zapísané v prvom riadku za názvom používateľského príbehu. Za týmito informáciami budú nasledovať funkcionálne upresnenie, ak takéto upresnenia existujú, a bude im predchádza riadok s textom „Funkcionálne upresnenia:“. Za týmto riadkom budú zapísané upresnenia v neočíslovanom zozname, každé na novom riadku. Pred každým ďalším používateľským príbehom bude voľný jeden riadok.

12.5.5.2 Zber nefunkcionálnych požiadaviek

Vstup: Ústny opis nefunkcionálnych požiadaviek na projekt

Výstup: Formálne zapísané požiadavky na projekt

Zodpovedný: Produktový vlastník, rádový používatelia

Každý používateľský príbeh môže byť doplnený o nefunkcionálne požiadavky. O potrebe doplnenia nefunkcionálnych požiadaviek rozhoduje produktový vlastník. Nefunkcionálne požiadavky budú doplnené za funkcionálne upresnenia a bude im predchádzať riadok s textom „Nefunkcionálne požiadavky“. Požiadavky budú zapísané v neočíslovanom zozname, každá na novom riadku.

12.5.5.3 Analýza požiadaviek

Vstup: Formálne zapísané požiadavky na projekt

Výstup: Formálne zapísané požiadavky na projekt s hodnotením požiadaviek

Zodpovedný: Produktový vlastník, *Scrum master*, Analytik

Scrum master spolu s analytikom zhodnotia správnosť zadaných používateľských príbehov, funkcionálnych upresnení a nefunkcionálnych požiadaviek. V dokumente bude pod nefunkcionálne požiadavky doplnený nový riadok s textom „Zhodnotenie: “, za ktorým na nových riadkoch budú štruktúrovane zapísané nedostatky požiadaviek. Ak žiadne nedostatky neexistujú, pod textom „Zhodnotenie:“ bude text „Schválené“ zvýraznené štýlom bold.

12.5.5.4 Rozdelenie požiadaviek na úlohy

Vstup: Formálne zapísané požiadavky na projekt s hodnotením požiadaviek

Výstup: Zapísané požiadavky a úlohy na projekt

Zodpovedný: Tím, Manažér podpory vývoja

Ak boli požiadavky schválené a vznikli nové požiadavky, alebo pôvodné požiadavky boli zmenené, tím rozdelí tieto požiadavky zapísané v tvare používateľských príbehov na úlohy. Súčasťou procesu je aj zanesenie nových používateľských príbehov, alebo zmenených používateľských príbehov do nástroja na manažment projektov podľa kapitoly 6 *Vloženie používateľských príbehov do nástroja Redmine*.

12.5.5.5 Aktualizácia požiadaviek

Vstup: Formálne zapísané požiadavky na projekt s hodnotením požiadaviek

Výstup: Formálne zapísané požiadavky na projekt

Zodpovedný: Produktový vlastník, *Scrum master*

Ak je potrebné zmeniť zadanie niektorých funkcionálnych alebo nefunkcionálnych požiadaviek v projekte, *scrum master* pomáha produktovému vlastníkovi s vysvetlením a opravením problémových častí. Pôvodný dokument sa archivuje a časť ktorá obsahovala zhodnotenie používateľského príbehu bude vymazaná. Spôsob zápisu opravených požiadaviek je rovnaký ako v bodoch 5.1 a 5.2

12.5.5.6 VYTVORENIE AKCEPTAČNÝCH TESTOV

Vstup: Formálne zapísané požiadavky na projekt

Výstup: Formálne zapísané akceptačné testy

Zodpovedný: Výkonný manažment, Tester

Výkonný manažment spolu s členom týmu zodpovedného za testovanie vytvoria scenáre testovania nových používateľských príbehov. Testy budú zapísané v dokumente. Štruktúra dokumentu bude nasledovná:

Používateľské príbehy zvýraznené štýlom bold, nasledované scenárom testovania, vstupmi a očakávanými výstupmi na nových riadkoch.

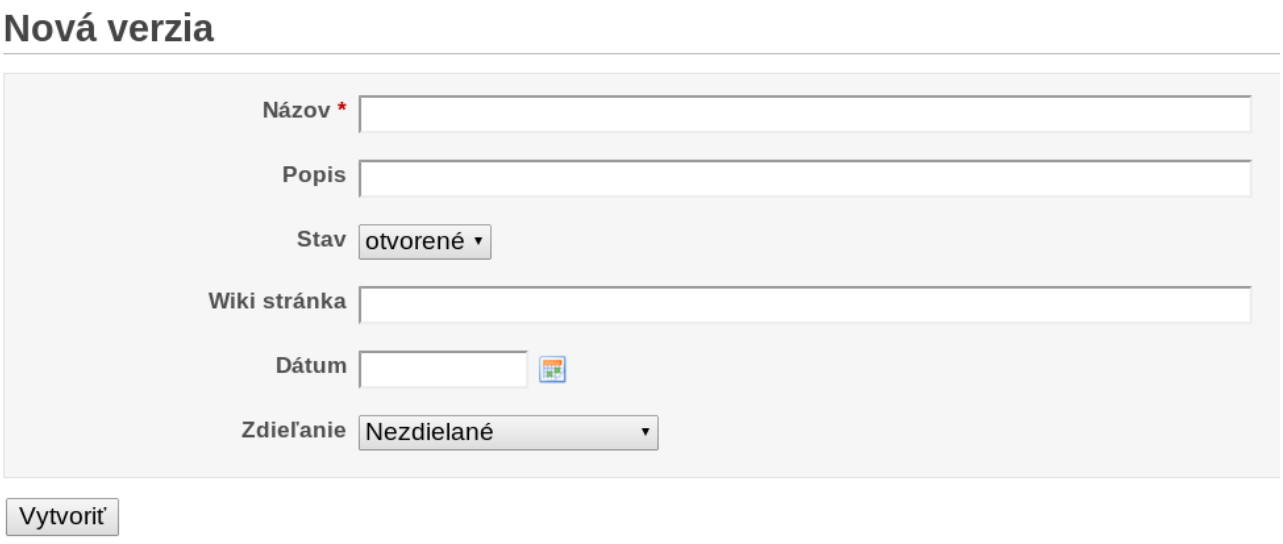
12.5.6 VLOŽENIE POUŽÍVATEĽSKÉHO PRÍBEHU DO NÁSTROJA REDMINE

Online nástroj na manažovanie projektov Redmine používame cez webové rozhranie, preto aj nasledujúci návod popisuje prácu s webovým rozhraním pri pridávaní nových používateľských príbehov a ich aktualizácii.

12.5.6.1 VYTVORENIE VERZIE PRE POUŽÍVATEĽSKÉ PRÍBEHY

Potrebné vykonať práve raz.

1. Otvorenie karty „Projekty“
2. Výber projektu
3. Otvorenie karty „Nastavenia“
4. Otvorenie karty „Verzie“
5. Kliknúť na „Nová verzia“
6. Vyplnenie formulára – Obrázok 2:
   - Pole „Názov“ vyplniť na „*Backlog*“
   - Pole „Popis“ vyplniť na „Reprezentuje *product backlog* (*user stories, features a tasks*).“
   - Pole „Stav“ nastaviť na „Otvorené“
   - Pole „Wiki stránka“ nevypĺňať
   - Pole „Dátum“ vyplniť na aktuálny dátum
   - Pole „Zdieľanie“ nastaviť na „Nezdieľané“
7. Vytvorenie verzie tlačidlom „Vytvoriť“

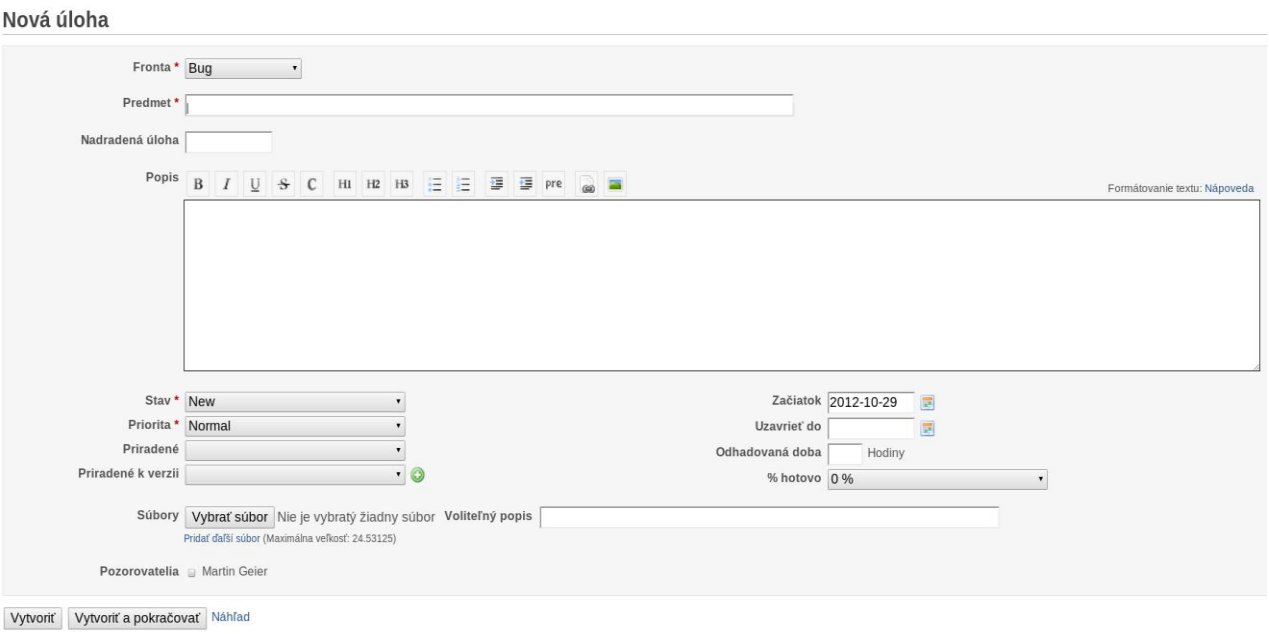


Obrázok 8: Formulár vytvorenia novej verzie

12.5.6.2 VLOŽENIE NOVÉHO POUŽÍVATEĽSKÉHO PRÍBEHU

Opis vloženia nového používateľského príbehu:

1. Otvorenie karty „Projekty“
2. Výber projektu
3. Otvorenie karty „Nová úloha“
4. Vyplnenie formulára – Obrázok 3:
   - Pole „Fronta“ zmeniť na „Idea“
   - Pole „Predmet“ vyplniť na názov používateľského príbehu vo formáte „Ja ako <roľa>, chcem <cieľ>, aby som mohol <dôvod>“, časť <dôvod> nie je povinná
   - Pole „Popis“ vyplniť na:
     - Ak existujú funkcionálne upresnenie, do riadku zapísať výraz „Funkcionálne upresnenie: “, štýlom „bold“, na nasledujúce riadky jednotlivé body, každý na nový riadok štýlom „Zoznam“
     - Ak existujú nefunkcionálne požiadavky, do riadku zapísať „Nefunkcionálne požiadavky: “, štýlom „bold“, na nasledujúce riadky jednotlivé body, každý na nový riadok štýlom „Zoznam“
   - Pole „Stav“ zmeniť na „New“
   - Pole „Priorita“ vyplniť podľa priority z dokumentu ktorú zadal produktový vlastník pri udávaní funkcionálnych požiadaviek
   - Pole „Priradené“ nechať nevyplnené
   - Pole „Priradené k verzii“ zmeniť na „Backlog“
   - Pole „Začiatok“ vyplniť na aktuálny dátum
   - Pole „Uzavrieť do“ nevypĺňať
   - Pole „Odhadovaná doba“ zmeniť podľa dátumu vyžadovaného produktovým vlastníkom
   - Pole „Hotovo“ nastaviť na „0 %“
5. Potvrdiť vytvorenie úlohy tlačidlom „Vytvoriť“



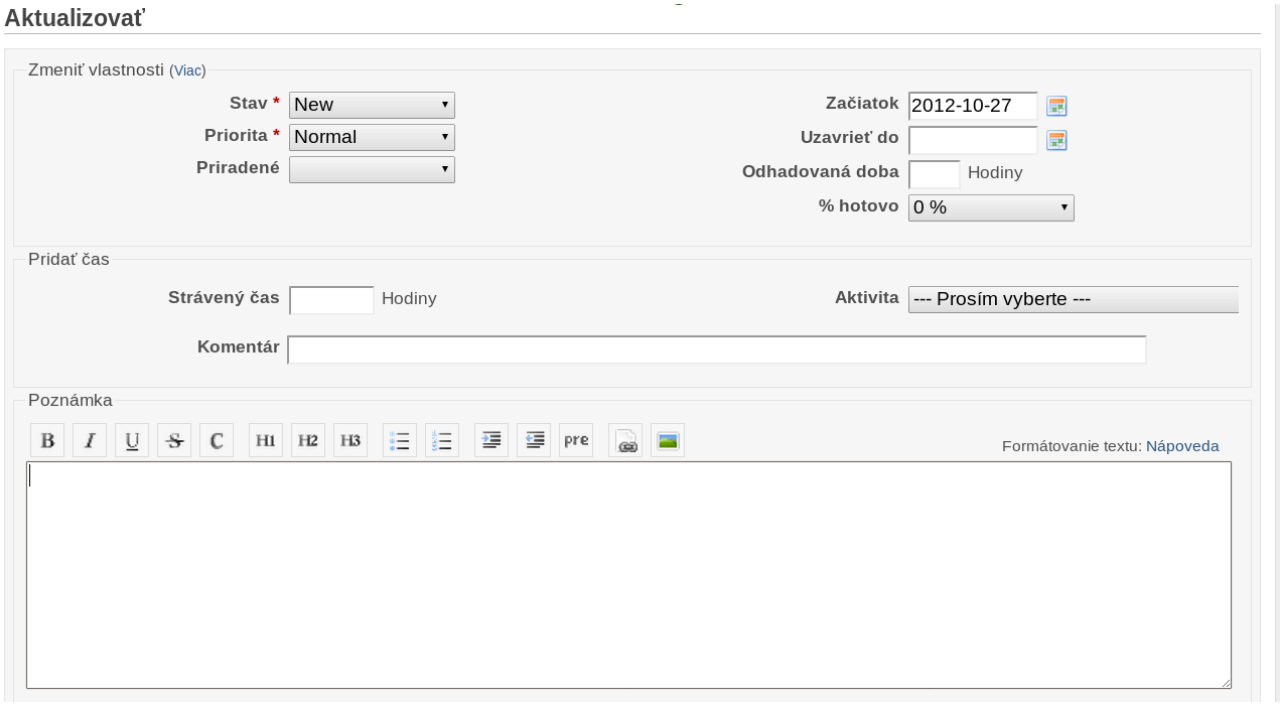
Obrázok 9: Formulár vytvorenia novej úlohy

12.5.6.3 ZMENA STAVU POUŽÍVATEĽSKÉHO PRÍBEHU

1. Otvorenie karty „Projekty“
2. Výber projektu
3. Otvorenie karty „Úlohy“
4. Výber úlohy ktorej stav ideme meniť
5. Vyvolanie aktualizácie úlohy kliknutím na tlačidlo „Aktualizovať“
6. Zmena stavu úlohy – Obrázok 4:

- Úloha bola vyriešená
  - Pole „Stav“ nastaviť na „Resolved“
  - Pole „% hotovo“ nastaviť na „100 %“
  - Pole Strávený čas vyplniť podľa času práce stráveného na úlohe od posledného zmenu stavu
  - Pole „Poznámka“ vyplniť podla informácií ktoré mohli vzniknúť pri riešení úlohy a je predpoklad ich riešenia v budúcnosti
- V úlohe bola vyriešená konkrétna časť
  - Pole „% hotovo“ nastaviť na percentuálnu časť podľa množstva vyriešených podúloh zo všetkých úloh prislúchajúcich danému používateľskému príbehu
  - Pole „Strávený čas“ vyplniť podľa času stráveného na úlohe od poslednej zmene stavu
  - Pole „Stav“ zmeniť na „In progress“

7. Uložiť zmenu stavu tlačidlom „Potvrdiť“



Obrázok 10: Formulár aktualizovania úlohy

12.6 Zmena funkcionálnych upresnení alebo nefunkcionálnych požiadaviek

Postup zmeny obsahu používateľského príbehu.

1. Otvorenie karty „Projekty“
2. Výber projektu
3. Otvorenie karty „Úlohy“
4. Výber úlohy ktorú chceme upraviť
5. Stlačiť tlačidlo Duplikovať
6. Vyplniť zmenené polia podla predchádzajúcich pravidiel
7. Uložiť upravenú úlohu tlačidlom „Vytvoriť“
8. Vybrať pôvodnú úlohu
9. Pole „Stav“ zmeniť na „*Rejected*“
10. Uložiť zmenu stavu tlačidlom „Potvrdiť“

12.7 METODIKA MANAŽMENTU ÚLOH

12.7.1 ÚVOD

Účelom tejto metodiky je zjednotiť a štandardizovať postup vytvárania a kontroly úloh stanovením jednotných pravidiel za účelom zvýšenia kvality a efektivity monitorovania projektu. Je určená najmä pre projekty, ktoré sa riadia metodikou Scrum.

Opisuje procesy vytvárania nových úloh, ich plánovanie a prideľovanie, následné riešenie a sledovanie stavu dokončenia a nakoniec vyhodnotenie.

Upravuje roly a zodpovednosti všetkých členov tímu v jednotlivých procesoch.

Na manažment úloh bude používaný nástroj Redmine.

12.7.2 SLOVNÍK POJMOV

- *úloha* – pozri kap. 3.1
- *Redmine* – flexibilná webová aplikácia pre účely manažmentu projektu
- *Scrum* – agilná metóda vývoja softvéru
- *Scrum master* – rola v tíme; osoba postavená medzi tímom a zákazníkom zodpovedná za komunikáciu
- *Product backlog* – súhrn všetkých vlastností a funkcií softvéru
- *User story* – funkcia alebo vlastnosť softvéru
- *Šprint* – časové obdobie, v ktorom prebehnú všetky procesy a vyhodnotenie

12.7.3 ÚLOHA

12.7.3.1 DEFINÍCIA

Pod pojmom úloha sa na účely tejto metodiky rozumie produkčná / výkonná aktivita, resp. určité množstvo práce s danými relevantnými vstupmi a cieleným výstupom.

Úloha má byť dostatočne krátka a jasne definovaná, aby bolo možné sledovať jej priebeh. Úroveň granularity má byť na čo najnižšej úrovni, aby sa predišlo viacznačnosti riešenia a nedorozumeniam.

Tabuľka 1 zobrazuje základné atribúty úlohy a ich popis.

TABUĽKA 2 - ATRIBÚTY ÚLOHY

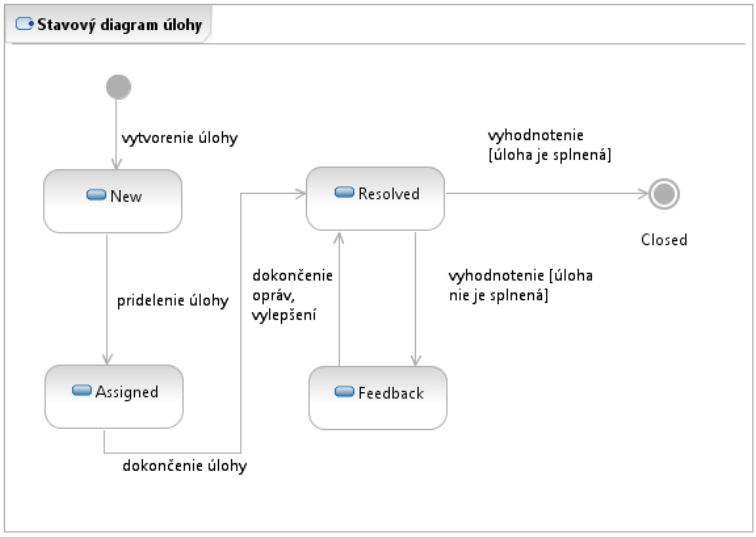
Úloha	
Atribút	Popis
Povinné:	
Predmet	krátke a výstižné pomenovanie úlohy
Opis	doplňujúci text, ktorý bližšie špecifikuje zadanie úlohy
Status	stav úlohy (pozri 12.7.3.2)
Priorita	úroveň naliehavosti vyriešenia úlohy (pozri 12.7.3.3)
Zodpovedná osoba	osoba zodpovedná za vyriešenie úlohy
Termín	dátum očakávaného vyriešenia úlohy
% dokončenia	percentuálny zápis stavu úlohy
Voliteľné:	
Cieľová verzia	označenie verzie výstupu (vlastné pomenovanie)
Nadradená úloha	úloha, ktorej vyriešenie je nutnou podmienkou pre riešenie tejto úlohy
Prílohy	doplňujúce materiály k riešeniu

12.7.3.2 STAVY

Úloha nadobúda nasledujúce stavy:

- *New* – vytvorená, nepridelená úloha
- *Assigned* – pridelená úloha zodpovednej osobe; na úlohe sa pracuje
- *Resolved* – úloha je vyriešená a čaká na vyhodnotenie
- *Feedback* – oprava, doplnenie, vylepšenie riešenia; na úlohe sa pracuje
- *Closed* – úloha je definitívne uzavretá

Obrázok 1 znázorňuje jednotlivé stavy úlohy a prechody medzi nimi.



OBRÁZOK 11 – STAVOVÝ DIAGRAM ÚLOHY

12.7.3.3 PRIORITA

Úloha nadobúda nasledujúce možnosti prioritizácie:

- *Low* – nízka priorita; doplňujúca úloha; môže sa presunúť do ďalšieho šprintu
- *Normal* – východzia hodnota, neutrálna
- *High* – úlohu je potrebné vyriešiť v aktuálnom šprinte
- *Urgent* – vyriešenie tejto úlohy je podmienkou pre riešenie iných úloh; úlohu je potrebné vyriešiť do najbližšieho stretnutia
- *Immediate* – najvyššia priorita; okamžité riešenie úlohy

12.7.4 ROLY A ZODPOVEDNOSŤ

Tabuľka 2 špecifikuje jednotlivé roly a ich zodpovednosti v príslušných procesoch.

TABUĽKA 3 - ROLY A ICH ZODPOVEDNOSTI

Rola	Zodpovednosť	Proces
scrum master	zvolenie user story	vytvorenie úloh
	prioritizácia úloh	plánovanie a rozvrhovanie úloh
	pridelenie úloh členom, kontrola rozloženia práce v tíme	prideľovanie úloh
tím (všetci členovia)	rozdelenie user story na úlohy, definovanie úloh	vytvorenie úloh
	odhadovanie času potrebného na riešenie úlohy	plánovanie a rozvrhovanie úloh

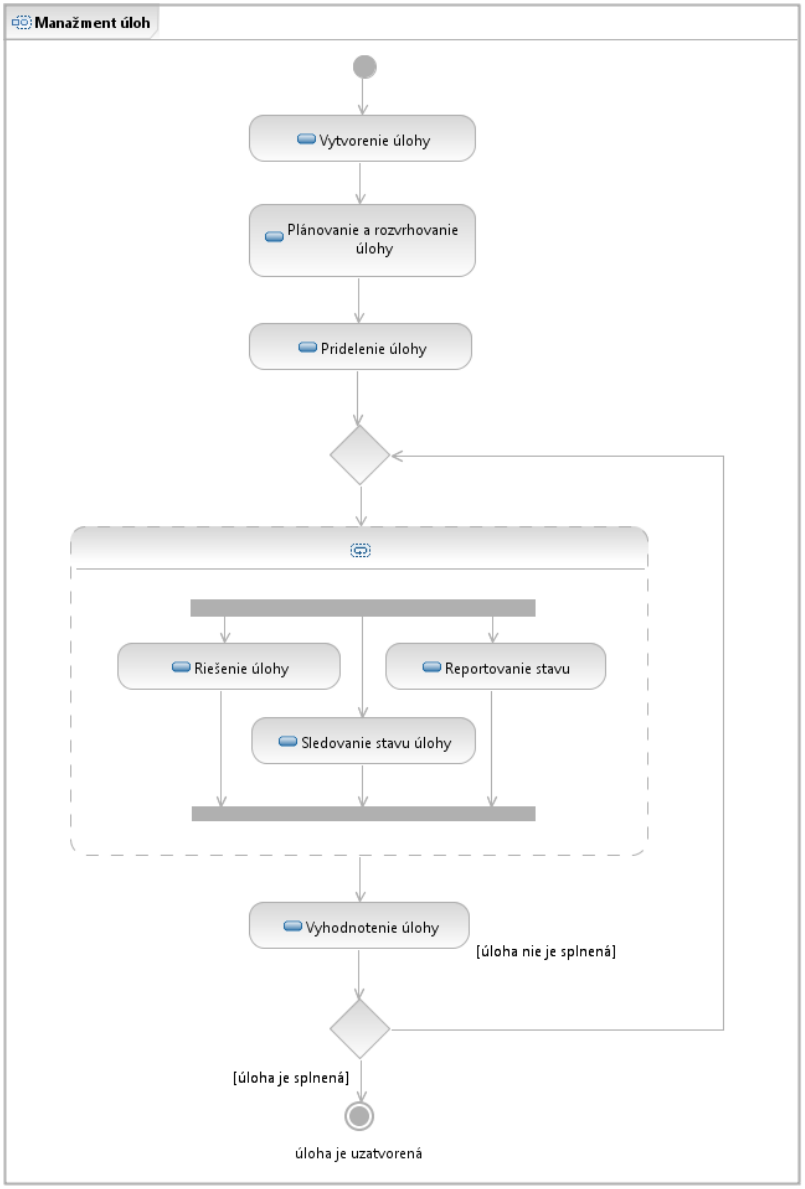
	rozdelenie úloh	prideľovanie úloh
manažér plánovania	predstavenie plánu šprintu, rozvrhovanie úloh	plánovanie a rozvrhovanie úloh
manažér monitorovania	monitorovanie práce, odhadovanie dokončenia, vytvorenie výslednej správy	sledovanie stavu úloh
vývojár	práca na pridelenej úlohe, aktualizovanie stavu dokončenia do manažovacieho systému	riešenie úloh
	prezentácia riešenia úlohy	vyhodnotenie úloh
tester	zhodnotenie riešenia úlohy	vyhodnotenie úloh
zapisovateľ	zapisovať a editovať úlohy v manažovacom systéme	plánovanie a rozvrhovanie úloh, prideľovanie úloh, vyhodnotenie úloh

12.7.5 PROCESY

Proces manažmentu úloh pozostáva z krokov zobrazených v tabuľke 3. Náväznosti medzi mini a ich vykonávanie v čase je znázornené na obrázku 2. Jednotlivé kroky sú opísané v príslušných kapitolách.

TABUĽKA 4 - PROCESY MANAŽMENTU ÚLOH

Krok	Proces	Kapitola
1	Vytvorenie úloh	12.7.5.1
2	Plánovanie a rozvrhovanie úloh	12.7.5.2
3	Pridelenie úloh	12.7.5.3
4	Riešenie úloh	12.7.5.4
5	Sledovanie stavu úloh	12.7.5.5
6	Vyhodnotenie úloh	0



OBRÁZOK 12 - DIAGRAM AKTIVÍT PRE MANAŽMENT ÚLOHY

12.7.5.1 VYTVORENIE ÚLOH

Vstup: user story

Výstup: zadefinované úlohy

Účastníci: tím, scrum master

Scrum master zvolí z product backlogu podľa priority user story, ktorý sa bude riešiť v aktuálnom šprinte. Tím rozdelí zvolený user story na jednotlivé úlohy a pre každú úlohu zadefinuje povinné atribúty:

- predmet – začína slovesom a obsahuje najviac 50 znakov
- opis – bližšie špecifikované detaily
- status – stav novej vytvorenej úlohy je New
- % dokončenia sa nastaví na nulu

Okrem povinných atribútov sa môžu podľa potreby definovať aj nepovinné.

Nasleduje proces plánovania a rozvrhovania úloh.

12.7.5.2 PLÁNOVANIE A ROZVRHOVANIE ÚLOH

Vstup: zadefinované úlohy

Výstup: ohodnotené úlohy

Účastníci: scrum master, manažér plánovania, tím, zapisovateľ

Manažér plánovania predstaví plán na aktuálny šprint. Na základe neho potom scrum master (manažér projektu) pridelí každej úlohe prioritu a vytvorí usporiadaný zoznam úloh podľa priority, pričom úlohy s najvyššou prioritou sú najvyššie.

Následne tím odhadne pre každú úlohu čas na jej vyriešenie. Manažér plánovania podľa odhadnutých časov vyberie tie úlohy, ktoré sa podarí vyriešiť v aktuálnom šprinte. Cieľom je vyriešiť všetky zadané úlohy pre aktuálny user story v aktuálnom šprinte. Ale ak sa kvôli rozvrhu musí nejaká úloha presunúť do ďalšieho šprintu, automaticky sa jej priraďuje najvyššia priorita.

Každej úlohe sa doplnia atribúty priorita a termín. Zapisovateľ zadá úlohy do manažovacieho systému.

Nasleduje proces pridelenia úloh.

12.7.5.3 PRIDELENIE ÚLOH

Vstup: ohodnotené úlohy

Výstup: pridelené úlohy zapísané v manažovacom systéme

Účastníci: scrum master, tím, zapisovateľ

Členovia tímu si rozdelia úlohy nasledovne: scrum master prechádza usporiadaným zoznamom úloh podľa priority a ku každej úlohe sa dobrovoľne prihlási niektorý člen. Ak sa prihlási viacero členov alebo nikto, tak člena poverí úlohou scrum master. Scrum master taktiež dohliada na rozloženie práce v tíme.

Každej úlohe sa doplní atribút zodpovedná osoba a upraví stav na Assigned. Zapisovateľ upraví stav úloh v manažovacom systéme.

12.7.5.4 RIEŠENIE ÚLOH

Vstup: pridelená úloha

Výstup: vyriešená úloha

Účastníci: vývojár

Vývojár (člen tímu) z plných síl svedomito pracuje na pridelenej úlohe. Zároveň pravidelne aktualizuje percentuálny stav dokončenia úlohy v manažovacom systéme. Po úspešnom dokončení úlohy upraví jej stav na Resolved.

Proces sa opakuje, kým úloha nie je vyriešená. Potom sa pokračuje vyhodnotením úlohy.

12.7.5.5 SLEDOVANIE STAVU ÚLOH

Vstup: úlohy

Výstup: správa o priebehu projektu

Účastníci: manažér monitorovania

Manažér monitorovania pravidelne kontroluje postup práce členov tímu na jednotlivých úlohách a porovnáva ich s plánom. V prípade nezrovnalostí kontaktuje scrum mastera a informuje ho o danej situácii. Po každom šprinte vyhotoví správu, v ktorej zhodnotí stav plnenia úloh. Správa musí obsahovať zoznam zadaných úloh pre šprint a ich percentuálne vyhodnotenie stavu riešenia. Podľa výsledkov v závere zhrnie, či projekt napreduje alebo mešká s plánom.

12.7.5.6 VYHODNOTENIE ÚLOH

Vstup: vyriešená úloha

Výstup A: uzavretá úloha

Výstup B: znovu riešená úloha

Účastníci: vývojár, testeri, zapisovateľ

Vývojár prezentuje testerom (tímu) riešenie svojej pridelenej úlohy. Testeri zhodnotia správnosť a úplnosť riešenia. Na základe záveru testerov sa postupuje:

A. ak úloha je splnená

Úlohe sa upraví stav na Closed. V tomto momente je už úloha definitívne uzavretá a nie je možné ju znova otvoriť. Ak sa neskôr zistia nedostatky, vytvorí sa nová úloha s rovnakým názvom doplneným o identifikátor verzie (#2, #3, #4, ...). Zapisovateľ upraví stav úlohy v manažovacom systéme.

B. ak úloha nie je splnená alebo sa zistili nové skutočnosti

Úlohe sa upraví stav na Feedback a je vrátená vývojárovi na opravu alebo doplnenie riešenia. Vývojár pracuje na úlohe. Zároveň pravidelne aktualizuje percentuálny stav dokončenia úlohy v manažovacom systéme. Po úspešnom dokončení úlohy upraví jej stav na Resolved.

12.7.6 APLIKOVANIE METODIKY V SYSTÉME REDMINE

V tejto časti sú popísané konkrétne kroky pre vybraný proces v prostredí Redmine.

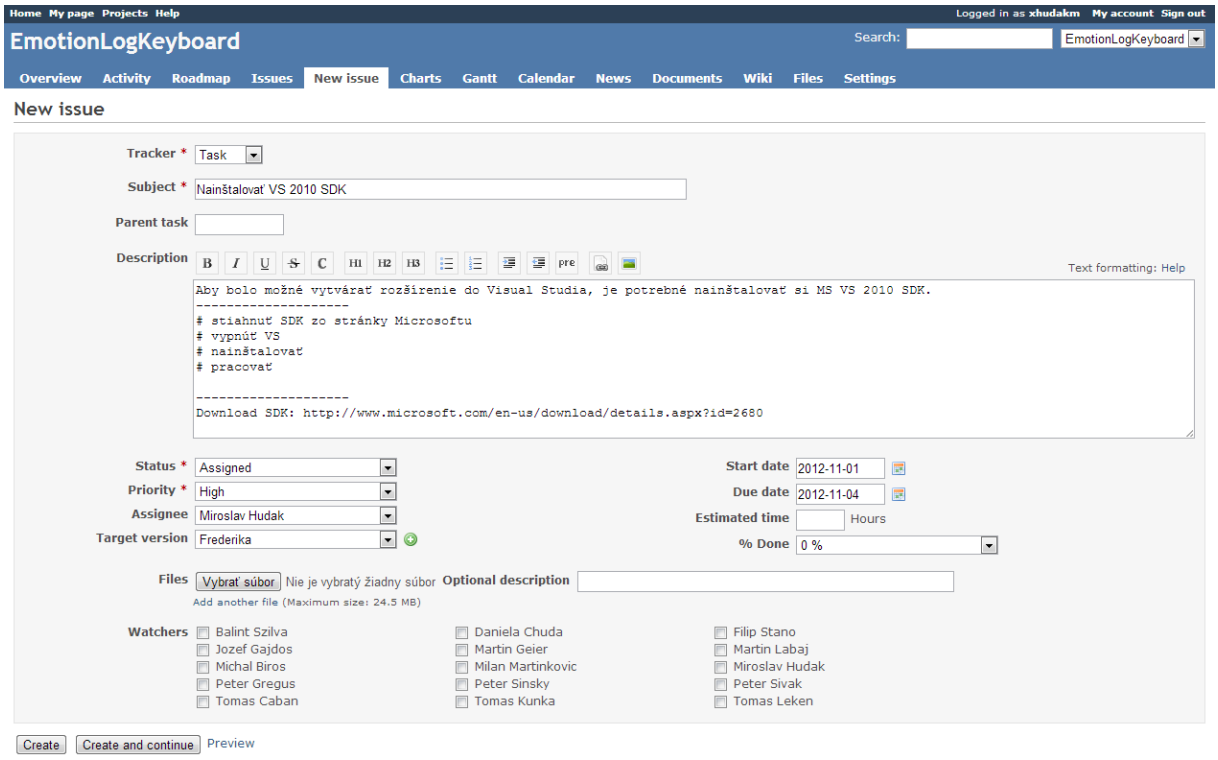
12.7.6.1 ZAPÍSANIE NOVEJ ÚLOHY

0. Pred vykonaním procesu je potrebné mať správne nakonfigurovanú aplikáciu Redmine a vytvorený projekt.
1. Na úvodnej stránke projektu vyberte záložku *Nová úloha* (*New issue*).
2. Pozn.: V záložke Úlohy (*Issues*) je možné prezerať si už zapísané úlohy.
3. Ako typ úlohy zvoľte *Task*.
4. Zadajte názov úlohy. Názov musí začínať slovesom a obsahovať najviac 50 znakov. Má jasne a výstižne definovať čo sa v úlohe očakáva. Bližšie detaily sú obsiahnuté v opise.
5. Vyplňte opis (*Description*). Opis obsahuje doplňujúce informácie k úlohe, ktoré majú pomôcť pri jej riešení. Delí sa na viacero častí:
    a. v prvej časti je rozpísané zadanie úlohy
    b. nasleduje krátky algoritmus riešenia, ktorý je písaný v číslovanom zozname
    c. nakoniec sú k dispozícii komentované externé odkazy a za nimi odkazy na iné úlohy alebo interné materiály

    Jednotlivé časti opisu musia byť viditeľne oddelené čiarou (postupnosťou približne 20 znakov „–“ alebo „_“).

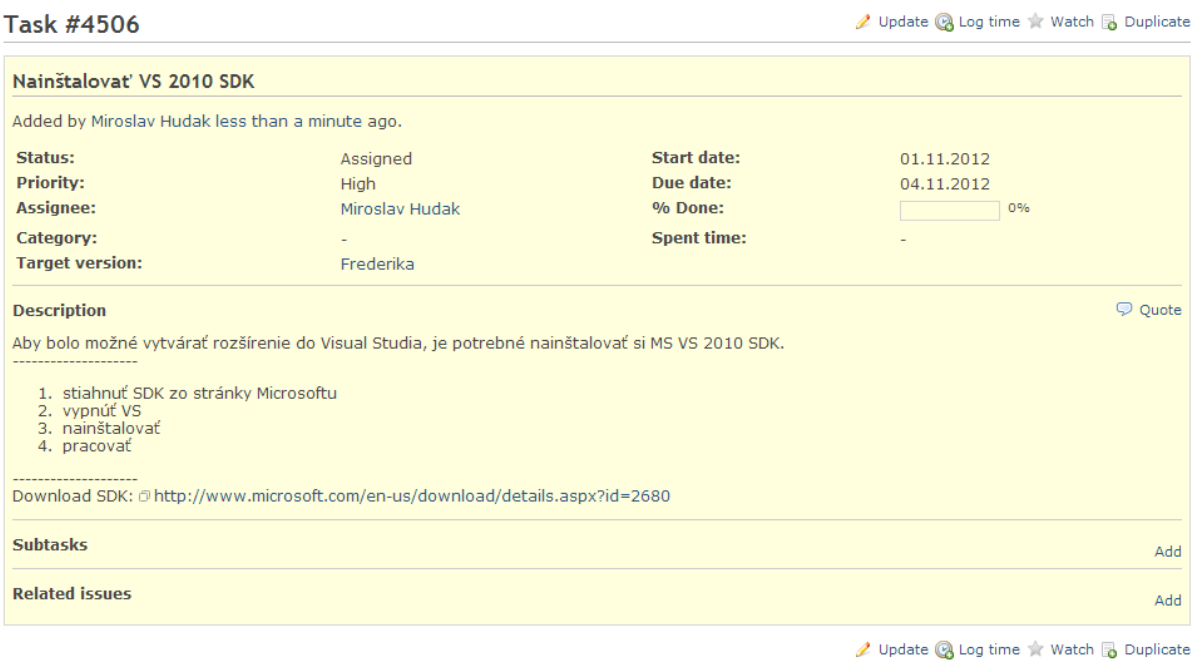
    Opis musí obsahovať všetky tri časti, ale nemusia byť všetky vyplnené, t.j. ak k nejakej časti nie je čo dodať, nechá sa prázdny riadok, urobí sa čiara a pokračuje sa ďalšou časťou. Najtriviálnejšia úloha bude teda v opise obsahovať minimálne prázdny riadok, čiara, prázdny riadok, čiara a prázdny riadok.
6. Ak je identifikovaná nadradená úloha, t.j. úloha, ktorej vyriešenie je nutnou podmienkou pre riešenie tejto úlohy, tak vložte jej ID do poľa Nadradená úloha (P*arent Task*).
7. Nastavte stav (*Status*) úlohy.
    a. Ak úloha ešte nebola pridelená, nastavte stav na *New*. V tomto prípade sa preskakuje krok 9.
    b. Ak úloha bola pridelená, nastavte stav na *Assigned* a v poli *Assignee* vyberte zo zoznamu zodpovednú osobu
8. Nastavte príslušnú hodnotu priority (*Priority*) – prioritu, ktorú určil scrum master tejto úlohe v procese plánovania a rozvrhovania úloh. Ak tento proces ešte neprebehol, nastavte hodnotu na *Normal*.

9. Zo zoznamu verzií vyberte cieľovú verziu pre túto úlohu. Ak taká verzia neexistuje, kliknite na tlačidlo  a vyplnením dialógového okna zadefinujte novú verziu. Opakujte tento krok.
10. Do poľa *Začiatok* (*Start date*) vložte dátum, kedy bola úloha pridelená. Ak ešte úloha nebola pridelená, preskočte tento krok. Dátum sa zadáva vo formáte rrrr-mm-dd.
11. Do poľa *Uzavrieť do* (*Due date*) vložte dátum očakávaného ukončenia úlohy. Dátum sa zadáva vo formáte rrrr-mm-dd.
12. Nastavte pole *% hotovo* (*% Done*) na 0 %.
13. Pridajte prílohy ak existujú.
14. Úlohu vytvorte stlačením tlačidla *Vytvoriť* (*Create*).



OBRÁZOK 13 - VYTVORENIE ÚLOHY V REDMINE

15. Po vytvorení úlohy je zobrazený súhrn (pozri obr. 4).
16. K vytvorenej úlohe je možné vytvoriť podúlohy. Kliknite na *Add*, vytvorí sa nový formulár pre vytvorenie úlohy a opakujte celý proces od začiatku.



OBRÁZOK 14 - POHĽAD NA VYTVORENÚ ÚLOHU

12.8 Manažment verzií a odovzdanie verzie softvérového artefaktu v nástroji *Visual Studio* do verziovacieho systému platformy *Team Foundation Server*

12.8.1 Úvod

Táto metodika sa skladá z dvoch častí. Prvá časť sa zaoberá manažmentom verzií všeobecne a druhá časť detailne opisuje odovzdanie verzie softvérového artefaktu v nástroji *Visual Studio* do verziovacieho systému platformy *Team Foundation Server*.

12.8.2 Pojmy

Pojem	Vysvetlenie
Visual Studio	Integrované vývojové prostredie
Team Foundation Server	Platforma obsahujúca verziovací systém
Softvérový artefakt	Ucelená časť systému
Značkovanie	Priradenie značky verzii softvéru
Beta testovanie	Testovanie koncovým používateľom

12.8.3 Roly, zodpovednosti a procesy

Rola	Zodpovednosť	Proces
Administrátor	• Správa verziovacieho systému	• Konfigurovať verziovací systém
Plánovač	• Plánovanie rozdelenia vyvojových vetiev • Značkovanie verzií	• Naplánovať rozdelenie vývojových vetiev • Označkovať verziu
Programátor	• Vývoj verzií softvéru • Zlučovanie	• Odovzdať verziu softvérového artefaktu • Opraviť chyby • Zlúčiť verzie
Integrátor	• Zlučovanie	• Zlúčiť verzie • Zlúčiť vývojové vetvy
Tester	• Testovanie verzií	• Otestovať verziu

12.8.4 Procesy na vyššej úrovni

Krok	Názov	Kapitola
1	Konfigurovať verziovací systém	4.1
2	Naplánovať rozdelenie vývojových vetiev	4.2
3	Odovzdať verziu softvérového artefaktu	4.3

4	Zlúčiť verzie	4.4
5	Zlúčiť vývojové vetvy	4.5
6	Otestovať verziu	4.6
7	Opraviť chyby	4.7
8	Označkovať verziu	4.8

12.8.4.1 KONFIGUROVAŤ VERZIOVACÍ SYSTÉM

Vstup: Počítače s nenakonfigurovaným verziovacím systémom

Výstup: Počítače s nakonfigurovaným verziovacím systémom

Roly: Administrátor

Administrátor nainštaluje verziovací systém na všetkých počítačoch, ktoré sa budú podieľať na verziovaní vyvíjaného softvéru. Administrátor povolí prístup všetkým programátorom, ktorí budú pracovať s verziovacím systémom. Administrátor kontaktuje plánovača o vykonaní konfigurácie verziovacieho system.

12.8.4.2 NAPLÁNOVAŤ ROZDELENIE VÝVOJOVÝCH VETIEV

Vstup: Požiadavka na naplánovanie rozdelenia vývojových vetiev

Výstup: Rozdelené vývojové vetvy

Roly: Plánovač, Administrátor

Plánovač buď naplánuje nové alebo preplánuje staré rozdelenie vývojových vetiev. Administrátor naplánované vetvy fyzicky vytvorí vo verziovacom systéme a upozorní zúčastnených programátorov o zmenách.

12.8.4.3 ODOVZDAŤ VERZIU SOFTVÉROVÉHO ARTEFAKTU

Vstup: Vykonané zmeny v zdrojových kódoch softvéru

Výstup: Odovzdaná verzia softvérového artefaktu vo verziovacom systéme

Roly: Programátor

Programátor najprv implementuje časť systému na základe požiadaviek plánovača alebo testerov, ak je potrebné opraviť časť zdrojového kódu, a následne odovzdá verziu softvérového artefaktu do verziovacieho systému. Ak systém zamietne odovzdanie verzie z dôvodu nemožného automatického zlúčenia odovzdávaných súborov s už existujúcimi súbormi na serveri, programátor sa najprv pokúsi manuálne zlúčiť požadované súbory s existujúcimi súbormi a ak daný problém nie je schopný vyriešiť, kontaktuje integrátora a ten manuálne zlúči súbory, pri ktorých nastal konflikt.

12.8.4.4 Zlúčiť verzie

Vstup: Rôzne verzie

Výstup: Zlúčené verzie

Roly: Integrátor, Programátor

Integrátor alebo programátor manuálne zlúči verzie, ktoré sa nedali zlúčiť automaticky.

12.8.4.5 Zlúčiť vývojové vetvy

Vstup: Vývojové vetvy

Výstup: Vetva zlúčená zo vstupných vývojových vetiev

Roly: Integrátor

Integrátor zlúči vývojové vetvy. Integrátor kontaktuje programátorov a plánovača o zlúčení vývojových vetiev.

12.8.4.6 Otestovať verziu

Vstup: Odovzdaná verzia softvéru

Výstup: Otestovaná verzia softvéru

Roly: Tester

Tester podľa akceptačných testov otestuje odovzdanú verziu softvérového artefaktu. Ak všetky testy úspešne prebehnú, verzia ide ďalej na označkovanie. V opačnom prípade je verzia vrátená programátorovi, aby opravil všetky zistené chyby. Tester v tomto prípade upozorní programátora o vrátení verzie na opravenie.

12.8.4.7 Opraviť chyby

Vstup: Verzia s chybami

Výstup: Opravená verzia

Roly: Programátor

Programátor dostane späť verziu aj s popisom chýb, ktoré treba opraviť. Programátor opraví všetky vzniknuté chyby a odovzdá opravenú verziu na opätovné pretestovanie.

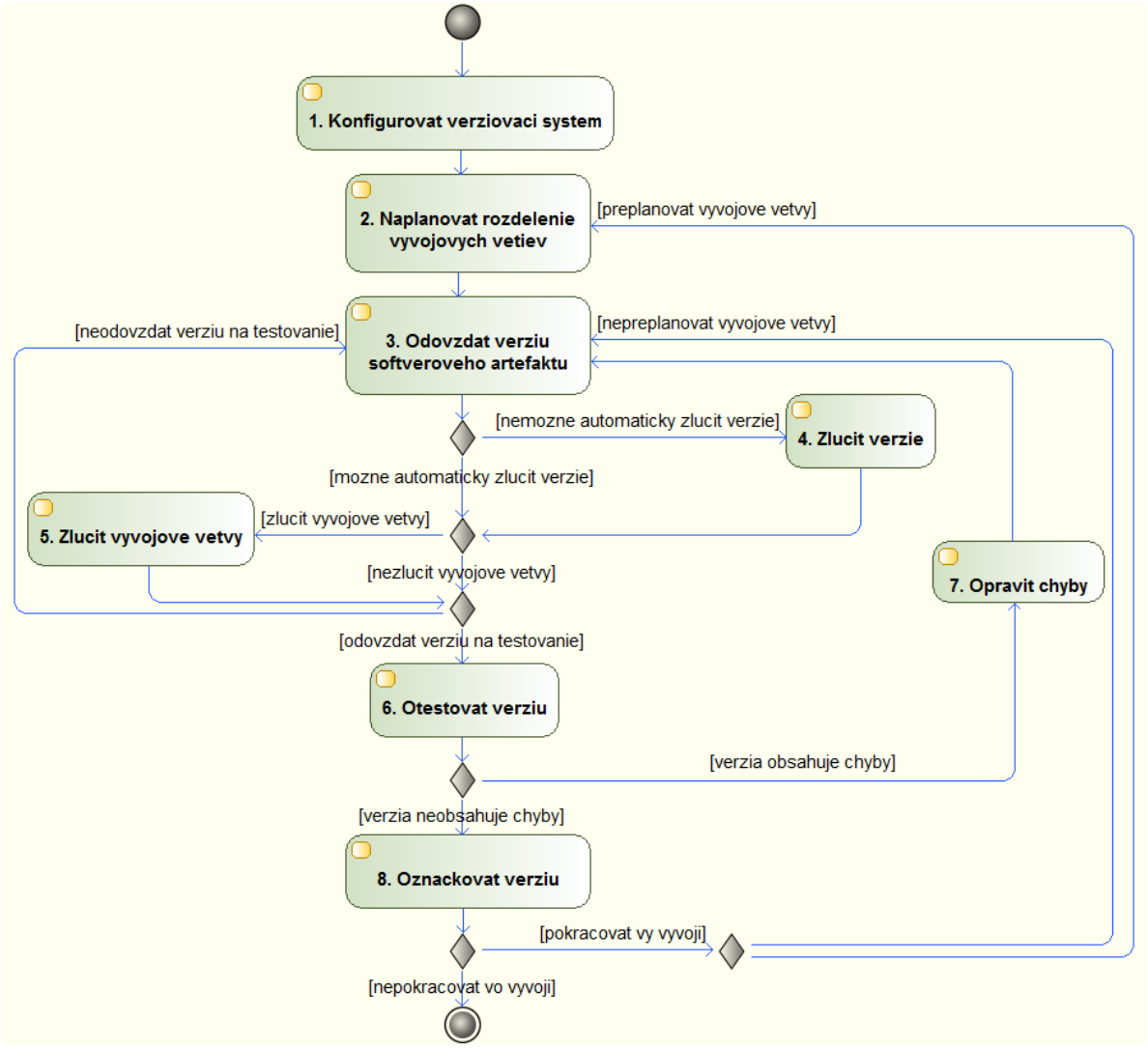
12.8.4.8 Označkovať verziu

Vstup: Otestovaná verzia

Výstup: Označkovaná verzia

Roly: Plánovač

Plánovač dostane otestovanú a zároveň plne funkčnú verziu a podľa jej stavu jej priradí značku. Označkovaná verzia je potom vypustená, buď ako *release* alebo na *beta* testovanie.



Obrázok 4.1 – Diagram aktivít zobrazujúci postupnosť procesov manažmentu verzií

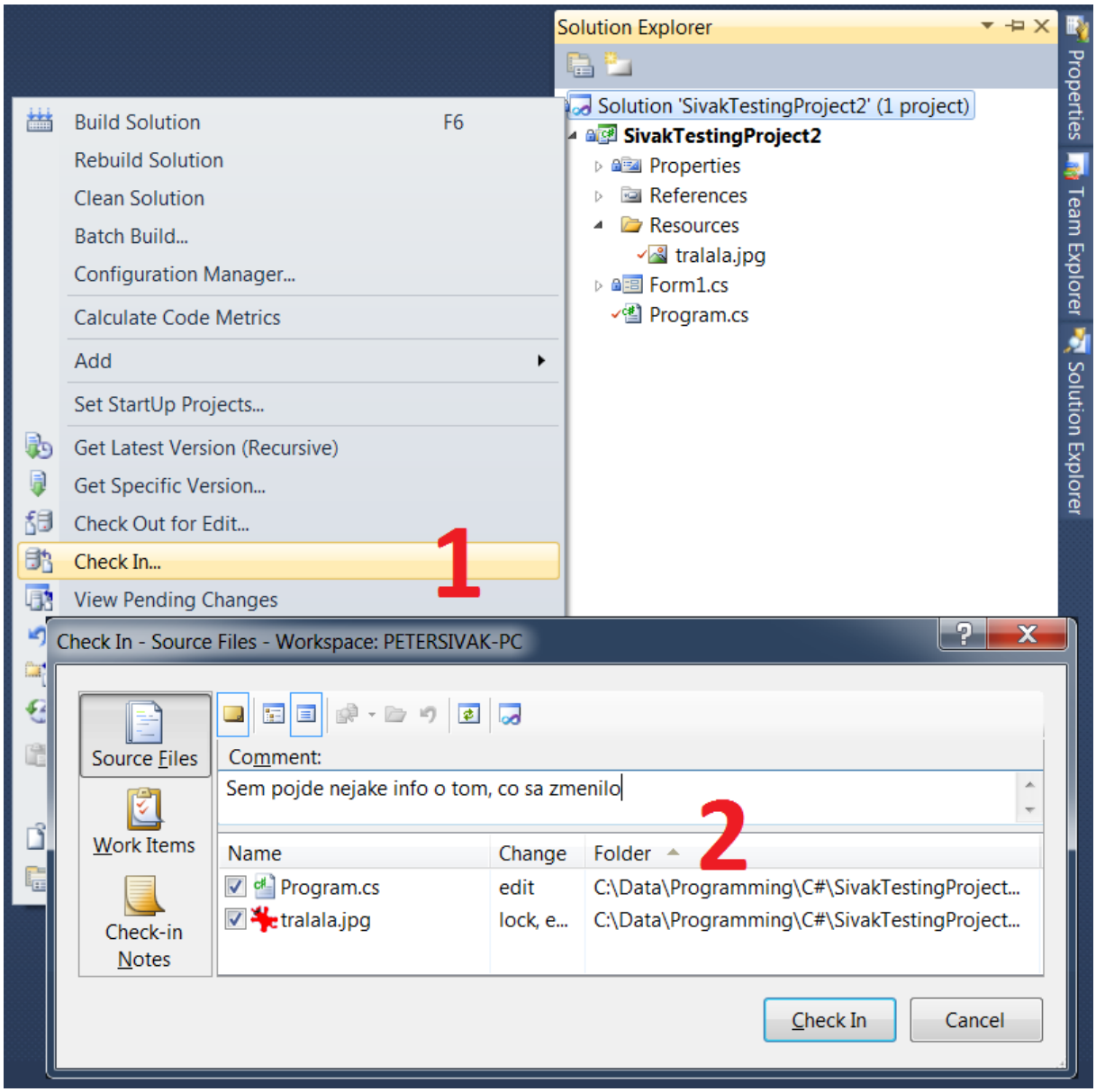
12.8.5 ODOVZDANIE VERZIE SOFTVÉROVÉHO ARTEFAKTU V NÁSTROJI *VISUAL STUDIO* DO VERZIOVACIEHO SYSTÉMU PLATFORMY *TEAM FOUNDATION SERVER*

Táto kapitola detailne opisuje proces odovzdania verzie softvérového artefaktu v nástroji *Visual Studio* do verziovacieho systému platformy *Team Foundation Server*. Verzie použitých nástrojov sú *Visual Studio 2010* a *Team Foundation Server 2010*. V každom z nasledujúcich procesov sa predpokladá, že programátor má spustený nástroj *Visual Studio*.

12.8.5.1 HLAVNÝ TOK

1. Na karte „*Solution Explorer*“ zvoliť možnosť „*Check In...*“ z vyrolovaného menu projektu. Pokiaľ karta „*Solution Explorer*“ nie je viditeľná, zvoliť možnosť „*View*“ z hlavného menu aplikácie a potom „*Solution Explorer*“.
2. Označiť súbory, ktoré sa zmenili, vyplniť komentár pre odovzdanie a zvoliť možnosť „*Check In*“. Komentár musí byť vždy vyplnený a má vystihovať, aké zmeny nastali oproti predošlej verzii. Komentár je celý písaný v slovenskom jazyku. Každá zmena bude vyjadrená presne v jednom riadku komentára. Ak nastalo viac zmien, bude komentár obsahovať viac riadkov. Každý riadok bude mať nasledovnú štruktúru:
   - <typ>: <text>

   teda typ zmeny, dvojbodka, medzera a text zmeny. Za <typ> bude nahradené slovo „*Add*“, ak sa jedná o novú funkcionalitu, alebo „*Fix*“, ak sa jedná o opravenie alebo zmenenie starej funkcionality. Za <text> bude nahradená jedna alebo viac viet popisujúce danú zmenu, pričom každá veta musí začínať veľkým písmenom a končiť bodkou.



Obrázok 5.1 – Hlavný tok odovzdania verzie softvérového artefaktu

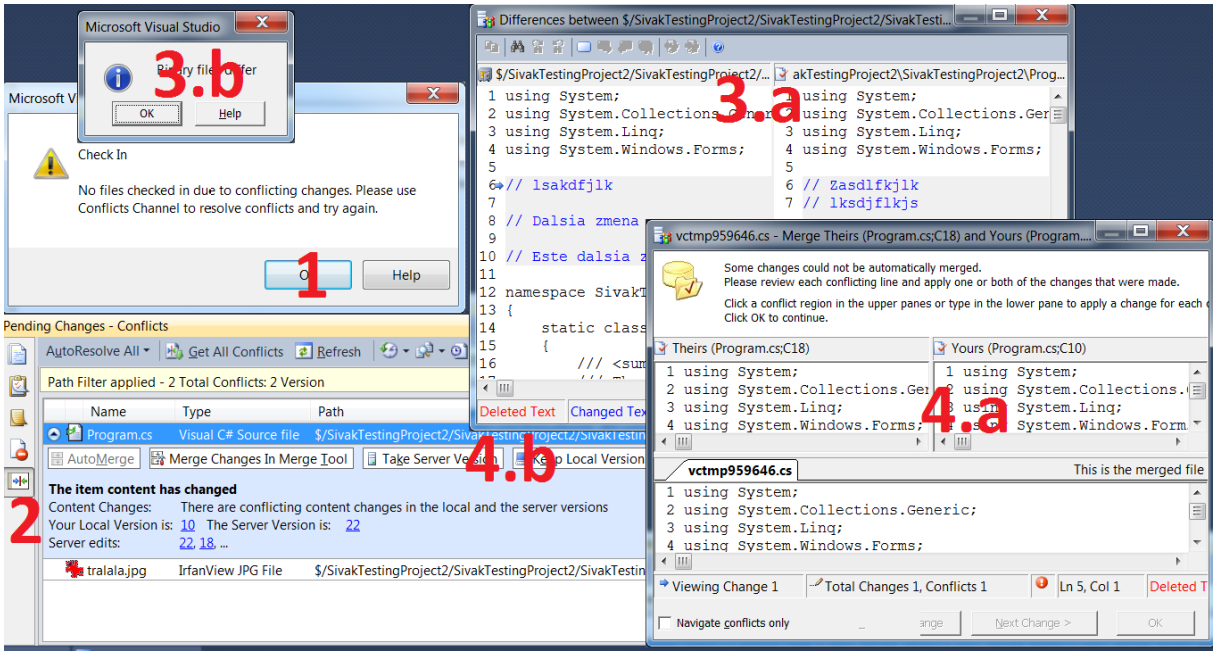
12.8.5.2 Alternatívny tok

Aktivuje sa po kroku 2. hlavného toku, ak sa nepodarilo automaticky zlúčiť nejaké odovzdávané súbory s už existujúcimi súbormi na serveri a zobrazí sa dialóg, ktorý na túto skutočnosť upozorňuje.

1. Zvoliť možnosť „*Ok*“ pre potvrdenie dialógu.
2. Zobrazí sa dialóg so zoznamom súborov, pri ktorých nastal konflikt.
3. Pre každý súbor zo zoznamu zvoliť možnosť „*Compare Local to Server*“.
    a. Ak sa jedná o textový súbor, zobrazia sa dve okná vedľa seba, kde sú zobrazené všetky rozdiely medzi danými súbormi.
    b. Ak sa jedná o binárny súbor, zobrazí sa dialóg so správou „*Binary files differ*“. V tomto prípade musí programátor

manuálne zistiť, aké sú rozdiely medzi verziou súboru na serveri a na jeho lokálnom počítači v závislosti od typu binárneho súboru.

4. Pre každý súbor:
    a. Ak sa jedná o textový súbor, zvoliť možnosť „*Merge Changes In Merge Tool*“. Následne sa zobrazia tri okná reprezentujúce verziu súboru na serveri, verziu súboru na lokálnom počítači a výslednú verziu súboru, ktorú programátor manuálne opraví, aby bola kompatibilná s verziou na serveri.
    b. Ak sa jedná o binárny súbor, zvoliť možnosť „*Take server version*“, ak chce programátor použiť verziu súboru, ktorá sa nachádza na serveri, alebo „*Keep Local Version*“, ak chce programátor použiť verziu súboru, ktorá sa nachádza na jeho lokálnom počítači.



Obrázok 5.2 – Alternatívny tok odovzdania verzie softvérového artefaktu

12.10 METODIKA PRE PÍSANIE ZDROJOVÝCH KÓDOV V C#

Cieľom metodiky písania zdrojových kódov je zadefinovať konvencie písania programových kódov v jazyku C#. A takýmto spôsobom zlepšiť čitateľnosť, prehľadnosť a konzistenciu zdrojových kódov. Metodika hovorí o tvorení mien, štruktúrovania zdrojových kódov a spôsobe písania komentárov.

12.10.1 KONVENCIA PRE POMENOVÁVANIE NAMESPACOV, DÁTOVÝCH TYPOV, METÓD A FIELDOV

1. Všetky mená verejných typov, metód a fieldov musia začínať veľkým písmenom.
2. Všetky mená privátnych a protekčných typov, metód a fieldov sa budú začínať malým písmenom.
3. Všetky názvy musia byť v angličtine a bude sa používať ťaví hrb.
4. *Namespace* vytvorené našim tímom sa musí začínať *FIIT.Team10. ...* (napríklad *FIIT.Team10.Recommendation*).
5. Namespace by mal oddeľovať logický celok aplikácie.
6. Manespace pre extension metódy sa musí volať *FIIT.Team10.Extensions...* jednotlivé pod-namespacesy sa budú volať podľa triedy alebo rozhrania ktoré rozširujú napríklad namespace rozširujúce triedu *Random* sa bude volať *FIIT.Team10.Extensions.RandomExtensions*.
7. Názvy rozhraní sa musia začínať veľkým *I* .
8. Premenné v rámci metódy sa začínajú malým písmenom.
9. Jednopísmenové premenné v rámci metódy treba používať z uvážením.
10. Iba privátne a protekčné atribúty triedy, ktoré sa využívajú iba v jednej metóde alebo jednej automatickej vlastnosti (*getter/ seter*) sa musia začínať podtržníkom, to znamená, že tento atribút nesmie používať iná metóda.
11. Generické typy sa budú označovať veľkými písmenami *T, U, V, W, Z* , ak budú vo význame kľúč- hodnota tak *K, V* .

12.11 KONVENCIE LEXIKÁLNEHO ŠTRUKTÚROVANIA A ÚPRAVY ZDROJOVÉHO KÓDU

1. Množinové zátvorky ohraničujúce *scope* sa budú písať pod seba.
2. Ak za podmienkou alebo cyklom nasleduje len jednoriadkový scope, ktorý je dlhý je vhodné ho dať na nasledujúci riadok a odsadiť od okraja.
3. Ak je lambda výraz dlhší ako jeden riadok, je potrebné ho vyňať do privátnej metódy.
4. Pri fieldoch a metódach písať *this*.
5. Statické *fieldy* volať vždy cez meno triedy, nikdy nie cez *this* alebo bez neho.

6. Vhniezdené typy budú len privátne alebo protekčné.
7. V zdrojových kódoch sa nesmie používať kľúčové slovo *var*, s výnimkou použitia anonymných objektov.
8. Každá trieda musí mať uvedený konštruktor aj keď bude prázdny.
9. Nesmie sa vyhadzovať výnimka *Exception*. Budú sa používať iba konkrétne typy výnimiek.
10. Nemala by sa odchytávať výnimka *Exception*.
11. Mali by sa kontrolovať vstupné parametre metód, a vyhadzovať výnimky *ArgumentException* a *ArgumentNullException*, s uvedeným menom vstupného parametru. Ak bude nutné ich odchytávať do komentáru nad *catch* blok sa uvedie dôvod.

12.11.1 Konvencia komentovania zdrojových kódov

1. Každý súbor bude mať hlavičku v ktorej bude uvedený autor súboru a dátum vytvorenia, pri každej zmene sa aktualizuje dátum zmeny. Ak ho menil iný člen týmu pripíše svoje meno aby bolo jasné kto a kedy robil poslednú zmenu.
2. Na popisovanie dátových typov, metód a fieldov sa budú používať automatické komentáre začínajúce /// .
3. Každá trieda musí mať komentár, ktorý obsahuje jej popis, ostatné členy triedy sa budú komentovať podľa potreby.
4. Rozhrania musia byť okomentované ako celok, ale aj ich každá metóda a aj automatická premenná musí byť komentovaná.
5. Zložitejšie lambda výrazy musia mať aspoň krátky komentár.

12.11.2 Konvencia štruktúrovania zdrojových kódov

1. Každý manespace bude vložený v priečinku, ktorý ma rovnaké meno ako *namespace*.
2. Každá významná trieda alebo rozhranie musí byť vo vlastnom súbore.
3. Je vhodné použiť direktívu *#region* na členenie kódu. Osobitne pre *fieldy*, konštruktory, vlastnosti, verejné metódy a podobne.

12.11.3 Poznámka

Member – člen typu – metóda, atribút, generický atribút

Field – atribút, automatický atribút.

Typ – trieda, rozhranie, štruktúra, enumeračný typ, primitívny typ.

Lambda výraz – anonymná funkcia.

Generický typ – parametrizované typy (v C++ templates)