PRÍLOHA B: METODIKY

Manažment prehliadok

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGII

Metodika k predmetu
Manažment projektov softvérových a informačných systémov
Manažment prehliadok

Autor: Filip Baďura, Bc.
Odbor: Informačné systémy
Šk.rok: 2011/2012
Cvičiaci: Ing. Andrej Danko, PhD.

1 Úvod

Táto metodika slúži na určenie postupov, ktorými by sme sa mali riadiť pri tvorbe nášho projektu v rámci predmetu tímový projekt.

2 Roly a zodpovednosti

V tejto časti je zoznam rolí, ktoré zasahujú do manažmentu prehliadok, spolu s ich úlohami v rámci tímu.

2.1 Moderátor

Zodpovednosti:

- Moderovanie stretnutia k prehliadke
- Vedenie agendy
- Smerovanie diskusie

2.2 Zapisovateľ

Zodpovednosti:

- Vypracovanie zápisnice o stretnutí

2.3 Inšpektor

Zodpovednosti:

- Kontrola zdrojového kódu
- Kontrola dokumentácie
- Identifikovanie chýb
- Komentovanie zistených chýb

2.4 Autor zdrojového kódu

Zodpovednosti:

- Krátka prezentácia nim vytvoreného zdrojového kódu
- Oprava zistených nedostatkov v zdrojovom kóde

2.5 Autor dokumentácie

Zodpovednosti:

- Krátka prezentácia nim vytvorenej dokumentácie
- Oprava zistených nedostatkov v dokumentácií

2.6 Projektový manažér

Zodpovednosti:

- Naplánovanie stretnutia k prehliadke
- Rozdelenie rolí a zodpovednosti

3 Procesy v manažmente prehliadok

3.1 Plánovanie

Vstup: požiadavka na vykonanie prehliadky (kódu, dokumentácie)

Výstup: plán stretnutia, rozdelenie úloh

Zodpovedný: projektový manažér

Projektový manažér naplánuje obsah, čas a dátum konania stretnutia za účelom vykonania prehliadky. Taktiež rozdelí medzi členov týmu jednotlivé roly.

3.2 Príprava

Vstup: zdrojový kód, dokumentácia

Výstup: krátka prezentácia vytvorená autormi dokumentácie a zdrojových kódov

Zodpovední: autori dokumentácie, autori zdrojového kódu

Autori zdrojového kódu si pripravia krátke prezentácie o časti kódu, ktorú vytvorili a rovnako tak si vytvoria krátke prezentácie aj autori dokumentácie k projektu. Autori zdrojových kódov by mali stručne objasniť funkcionalitu prezentovaného kódu, pravidlá použité pri jeho písaní, prípadne poukázať na nevyriešené problémy. V prípade, že mali za úlohu opraviť chyby nájdené počas predchádzajúceho stretnutia, súčasťou ich prípravy je aj ich oprava.

3.3 Stretnutie

Vstup: prezentácie autorov

Výstup: log so zoznamom chýb

Zodpovední: moderátor

Moderátor na začiatku skontroluje, či sú všetci zúčastnení pripravení. Pokiaľ tomu tak nie je stretnutie sa presúva (návrat k procesu 3.1). Autori (zdrojového kódu aj dokumentácie) odprezentujú pripravené prezentácie. Moderátor prevedie všetkých zdrojovým kódom a dokumentáciou a inšpektori poukazujú na chyby. Moderátor všetky tieto chyby zapíše.

3.4 Oprava nájdených chýb

Tento proces sa môže vykonávať v dvoch variantách, a preto obsahuje dva podprocesy.

3.4.1 Oprava chýb priamo na stretnutí

Vstup: log so zoznamom chýb

Výstup: opravený zdrojový kód a dokumentácia

Zodpovední: autor zdrojového kódu, autor dokumentácie

Ak nájdené chyby nie sú závažné a autori sú schopní ich rýchlo opraviť uskutoční sa táto oprava priamo na stretnutí.

3.4.2 Oprava chýb mimo stretnutia

Vstup: log so zoznamom chýb

Výstup: opravený zdrojový kód a dokumentácia

Zodpovední: autor zdrojového kódu, autor dokumentácie

V prípade, že nájdené chyby sú zložitejšie a na ich opravu je potrebný dlhší čas autori neopravujú tieto chyby okamžite ale ich oprava je súčasťou prípravy na ďalšiu prehliadku, na ktorej sa opäť opravované časti skontrolujú.

3.5 Overenie opraveného zdrojového kódu a dokumentácie

Vstup: zdrojový kód, dokumentácia

Výstup: zápis o opravených chybách

Zodpovední: inšpektori

Inšpektori v tomto procese overia, či chyby boli opravené. Jedná tak o chyby nájdené počas predchádzajúceho cvičenia, ako aj chyby opravované počas stretnutia.

3.6 Vypracovanie záznamu o vykonanej prehliadke

Vstupy: poznámky moderátora zo stretnutia, log chýb, zápis inšpektorov

Výstup: zápisnica o vykonanej prehliadke

Zodpovední: zapisovateľ

Zapisovateľ na základe poznámok vedených počas stretnutia (svojich a aj poznámok ostatných členov) vytvorí záznam o stretnutí k prehliadke.

3.7 Diagram procesov

Na obr.1 je znázornená postupnosť procesov pri manažmente prehliadok.

Obrázok 1: Diagram procesov pri manažmente prehliadok

4 Vypracovanie záznamu o vykonanej prehliadke

V tejto časti je opísaný postup pri vypracovávaní záznamu o vykonanej prehliadke, tvorba jej šablóny a vloženie do Redmine-u.

4.1 Vytvorenie šablóny pre zápisnicu o prehliadke

Aby boli všetky záznamy o prehliadkach rovnako štruktúrované, pre ich vytvorením je potrebné vytvoriť šablónu.

Prehliadka č. X

Dátum	
Začiatok	
Koniec	

Zoznam zúčastnených

Meno a priezvisko	**Rola**

Zoznam zistených chýb

No.	**Opis**	**Umiestnenie**	**Autor**	**Stav**

Obrázok 2: Šablóna pre záznam o prehliadke

Inštrukcie k šablóne pri tvorbe záznamu:

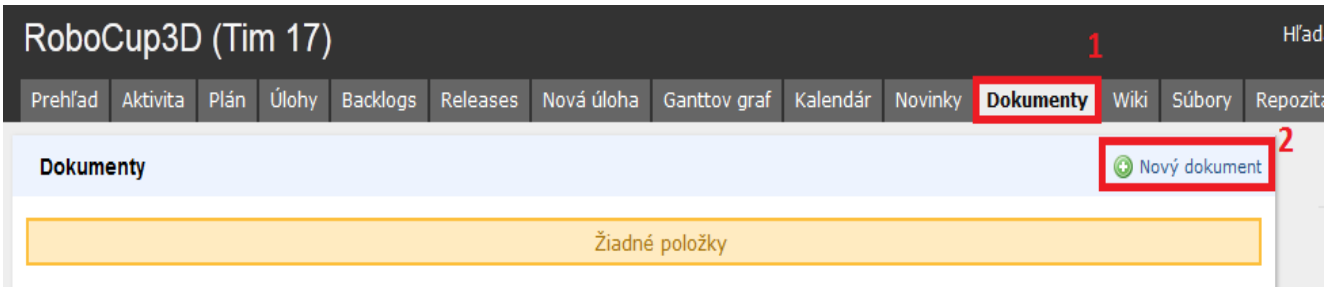
1.Dátum – dátum vykonávania prehliadky
2.Začiatok – čas začiatku stretnutia k prehliadke
3.Koniec – čas ukončenia stretnutia k prehliadke

Zoznam zúčastnených
1.Meno a priezvisko – mená a priezviská všetkých zúčastnených na stretnutí
2.Rola – rola a zodpovednosť každého zúčastneného
Zoznam zistených chýb
1.No. - číslo chyby
2.Opis – stručný opis zistenej chyby
3.Umiestnenie – určenie kde sa chyba vyskytla – zdrojový kód (je dobré uviesť číslo riadku)
- dokumentácia (treba uviesť aspoň číslo kapitoly)
4.Autor – autor/i časti kódu alebo dokumentácie, v ktorej sa chyba vyskytla

5.Stav – informácia, či sa chyba už opravila (priamo na stretnutí) alebo bude opravená a opäť kontrolovaná na ďalšom stretnutí

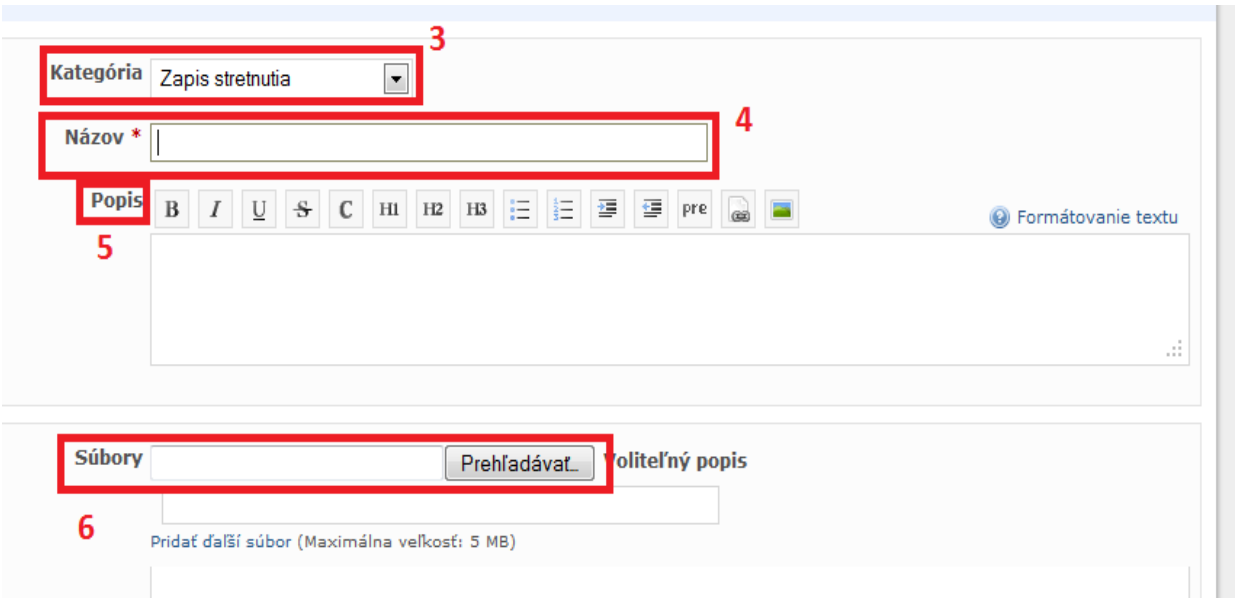
4.2 Vytvorenie záznamu o stretnutí v nástroji Redmine

Táto kapitola hovorí o tom ako vytvoriť a vložiť záznam o stretnutí k prehliadke v nástroji Redmine.



Postup pri vytváraní záznamu o prehliadke:

1. V hlavnej obrazovke kliknúť na *Dokumenty*
2. Ďalej kliknúť *Nový dokument* pre vytvorenie nového dokumentu



3. Vybrať kategóriu
4. Vyplniť názov dokumentu
5. Zadať stručný popis k dokumentu
    - Dátum

- Čas
- Zoznam zúčastnených

6. Vložiť záznam o prehliadke vytvorený podľa šablóny

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Metodika

Manažment zberu požiadaviek
Bc. Tomáš Blaho

Študijný program: Informačné systémy
Ročník: 1
Predmet: Manažment projektov softvérových a informačných systémov
Cvičiaci: Ing. Andrej Danko, PhD.
Dátum: 11.11.2011
Ak. rok: 2011/2012

1. Úvod

Účelom tejto metodiky je definovanie postupu zbierania informácii v malom softvérovóm tíme riadiacim sa SCRUM metódou vývoja, potrebných pre tvorbu požadovaného softvéru, a následné vytvorenie používateľskej príručky. Na vytváranie používateľskej príručky sa používa tímový wiki systém. Touto metodikou sa bude riadiť Tím 17 žije... počas vypracovávania projektu RoboCup 3D v akademickom roku 2011/2012.

2. Použité pojmy

SCRUM – metóda agilného vývoja softvéru
SCRUM Master – rola v metóde agilného vývoja softvéru, dohliada na dodržiavania scrum a agile princípov
Wiki / wiki systém – je webová stránka, ktorá umožňuje návštevníkom pridávať, odstraňovať a upravovať svoj obsah, zakladá sa na myšlienke spoločného spravovania informácií na web stránkach

3. Použitá literatúra a zdroje

[1] MSI Home page, http://www2.fiit.stuba.sk/~bielik/courses/msi-slov/msi-main.html
[2] Wikipedia Home page, http://en.wikipedia.org/wiki/Main_Page
[3] Wikipedia Robocup 3D 2011/2012, http://wiki.holak.net/wiki/
[4] Introduction to scrum, http://www.mountaingoatsoftware.com/topics/scrum
[5] Software development, http://www.integrio.net/

4. Roly zasahujúce do manažmentu zberu informácii

Táto kapitola opisuje roly, ktoré zasahujú do manažmentu zberu požiadaviek podľa metódy vývoja Scrum, pričom vývojový tím je prispôsobený rolám bežného tímu. K rolám sú určené prislúchajúce zodpovednosti.

4.1 Vlastník produktu

Je zadávateľ požiadaviek na vytvorenie softvéru a aktívne sa zúčastňuje stretnutí s týmom. Reprezentuje v sebe vlastníka produktu, ale aj užívateľov, ktorý ho budú používať.

- Nahlásenie požiadavky

4.2 Scrum master

Dohliada nad dodržiavaním SCRUM metódy vývoja v softvérovom tíme a riadi tímové stretnutia, avšak na zbere požiadaviek sa nepodieľa.

4.3 Vývojový tím

Jeho členovia sa zúčastňujú na zbere požiadaviek, vytvorení úloh, ohodnotení úloh, vytvorení diagramov, implementácii softvéru a testovaní softvéru.

4.3.1 Projektový Manažér

Je vedúci člen tímu. V prípade potreby komunikuje so zadávateľom aj mimo stretnutí. Zhromažďuje požiadavky a na ich základe vytvára zoznam úloh. V prípade potreby rozdeľuje prácu na projekte.

- Spracovanie požiadavky
- Zaradenie požiadavky do zoznamu úloh
- Riešenie nedostatku v požiadavke

4.3.2 Analytik

Na základe podkladov od projektového manažéra(po spracovaní požiadaviek) vytvára diagramy, podľa ktorých bude implementovaný softvér a vytvorená používateľská príručka.

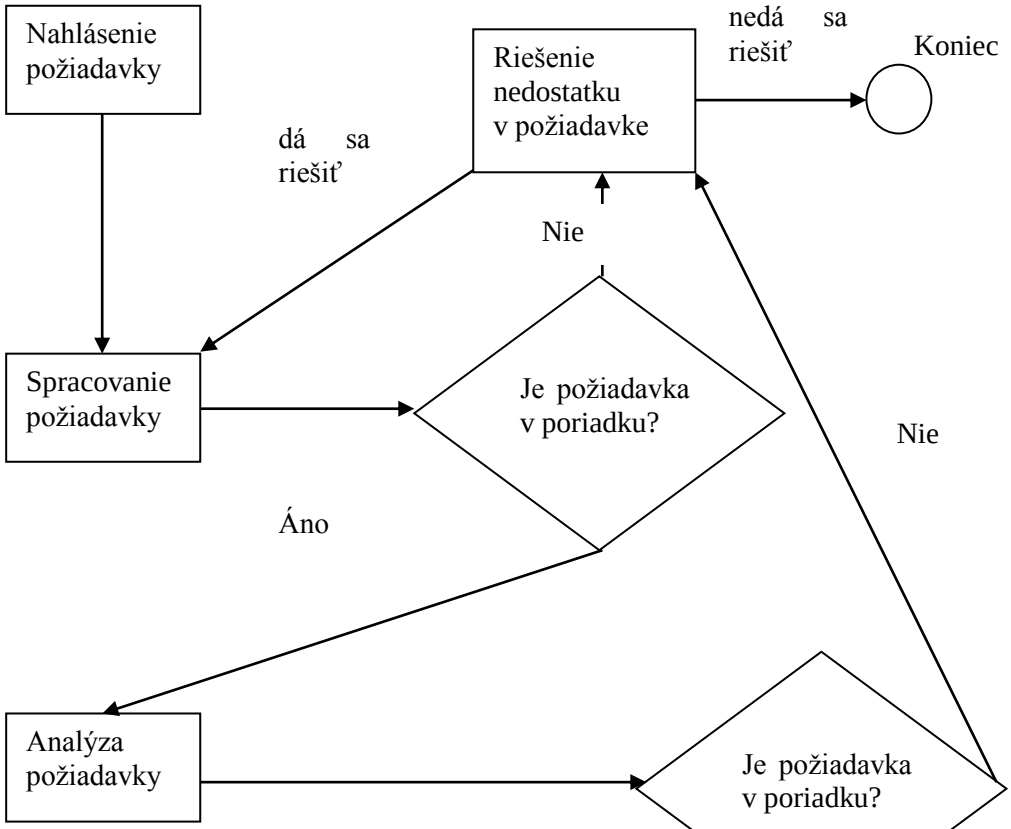
- Analýza požiadavky
- Vytvorenie diagramov z požiadaviek
- Nahlásenie nedostatku v požiadavke

4.3.3 Vývojár

Na základe diagramov od analytika implementuje produkt, ktorý sa bude následne testovať. Pri implementácii môže byť taktiež odhalený nedostatok v požiadavke.

- Nahlásenie nedostatku v požiadavke

5. Procesy v manažmente zberu informácii



obr. 5-1 Diagram procesov v manažmente zberu informácii

V diagrame procesov môžeme vidieť následnosť procesov, ktoré zasahujú do zberu požiadaviek. Vlastník produktu pri rozhovore s tímom vysvetľuje, čo potrebuje, aby bolo vytvorené, a teda nahlási požiadavku. Následne projektový manažér požiadavku spracuje, vyhodnotí jej riešiteľnosť a náročnosť. Následne sa buď požiadavka posunie na analýzu, alebo sa ďalej nerieši. Analytik ďalej spracuje požiadavku a vytvorí diagramy potrebné pri ďalšom vývoji. Ak je všetko v poriadku požiadavka sa transformuje na úlohu . Ak nie analytik problém vráti projektovému manažérovi a ten rieši tento problém so zadávateľom úlohy. Tak isto pri vývoji môže byť odhalená chyba v úlohe, ktorá mohla byť spôsobená nepresnosťou požiadavky, v tom prípade sa problém pokúsi vyriešiť analytik, alebo ten problém môže posunúť projektovému manažérovi, ktorý tento problém rieši s zadávateľom úlohy.

5.1 Nahlásenie požiadavky

Vstupy: -

Výstupy: neupravené požiadavky

Roly: vlastník produktu, projektový manažér

Vlastník produktu podá buď slovne, alebo písomne požiadavky na vývoj softvéru. Projektový manažér sa po oboznámení sa s požiadavkami pýta vlastníka, aby vhodne doplnil požiadavky.

5.2 Spracovanie požiadavky

Vstupy: neupravené požiadavky

Výstupy: upravené požiadavky

Roly: vlastník produktu, projektový manažér

Projektový manažér upravuje požiadavky vlastníka produktu, rozdeľuje požiadavky na jednoduchšie a lepšie riešiteľné požiadavky, triedi a dopĺňa popis k požiadavkám. V prípade, že zistí nedostatok v požiadavke, alebo protichodnosť požiadavky, komunikuje s vlastníkom úlohy, oboznámi ho s problémom a navrhne riešenia. Spolu s vlastníkom dospejú k záveru, či úlohu riešiť alebo nie.

5.3 Zaradenie požiadavky do zoznamu úloh

Vstupy: zanalyzovaná požiadavka
Výstupy: úloha v zozname úloh

Roly: projektový manažér

Projektový manažér dostane výsledky podrobnej analýzy požiadavky (diagramy pre vývoj softvéru) a rozhodne, či zaradiť požiadavku medzi úlohy, ktoré sa budú riešiť v danom šprinte.

5.4 Analýza požiadavky

Vstupy: upravená požiadavka
Výstupy: zanalyzovaná požiadavka (diagramy)

Roly: analytik

Analytik sa pozrie na problém obsiahnutý v požiadavke podrobne, pokúsi sa navrhnúť diagramy, podľa ktorých by vývojár implementoval softvér a podľa ktorých sa projektový manažér rozhodne, či požiadavku zaradiť do zoznamu úloh pre tento šprint. V tejto fáze sa môže objaviť nedostatok v požiadavke, ktorý analytik posunie projektovému manažérovi.

5.5 Nahlásenie nedostatku v požiadavke

Vstupy: upravená požiadavka / zanalyzovaná požiadavka
Výstupy: problém v požiadavke

Roly: projektový manažér / analytik

Problém v požiadavke môže objaviť v prvej fáze projektový manažér, kedy problém hneď diskutuje so zákazníkom, alebo problém v požiadavke objaví analytik, ktorý problém posúva projektovému manažérovi a ten ho rieši. Problém v požiadavke môže objaviť nepriamo aj vývojár, ten však hlási nesplniteľnosť úlohy a analytik zisťuje, kde nastal problém pri návrhu riešenia.

5.6 Riešenie nedostatku v požiadavke

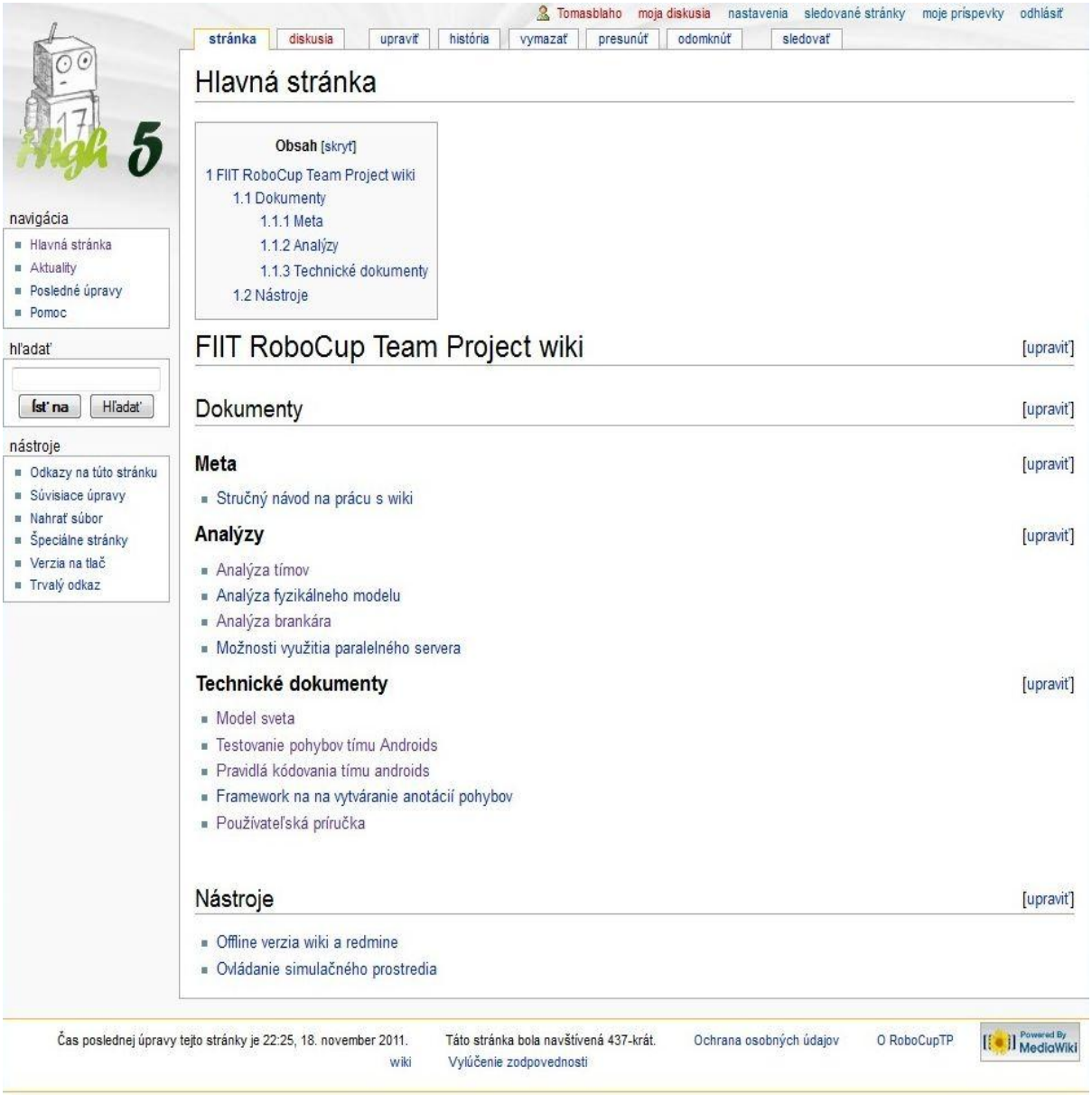
Vstupy: problém v požiadavke
Výstupy: opravená požiadavka / zamietnutá požiadavka

Roly: projektový manažér, vlastník produktu

Projektový manažér v prípade objavenia nedostatku v požiadavke konzultuje opravu s vlastníkom produktu, alebo požiadavku zrušia pre tento šprint.

6. Tvorba internej používateľskej príručky v tímovom wiki systéme

Táto časť obsahuje popis vytvárania používateľskej príručky v tímovom wiki systéme, kde je na tento účel vyhradené miesto. Správne vytváranie používateľskej príručky a dodržanie nižšie uvedeného postupu vytvorí prehľadný návod, ako inštalovať podporné a nutné prostriedky pre spustenie softvéru a ako používať vytvorený softvér. Tím sa uľahčí práca pre tímy nadväzujúce na prácu tímu. Predpokladom pre správne fungovanie wiki systému je autentifikácia pre prístup do systému, aby bolo jasné, kto, čo, kedy upravoval.



obr. 6-1 Wiki systém Robocup 2011

Na obrázku 6.1 je znázornené vyhradené miesto pre tvorbu používateľskej príručky (posledná položka v Technických dokumentoch) a autentifikovaný užívateľ (úplne hore písmom s červenou farbou).

Používateľská príručka:

- môže byť vytvorená, alebo upravená ktorýmkoľvek členom tímu, ktorý vytvorí niektorú časť softvéru, alebo sa podieľal na tvorbe časti softvéru, alebo bol poverený vytvoriť používateľskú príručku,
- sa vytvára na základe vytvoreného softvéru a opisuje sa v nej reálne fungovanie softvéru, nie čo by softvér mal robiť, ktorý bol vytvorený počas jedného šprintu, ak sa v ďalšom šprinte funkcionalita zmení, je nutné upraviť aj príručku,
- v používateľskej príručke pre konkrétnu časť riešenia musia byť uvedené požiadavky na inštaláciu podporných prostriedkov pre časť riešenia, architektúra časti riešenia, popis činnosti časti aplikácie, konfigurovanie časti aplikácie
- pri ukončovaní projektu manažér dokumentácie vytvorí jeden finálny dokument zo všetkých častí.

Tvorba používateľskej príručky sa skladá z dvoch základných procesov, vytvárania a editovania používateľskej príručky.

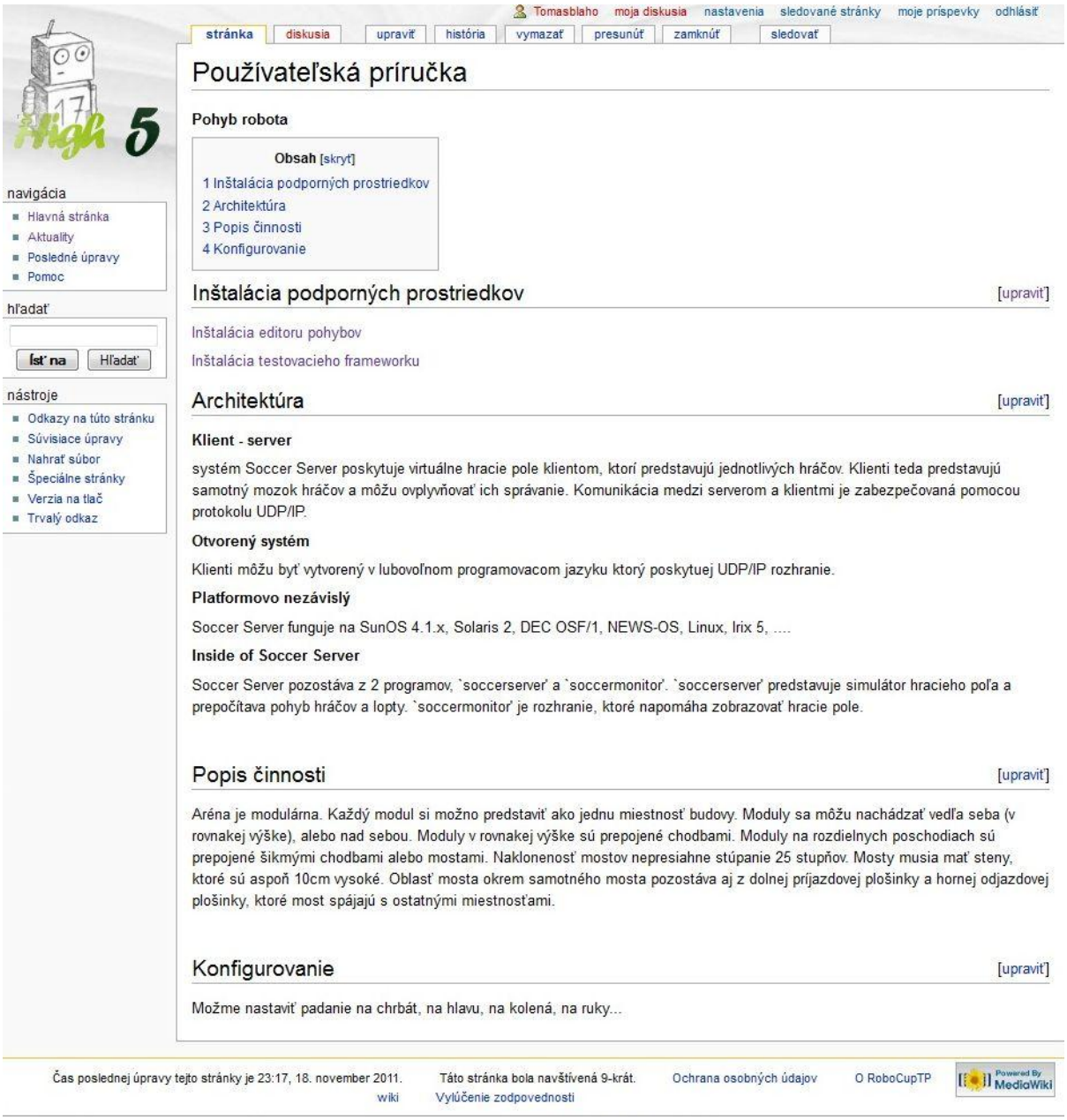
6.1 Vytváranie používateľskej príručky

Vstupy: fungujúci softvér, diagramy
Výstupy: používateľská príručka

Roly: tvorca príručky

Najskôr je nutné zvolenie správneho názvu pre časť používateľskej príručky. Potom tvorca vytvorí odkaz na prázdnu stránku s týmto názvom v mieste vyhradenom pre príručku. Následne tvorca vytvára obsah príručky, na základe fungujúceho softvéru a diagramov, podľa ktorých softvér vytvoril, k čomu mu slúži editor wiki systému s funkciami tučného písma, kurzíva, interného a externého odkazu, nadpisov, vloženia obrázkov, matematických vzorcov, vloženia podpisu s dátumom a časom a vloženia vodorovnej čiary.

Tvorca sa riadi týmito pravidlami:
používajte nadpis pre každú kapitolu (inštalácia, architektúra, popis, konfigurácia),
vyhýbajte sa externým odkazom, radšej text premiestnite na wiki,
ak je možné, použite na vysvetlenie obrázok.



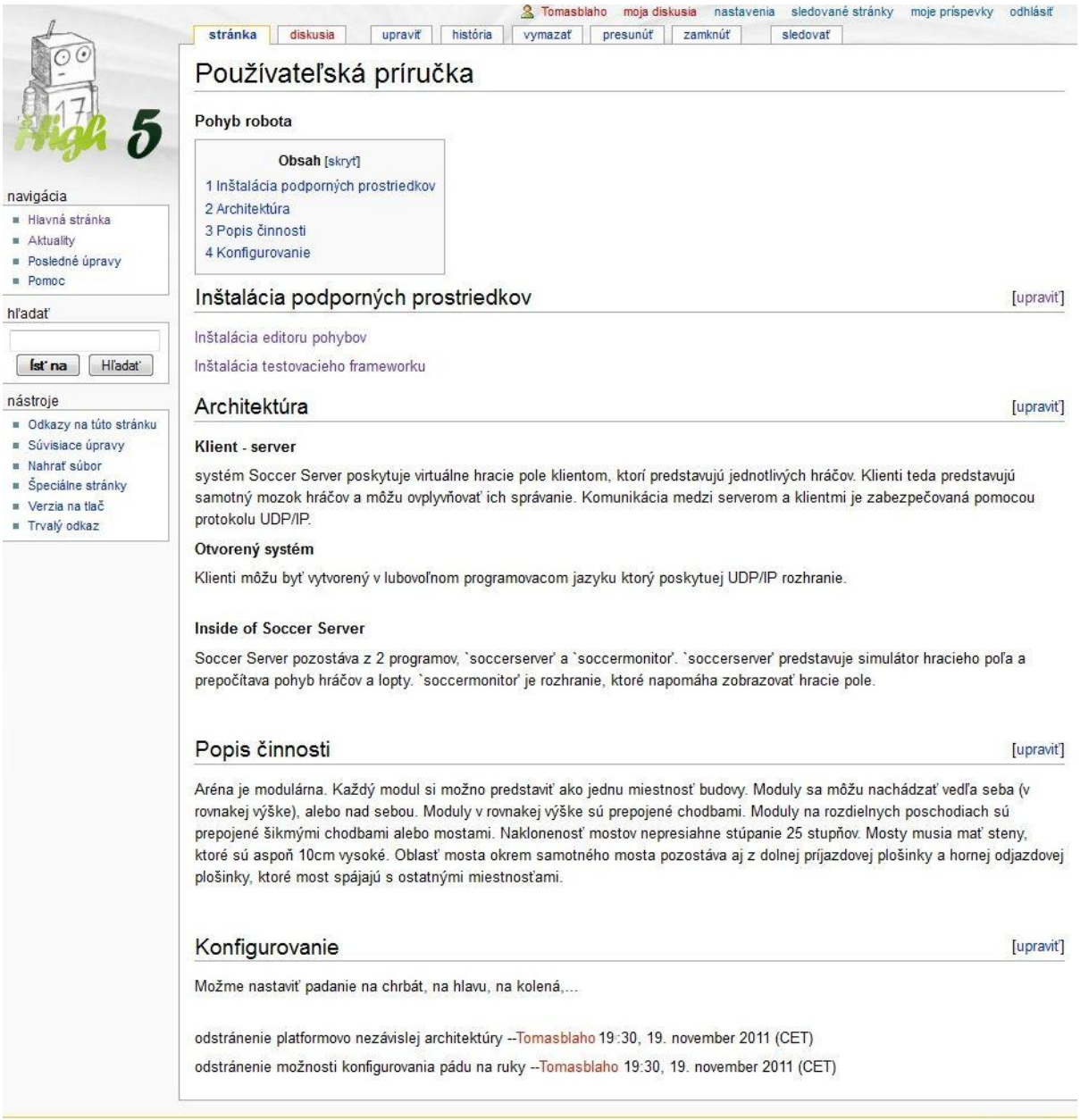
obr. 6-2 Vytvorená používateľská príručka

6.2 Editovanie používateľskej príručky

Vstup: príručka
Výstup: opravená príručka

Roly: tvorca príručky

Pokiaľ sa funkcionalita časti softvéru počas jeho vývoja zmení, je nutné upraviť aj používateľskú príručku. Aby však ostalo zaznamenané pre potreby tímu, čo sa zmenilo, kedy a kto to zmenil, tvorca ktorý upraví príručku pridá na konci stránky zoznam úprav a podpis s dátumom a časom.



obr. 6-3 Editovaná používateľská príručka

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Metodika

Manažment chýb

Bc. Peter Paššák

Študijný program: Softvérové inžinierstvo
Ročník: 1
Predmet: Manažment projektov softvérových a informačných systémov
Cvičiaci: Ing. Andrej Danko, PhD.
Dátum: 11.11.2011
Ak. rok: 2011/12

1 Úvod

Táto metodika slúži na určenie postupov, ktorými sa bude riadiť manažment chýb v tíme Tím 17 žije... počas vypracovávania projektu RoboCup.

2 Súvisiace metodiky

- Metodika manažmentu prehliadok
- Metodika testovania

3 Roly zasahujúce do manažmentu úloh

V tejto časti sú popísané jednotlivé roly, ktoré nejako zasahujú do procesov manažmentu chýb. Pre každú sú spísané prislúchajúce zodpovednosti.

3.1 Projektový Manažér

Predstavuje vedúceho člena tímu, ktorý vie o súčasnom stave projektu, požiadavkách naň a o tom, ktorý vývojár má čo na starosti.

- Analýza nahlásených chýb a ich následné potvrdenie alebo zamietnutie
- Prideľovanie nahlásených chýb na opravu

3.2 Používateľ

Je to používateľ daného softvérového systému, ktorý má možnosť nahlásiť prípadnú nájdenú chybu vo funkcionalite

- Nahlásenie nájdenej chyby

3.3 Vývojár

Je to člen tímu, ktorý má na starosti prácu na implementačnej časti projektu, s ktorou súvisí aj odstraňovanie chýb.

- Nájdenie chyby a jej posúdenie
- Nahlásenie opravenia alebo nemožnosti opravenia pridelenej chyby
- Nahlásenie nájdenej chyby

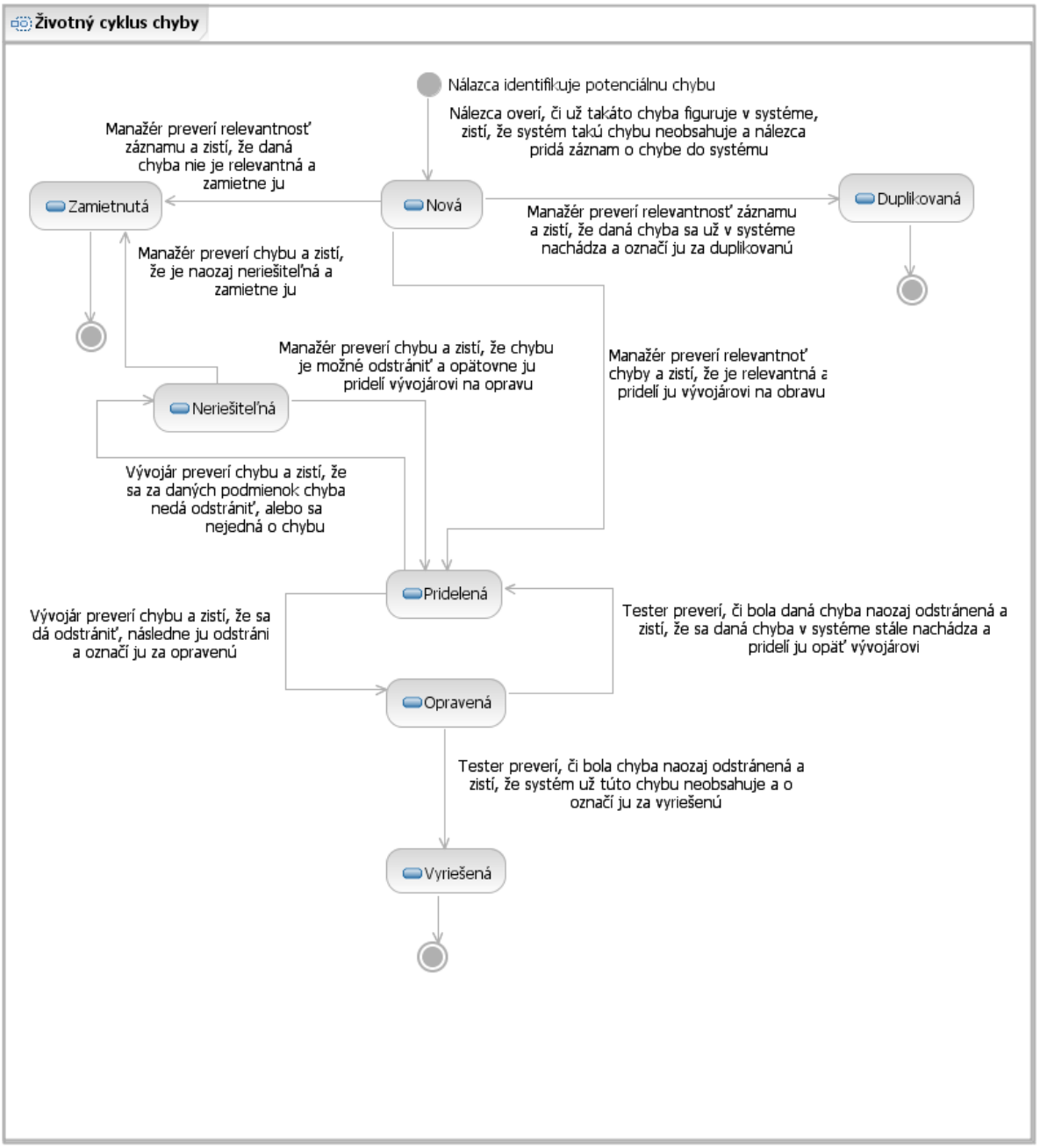
3.4 Tester

Je to člen tímu, ktorého úlohou je testovať funkcionalitu systému a nahlasovať nezrovnalosti v nej.

- Kontrola opravenej chyby
- Nahlásenie nájdenej chyby

4 Procesy v manažmente chýb

V tejto časti sú opísané procesy, ktoré prebiehajú v rámci manažmentu chýb, čo súvisí so životným cyklom chyby, ktorý znázorňuje nasledujúci diagram.



obr. 4-1 Životný cyklus chyby

4.1 Pridanie chybového hlásenia

Tento proces opisuje identifikovanie a následné pridanie chybového hlásenia do systému na správu chýb.

Vstupný stav chyby:

Výstupy stav chyby: nová

Roly: používateľ, vývojár, tester

Proces:

- Identifikovanie nezrovnalosti s funkcionalitou (môže nastať počas prehliadky zdrojového kódu ku ktorej sa vzťahuje metodika manažmentu prehliadok)
- Preverenie, či už do systému takáto chyba nebola pridaná
    a) Pridanie novej chyby, ak taká ešte neexistuje
    b) Doplnenie informácií k už identifikovanej
    c) Žiadna akcia

4.2 Potvrdenie chyby a jej priradenie vývojárovi

Tento proces opisuje postup projektového manažéra, ktorý musí chybu analyzovať a vyhodnotiť jej závažnosť. Následne môže chybu prideliť vývojárovi na opravu, alebo zamietnuť.

Vstupný stav chyby: nová

Výstupný stav chyby: pridelená, zamietnutá, duplikovaná

Roly: projektový manažér

Proces:

- Projektový manažér analyzuje chybu a
    a) zistí, že chyba nie je opodstatnená a nastaví jej stav na zamietnutá
    b) zistí, že daná chyba už je v systéme, nastaví stav na duplikovaná
    c) zistí, že daná chyba je opodstatnená a pridelí ju na opravu vývojárovi, čím zmení stav na pridelená

4.3 Opravenie chyby

Proces opisuje postup pri opravení chyby.

Vstupný stav chyby: pridelená

Výstupný stav chyby: opravená, neriešiteľná

Roly: vývojár

Proces:

- Vývojár reprodukuje a identifikuje chybu
- Vývojár zistí, že
    a) chyba sa dá odstrániť, odstráni ju a zmení stav chyby na opravená
    b) chyba sa nedá odstrániť a zmení stav na neriešiteľná

4.4 Overenie opravenia chyby

Proces opisuje postup pri overení opravenej chyby.

Vstupný stav chyby: opravená

Výstupný stav chyby: vyriešená, pridelená

Roly: tester

Proces:

- Tester skontroluje funkcionalitu, ktorá sa vzťahuje k danej chybe (k tomuto kroku sa vzťahuje metodika testovania)
- Tester zistí, že
    a) funkcionalita bola opravená, nastaví stav chyby na vyriešená
    b) funkcionalita nebola odstránená, vracia chybu vývojárovi a nastavuje stav chyby na pridelená

4.5 Overenie neriešiteľnej chyby

Proces opisuje postup pri overení neriešiteľnej chyby

Vstupný stav chyby: neriešiteľná

Výstupný stav chyby: zamietnutá, pridelená

Roly: projektový manažér

Proces:

- Projektový manažér analyzuje chybu a
    a) zistí, že chyba sa naozaj nedá odstrániť a nastaví jej stav na zamietnutá
    b) zistí, že daná chyba sa dá odstrániť, pridelí ju vývojárovi na opravu, čím zmení stav na pridelená

5 Pridanie chybového hlásenia do systému RedMine

Táto časť obsahuje popis pridania nového chybového hlásenia na nižšej úrovni v systému RedMine. Postup pridania chýb do systému je veľmi dôležitý, pretože pri nesprávnom postupe pridania chybového hlásenia ako je nedodržanie potrebného pomenovania, či vynechanie niektorých potrebných údajov, môže byť ťažké danú chybu identifikovať a následne odstrániť, čo nie je želaná situácia. Pridať chybové hlásenie do systému môže ľubovoľný člen tímu, ale aj iná osoba, ktorá má do systému prístup a povolenie pridávať chybové hlásenia. Predpoklad pre pridanie chybového hlásenia je, že daný používateľ v systéme identifikuje chybu. Postup vytvorenia novej chyby v systéme sa skladá z viacerých krokov a možností.

5.1 Identifikovanie nezrovnalosti s funkcionalitou

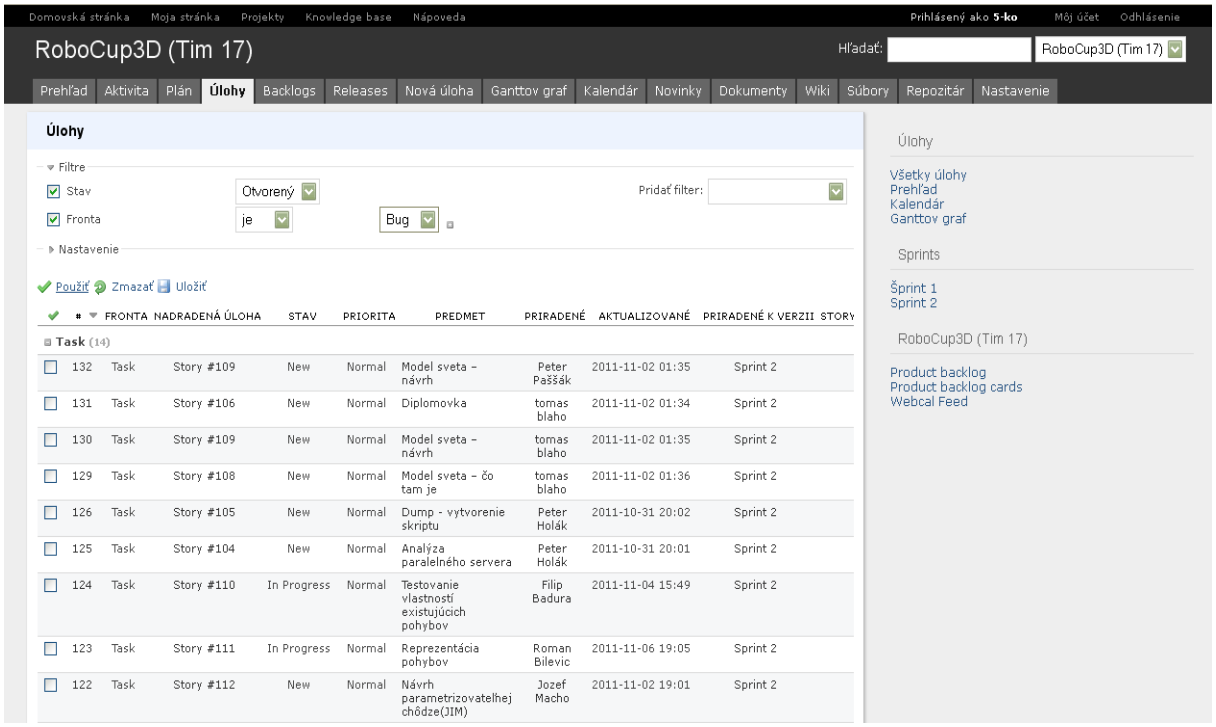
Postup krokov identifikovania nezrovnalosti s funkcionalitou

1. Ľubovoľný člen tímu, alebo iná osoba, ktorá má prístup do systému objaví nezrovnalosť, ktorú považuje za chybu.
2. Nálezca chyby si pripraví potrebné materiály, podľa druhu chyby, ako je obrázok obrazovky s chybovou správou a popis ako k danému postupu došlo, aby bolo možné túto chybu reprodukovať.
3. Nálezca chyby sa následne prihlási do systému a preverí, či sa v ňom už daná chyba nevyskytuje.

5.2 Preverenie, či už do systému takáto chyba nebola pridaná

Postup krokov preverenia jedinečnosti chyby v systéme:

1. Po prihlásení do systému RedMine je dôležité vybrať v menu záložku „Úlohy“, kde sa zobrazí obrazovka s úlohami a tu je dôležité pridať filter na položku „Fronta“, kde treba vybrať hodnotu „Bug“ a použiť tento výber.



obr. 5-1 Redmine - Úlohy

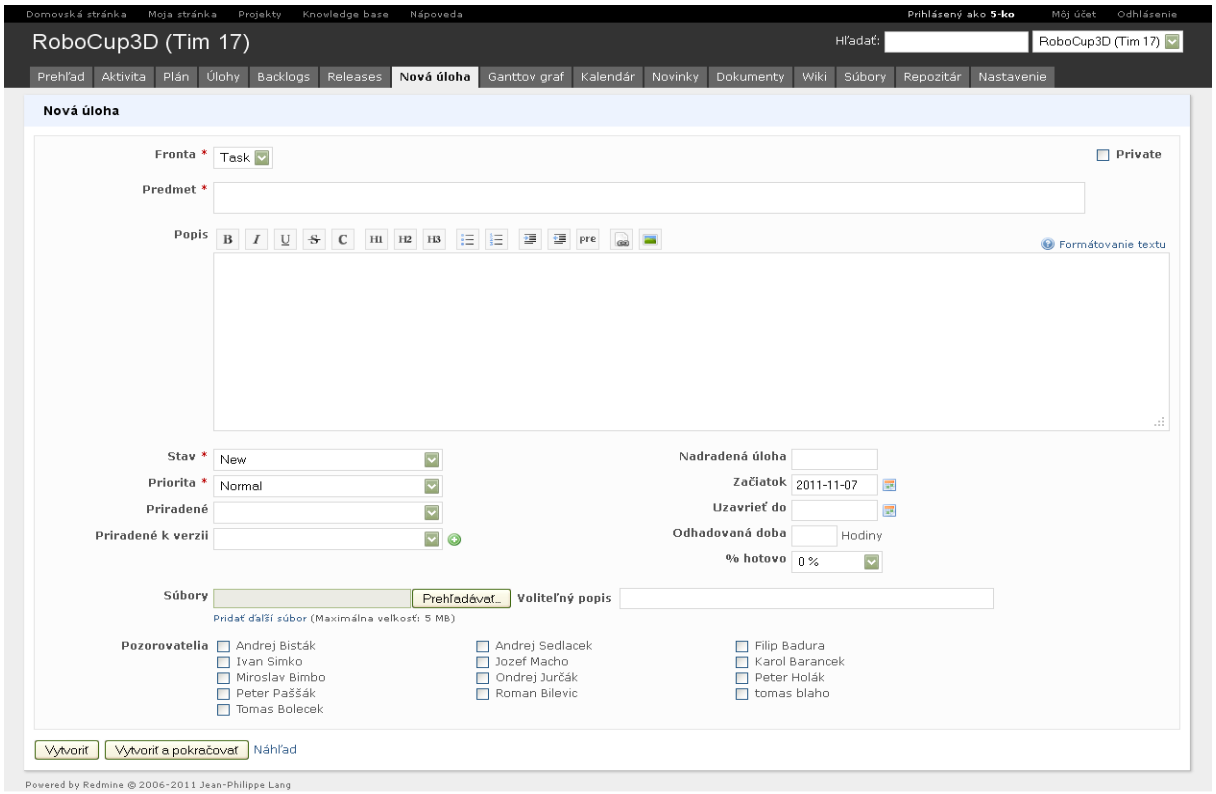
2. Po použití filtra sa zobrazia iba nahlásené chyby, kde daný používateľ vyhľadá chyby podobné tej, ktorú objavil. V prípade, že takú chybu neobjaví, pokračuje pridaním nového chybového hlásenia do systému. Tento krok je rozpísaný v kapitole 5.3. Ak objaví už existujúce chybové hlásenie, ktoré sa zhoduje s chybou, ktorú objavil, môže toto hlásenie buď

rozšíriť, ak pozná detailnejšie informácie, ako tie, ktoré sa v ňom nachádzajú, prípadne doplniť chýbajúce. Tento krok, je podrobne rozpísaný v kapitole 5.4. V prípade ak dané chybové hlásenie obsahuje všetko potrebné, nemusí používateľ robiť nič.

5.3 Pridanie nového chybového hlásenia

Postup krokov pri pridaní nového chybového hlásenia

1. Nálezca chyby sa prihlási do systému RedMine, kde zobrazí záložku „Nová úloha“.



obr. 5-2 Redmine - Nová úloha

2. Tu je dôležité dodržať všetky pokyny na správne pridanie nového chybového hlásenia.

Políčko	Hodnota	Príklad
Fronta	Bug	
Predmet	Verzia systému Časť systému Popis chyby	5.0.1. Agent JIM Nesprávny výpočet vzdialenosti lopta-hráč.
Popis	Detailný popis chyby Ako došlo k chybe Čo sa stalo Čo sa malo stať	Pri výpočte vzdialenosti medzi iným agentom a loptou dochádza k zlému výpočtu vzdialenosti, čo má negatívny dopad na správanie agenta. Pri behu simulácie. Vzdialenosť medzi loptou a hráčom má zápornú hodnotu. Vzdialenosť medzi loptou a hráčom má mať kladnú hodnotu zodpovedajúcu ich skutočnej vzdialenosti.
Stav	Nová	

Priorita	Normálna	
Súbory	Ostatné materiály, ktoré môžu pomôcť pri identifikácii chyby	Náhľad chybovej obrazovky, log chyby v systéme

Políčka, ktoré neboli spomenuté v tabuľke zostanú nevyplnené.

5.4 Doplnenie informácií k už existujúcemu chybovému hláseniu

1. Nálezca chyby sa prihlási do systému Redmine, a zobrazí záložku „Úlohy“
2. Tu vyberie úlohu, ktorá popisuje chybu, zobrazí ju a zvolí možnosť „Aktualizovať“
3. Doplní potrebné informácie k tomuto hláseniu a hlásenie uloží pomocou tlačidla „Potvrdiť“.

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

Manažment testovania

Testovanie pomocou mock objektov

Metodika

Autor: Roman Bilevic
Študijný odbor: Informačné systémy
Ročník: 1. Ing
Predmet: Manažment projektov softvérových a informačných systémov
Cvičiaci: Ing. Andrej Danko, PhD.
Ak. rok: 2011/2012
Semester: Zimný

Metodika testovania

1. Úvod

Účelom tohto dokumentu je určiť postup testovania vo všetkých fázach vývojového cyklu softvéru. Na nižšej úrovni je popísaná metodiky testovania pomocou mock objektov. Na testovanie sa používa nástroj jMock určený pre jazyk Java.

2. Použité pojmy

mock objekt – Objekt simulujúci funkcionalitu časti systému. Vytvára sa ako náhrada za ešte neimplementovanú časť systému, alebo ak by danú časť systému bolo zložité zahrnúť do testovania.
jMock – Knižnica pre jazyk Java umožňujúca vytváranie a nastavovanie mock objektov.
UML (Unified Modeling Language) – Grafický jazyk pre návrh softvérových systémov.
jednotkový test – Test určený na kontrolu jednej funkcie systému.

3. Identifikované roly

Rola	**Zodpovednosť v rámci manažmentu testovania**

Manažér projektu	• Testovanie požiadaviek
	  o Priatie požiadaviek
	  o Odstránenie duplicitných a nepotrebných požiadaviek
	  o Odstránenie defektov
	  o Určenie priorít požiadavkám
	• Testovanie implementácie
	  o určenie novej črty na implementáciu

Tester	• Testovanie požiadaviek
	  o Formulácia požiadaviek
	  o Zachytenie defektov
	  o Odstránenie defektov
	• Testovanie návrhu
	  o Overenie správnosti identifikovaných rolí
	  o Overenie správnosti interpretácie požiadaviek
	  o Overenie vhodnosti navrhnutých procesov
	  o Overenie vhodnosti reprezentácie dát
	  o Overenie vhodnosti navrhnutej architektúry systému
	• Testovanie implementácie
	  o Vytvorenie jednotkového testu
	  o Overenie testu
	  o Vytvorenie mock objektov
	  o Vykonanie testov
	• Testovanie nasadenia
	  o Kontrola externých softvérových komponentov
	  o Kontrola prostredia
	  o Kontrola integrity inštalačného balíka
	  o Kontrola základnej funkcionality
	  o Kontrola špecifickej funkcionality

Zákazník	• Testovanie požiadaviek o Odstránenie defektov

	• Testovanie návrhu o Overenie správnosti identifikovaných rolí o Overenie správnosti interpretácie požiadaviek o Overenie vhodnosti navrhnutých procesov o Overenie vhodnosti reprezentácie dát o Overenie vhodnosti navrhnutej architektúry systému
Vývojár	• Testovanie implementácie o Implementovanie črty o Refaktorovanie

4. Testovanie požiadaviek

Testovanie požiadaviek sa vykonáva bezprostredne po ich zbere (Proces zberu požiadaviek je opísaný v *Metodike pre spracovanie zoznamu požiadaviek*). Účelom je odhalenie nejednoznačnosti, nekompletnosti a nekonzistentnosti požiadaviek. Počas testovania sa samotná požiadavka upravuje a postup týchto zmien je vyjadrený stavovým diagramom na konci tejto kapitoly.

#	**Krok**	**Kapitola**
1.	Prijatie požiadaviek	4.1
2.	Odstránenie duplicitných a nepotrebných požiadaviek	4.2
3.	Formulácia požiadaviek	4.3
4.	Zachytenie defektov	4.4
5.	Odstránenie defektov	4.5
6.	Určenie priorít požiadavkám	4.6

4.1 Prijatie požiadaviek

Vstup: Požiadavky na produkt
Výstup: Zoznam požiadaviek
Zodpovedný: Manažér projektu

Manažér projektu prijme od zákazníka všetky požiadavky na produkt a vytvorí z nich zoznam.

4.2 Odstránenie duplicitných a nepotrebných požiadaviek

Vstup: Zoznam požiadaviek
Výstup: Užší zoznam požiadaviek
Zodpovedný: *Manažér projektu*

Manažér projektu odstráni zo zoznamu duplicitné požiadavky a požiadavky, ktoré na základe svojich expertných znalostí a skúseností označí za nepotrebné.

4.3 Formulácia požiadaviek

Vstup: Zoznam požiadaviek
Výstup: Zoznam formulovaných požiadaviek
Zodpovedný: Tester

Každá požiadavka sa sformuluje podľa šablóny tak, že sa jej štruktúra bude skladať z identifikačného čísla, názvu, popisu, odôvodnenia a mena zadávateľa.

4.4 Zachytenie defektov

Vstup: Zoznam formulovaných požiadaviek
Výstup: Zoznam defektov
Zodpovedný: Tester

Tester skontroluje požiadavky a zaznamená defekty v podobe rozporuplnosti medzi požiadavkami, konfliktov v odôvodneniach a nekompletnosti požiadaviek, pokiaľ bola prehliadnutá dôležitá funkcionalita systému.

4.5 Odstránenie defektov

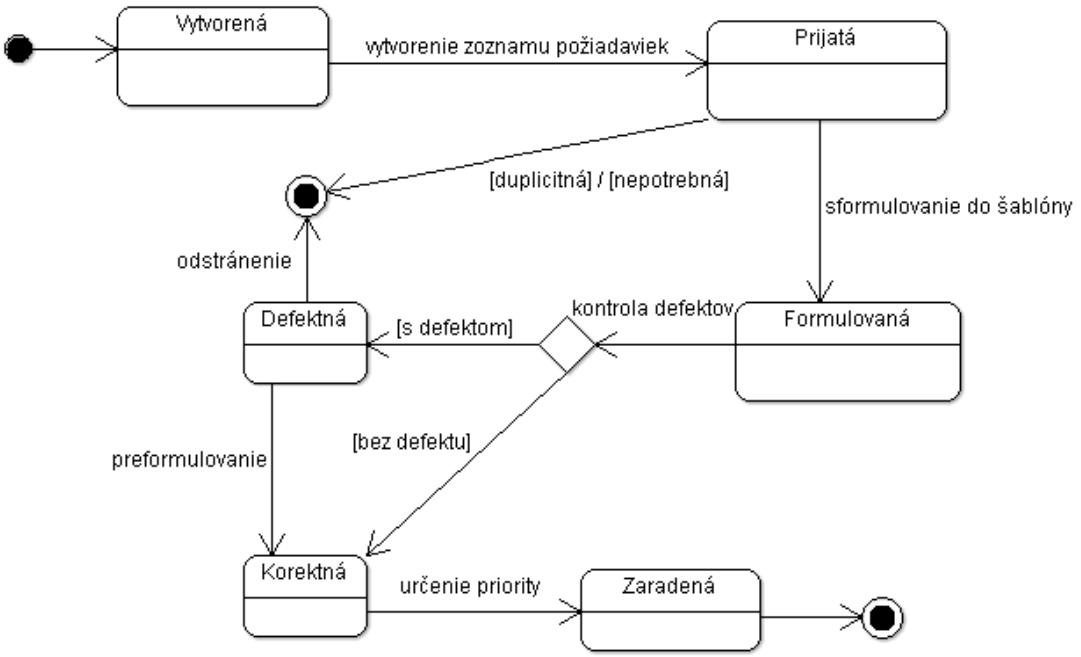
Vstup: Zoznam formulovaných požiadaviek, Zoznam defektov
Výstup: Zoznam korektných požiadaviek
Zodpovedný: Tester, Zákazník, Manažér projektu

Zodpovedné osoby sa zídu a vyjasnia si, či požiadavkám rozumejú všetci rovnako. Následne vyriešia defekty preformulovaním, odstránením alebo pridaním nových, správne sformulovaných požiadaviek.

4.6 Určenie priorít požiadavkám

Vstup: Zoznam korektných požiadaviek
Výstup: Usporiadaný zoznam požiadaviek
Zodpovedný: Manažér projektu

Manažér projektu určí prioritu jednotlivým požiadavkám a takto usporiadaný zoznam posunie do fázy plánovania.



Obrázok č.1: Stavový diagram požiadavky

5. Testovanie návrhu

Testovanie návrhu sa vykonáva bezprostredne po vytvorení všetkých UML diagramov, ktoré boli vytvorené na základe otestovaných požiadaviek. Účelom je overenie logickej stránky UML diagramov pomocou konzultácie so zákazníkom.

#	Krok	Kapitola
1.	Overenie správnosti identifikovaných rolí	5.1
2.	Overenie správnosti interpretácie požiadaviek	5.2
3.	Overenie vhodnosti navrhnutých procesov	5.3
4.	Overenie vhodnosti reprezentácie dát	5.4
5.	Overenie vhodnosti navrhnutej architektúry systému	5.5

5.1 Overenie správnosti identifikovaných rolí

Vstup: UML diagramy
Výstup: Zoznam identifikovaných nedostatkov v UML diagramoch
Zodpovedný: Tester, Zákazník

Tester predvedie vytvorené diagramy zákazníkovi a ten následne určí, či boli identifikované všetky roly hráčov, ktorý budú so systémom pracovať. Zákazník taktiež pri jednotlivých modeloch určí, či boli úlohy pridelené správnym hráčom.

5.2 Overenie správnosti interpretácie požiadaviek

Vstup: UML diagramy
Výstup: Zoznam identifikovaných nedostatkov v UML diagramoch
Zodpovedný: Tester, Zákazník

Tester predvedie vytvorené diagramy zákazníkovi a ten následne určí, či diagramy pokrývajú všetky požiadavky a taktiež, či sú požiadavky z logického hľadiska správne namodelované.

5.3 Overenie vhodnosti navrhnutých procesov

Vstup: UML diagramy
Výstup: Zoznam identifikovaných nedostatkov v UML diagramoch
Zodpovedný: Tester, Zákazník

Tester predvedie vytvorené diagramy zákazníkovi a ten následne určí, či súhlasí s jednotlivými vnútornými procesmi navrhovaného systému.

5.4 Overenie vhodnosti reprezentácie dát

Vstup: UML diagramy
Výstup: Zoznam identifikovaných nedostatkov v UML diagramoch
Zodpovedný: Tester, Zákazník

Tester predvedie vytvorené diagramy zákazníkovi a ten následne určí, či súhlasí s navrhnutými dátovými štruktúrami a postupom spracovania týchto dát.

5.5 Overenie vhodnosti navrhnutej architektúry systému

Vstup: UML diagramy
Výstup: Zoznam identifikovaných nedostatkov v UML diagramoch
Zodpovedný: Tester, Zákazník

Tester predvedie vytvorené diagramy zákazníkovi a ten následne určí, či súhlasí s navrhovanou architektúrou systému, rozložením funkcionality do jednotlivých komponentov a rozdelením tried do balíkov.

6. Testovanie implementácie

Testovanie implementácie prebieha iteratívne počas celej fázy implementácie a je založené na prístupe „Test driven development“. Účelom je overovať funkčnosť implementácie každej črty samostatne.

#	Krok	Kapitola
1.	Určenie novej črty na implementáciu	6.1
2.	Vytvorenie jednotkového testu	6.2
3.	Overenie testu	6.3
4.	Implementovanie črty	6.4
5.	Vytvorenie mock objektov	6.5
6.	Vykonanie testu	6.6
7.	Refaktorovanie	6.7

6.1 Určenie novej črty na implementáciu

Vstup: Zoznam čŕt na implementáciu
Výstup: Jedna črta na implementáciu
Zodpovedný: Manažér projektu

Manažér projektu určí jednu črtu zo zoznamu čŕt, ktorá sa bude v tejto iterácii implementovať.

6.2 Vytvorenie jednotkového testu

Vstup: Črta určená na implementáciu
Výstup: Jednotkový test pre danú črtu
Zodpovedný: Tester

Tester vytvorí na základe prípadov použitia a používateľských príbehov (angl. user stories) jednotkové testy tak, aby pokryl celú funkcionalitu danej črty.

6.3 Overenie testu

Vstup: Jednotkový test pre danú črtu
Výstup: Výsledok testu
Zodpovedný: Tester

Tester vykoná jeho vytvorený test nad už implementovaným kódom. Test musí byť neúspešný s tým, že dôvod jeho neúspechu je nesplnenie podmienok testu. Inak musí byť test prerobený.

6.4 Implementovanie črty

Vstup: Črta

Výstup: Implementácia črty
Zodpovedný: Vývojár

Vývojár napíše kód, ktorý predstavuje implementáciu črty alebo jej časti. Pri písaní kódu vývojár implementuje iba takú funkcionalitu, ktorá bude viesť k úspešnému vykonaniu jednotkového testu.

6.5 Vytvorenie mock objektov

Vstup: Popis požadovanej funkcionality mock objektu
Výstup: Mock objekt
Zodpovedný: Tester

Tester vytvorí mock objekty len v prípade, že implementácia črty očakáva reakcie tej časti systému, ktorá ešte nebola implementovaná, alebo má nahradiť tú časť systému, ktorej zapojenie do jednotkového testovania by bolo príliš zložité alebo náročné na čas. Vytváranie mock objektov je bližšie popísané v kapitole 8.

6.6 Vykonanie testov

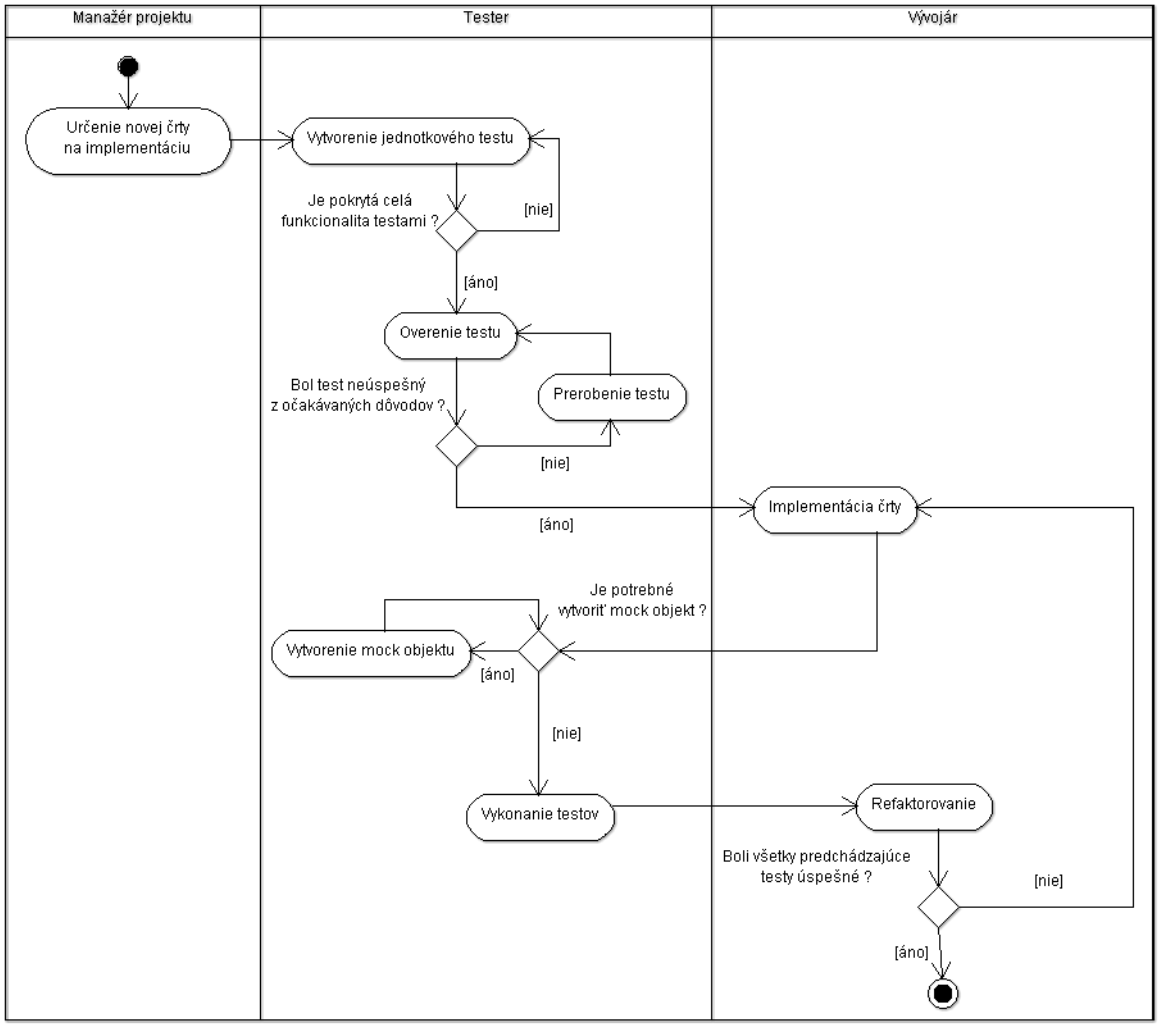
Vstup: Implementácia črty, Jednotkové testy pre danú črtu
Výstup: Správa o výsledku testu
Zodpovedný: Tester

Tester vykoná jednotkové test y a zaznamenáva ich priebeh a výsledky. Pokiaľ bol test úspešný, tak sa po refaktorovaní ukončí práca na aktuálnej črte, inak sa pokračuje v jej implementácii.

6.7 Refaktorovanie

Vstup: Kód implementácie
Výstup: Upravený kód implementácie
Zodpovedný: Vývojár

Vývojár má možnosť odstrániť nepotrebný kód, bez zmeny funkčnosti implementácie. Jedná sa predovšetkým o odstránenie mock objektov.



Obrázok č.2: Diagram aktivít testovania implementácie

7. Testovanie nasadenia

Testovanie nasadenia prebieha počas nasadzovania systému do nového prostredia ako súčasť odovzdávania systému zákazníkovi. Účelom je odhalenie nedostatkov prostredia pre správny beh systému a ovplyvnenie funkcionality systému novým prostredím. Výsledkom testovania funkcionality sú správy o chybách. (Proces vyvárania správy o chybách je bližšie popísaný v *Metodike manažmentu chýb.*)

#	**Krok**	**Kapitola**
1.	Kontrola prostredia	7.1
2.	Kontrola integrity inštalačného balíka	7.2
3.	Kontrola základnej fumkcionality	7.3
4.	Kontrola špecifickej funkcionality	7.4

7.1 Kontrola prostredia

Vstup: Zoznam softvérového vybavenia
Výstup: Zoznam chýbajúcich softvérových komponentov
Zodpovedný: Tester

Tester skontroluje, či sú v prostredí, kam sa má systém nasadiť nainštalované všetky softvérové komponenty potrebné na inštaláciu a beh systému, či je k dispozícii dostatočne veľký diskový priestor a či sú správne nastavené premenné prostredia.

7.3 Kontrola integrity inštalačného balíka

Vstup: Inštalačný balík
Výstup: Správa o kompletnosti inštalácie
Zodpovedný: Tester

Tester overí, či boli nainštalované všetky časti systému a boli prenesené všetky potrebné dáta.

7.4 Kontrola základnej funkcionality

Vstup: Zoznam základných funkcií
Výstup: Správa o chybách
Zodpovedný: Tester

Tester overí správnu funkčnosť základných funkcií systému, ktoré využívajú všetci používatelia.

7.5 Kontrola špecifickej funkcionality

Vstup: Zoznam špecifických funkcií pre jednotlivé používateľské skupiny
Výstup: Správa o chybách
Zodpovedný: Tester

Tester overí funkcionalitu špecifickú pre jednotlivé používateľské skupiny.

8. Pravidlá pre vytvorenie a nastavenie mock objektu (nástroj jMock)

Táto kapitola obsahuje podrobnú metodiky k vytvoreniu a nastaveniu mock objektu pomocou knižnice jMock.

#	Krok	Kapitola
1.	Vytvorenie mock objektu	8.1
2.	Vytvorenie a nastavenie očakávaní	8.2
3.	Inicializácia mock objektu	8.3
4.	Zdokumentovanie mock objektu	8.4

8.1 Vytvorenie mock objektu

Vstup: Popis triedy, ktorej mock objekt treba vytvoriť
Výstup: Časť kódu potrebného k vytvoreniu mock objektu požadovanej triedy bez potrebných nastavení
Zodpovedný: Tester

1. Vytvorenie testovacej triedy rozšírením triedy `TestCase`
   - `class ExampleTest extends TestCase`
   - názov testovacej triedy sa skladá z názvu triedy, ktorej mock objekt chceme vytvoriť (`Example`) a slova `Test`

2. Import tried potrebných na prácu s mock objektmi
   - `import org.jmock.Mockery;`
   - `import org.jmock.Expectations;`
   - `import org.jmock.Sequence;`
   - `import org.jmock.States;`
   - `import org.jmock.api.Expectation;`
   - `import org.jmock.api.ExpectationError;`

3. Vytvorenie kontextu, v ktorom testovaná trieda existuje (postupne sa bude dopĺňať)
   - `Mockery context = new Mockery();`

4. Vytvorenie metódy, ktorá vykoná test
   - `public void testFunctionOfExample()`
   - názov testovacej metódy sa skladá zo slova `test` a vhodného pomenovania funkcie, ktorú má otestovať pričom každé slovo pomenovania funkcie začína veľkým písmenom (`FunctionOfExample`)

5. Vytvorenie mock objektu
   - mock objekt sa vytvorí v rámci testovacej metódy
   - `final Example example = context.mock(Example.class);`
   - `Example` predstavuje názov triedy, ktorej mock objekt sa vytvára
   - názov mock objektu je rovnaký ako názov triedy, ktorej mock objekt sa vytvára a začína malým písmenom (`example`)

8.2 Vytvorenie a nastavenie očakávaní

Vstup: Popis funkcií, ktoré treba simulovať spolu s počtom ich volaní počas testovania. Trieda pre vytváranie mock objektu.
Výstup: Vytvorené očakávania pre daný mock objekt.
Zodpovedný: Tester

1. Vytvorenie zoznamu očakávaní, pokiaľ sa jedná o prvé očakávanie
   - `context.checking(new Expectations(){{ <telo očakávaní>}});`
   - *<telo očakávaní>* predstavuje zoznam jednotlivých očakávaní

2. Definovanie metódy, ktorá sa bude počas testovania volať spolu s očakávaným počtom jej volaní
   - štruktúra: *`<počet volaní>(<názov mock objektu>).<metóda>(<argument metódy>);`*
   - príklad: `oneOf(example).write("message");`

3. Určenie akcie, ktorá sa vykoná následkom volania
   - štruktúra: `will(<akcia>);`
   - príklad: `will(returnValue(0));`

4. Určenie podmienky pre vykonanie akcie, pokiaľ nejaké má

- štruktúra: `when(<názov stavového stroja>.is("<názov stavu>"));`
- príklad: `when(switch.is("off");`

5. Určenie následkov vykonanej akcie, pokiaľ nejaké má
   - štruktúra: `then(<názov stavového stroja>.is("<názov stavu>"));`
   - príklad: `then(switch.is("on");`

6. Zaradiť očakávanie do sekvencie, pokiaľ sa metódy majú volať v určitom poradí
   - štruktúra: `inSequence(<názov sekvencie>);`
   - príklad: `inSequence("calls");`

8.2 Inicializácia mock objektu

Vstup: Požadovaný začiatočný stav systému, ktorý má byť simulovaný mock objektom. Trieda pre vytváranie mock objektu.
Výstup: Inicializovaný mock objekt.
Zodpovedný: Tester

1. Určenie počiatočného stavu jednotlivých stavových strojov
   - štruktúra: `final States <názov stavového stroja> = context.states("<názov stavového stroja>").startsAs("<názov počiatočného stavu>");`
   - príklad: `final States switch = context.states("switch").startsAs("off");`

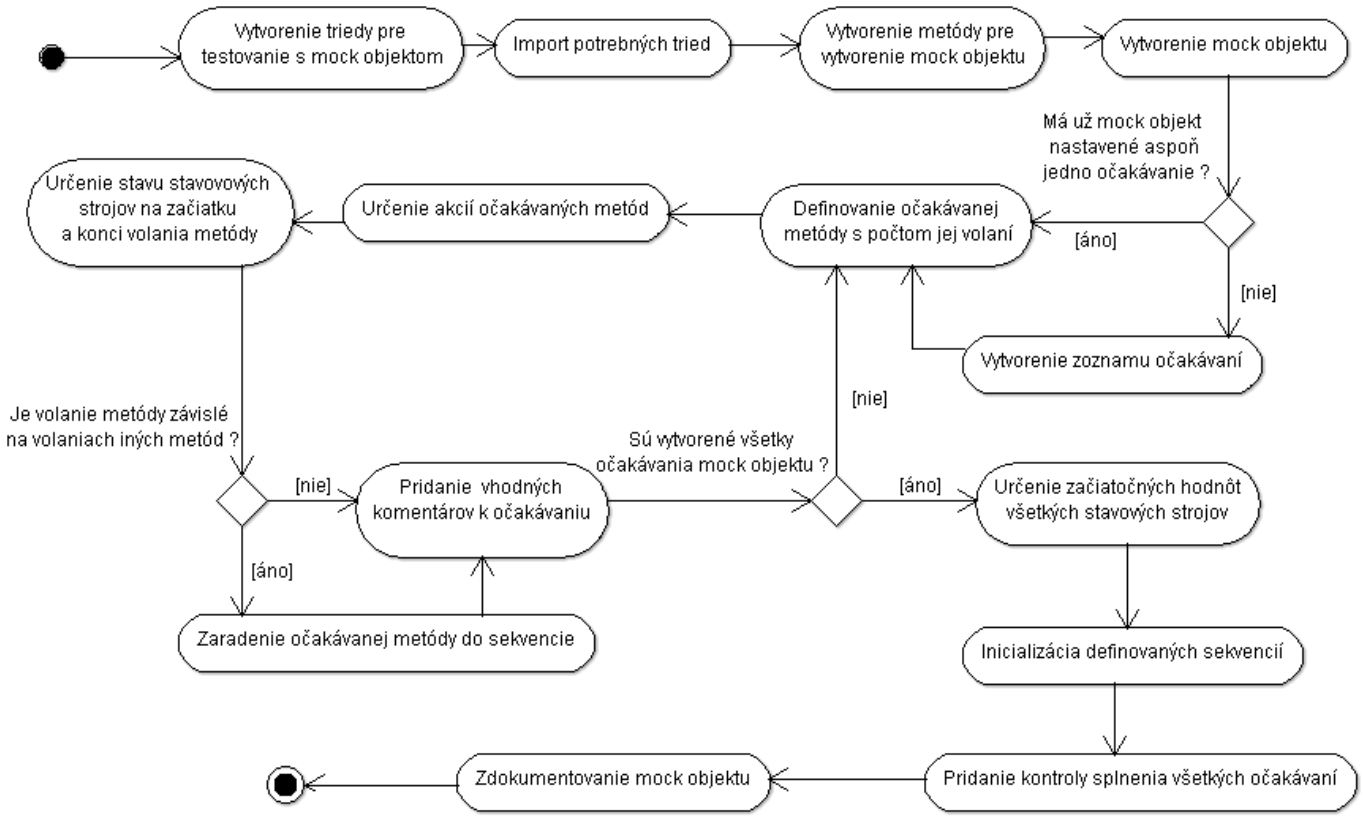
2. Inicializovanie sekvencie volaní metód, ktoré boli použité pri vytváraní očakávaní
   - štruktúra: `final Sequence <názov sekvencie> = context.sequence("<názov sekvencie>");`
   - príklad: `final Sequence calls = context.sequence("calls");`

3. Pridanie kontroly splnenia všetkých očakávaní
   - `context.assertIsSatisfied();`

8.2 Zdokumentovanie mock objektu

Vstup: Kód pre vytvorenie mock objektu
Výstup: Komentár v kóde
Zodpovedný: Tester

1. Meno testera
2. Dôvod vytvorenia mock objektu
3. Opis očakávaní a ich sekvencií
4. Opis simulovaných metód
5. Opis stavov mock objektu



Obrázok č.3: Diagram aktivít vytvorenia a nastavenia mock objektu

Manažment plánovania – spracovanie zoznamu požiadaviek

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

Metodika spracovania zoznamu požiadaviek

Bc. Andrej Bisták

Študijný program: Softvérové inžinierstvo
Ročník: 1
Predmet: Manažment projektov softvérových a informačných systémov
Cvičiaci: Ing. Andrej Danko, PhD.
Ak. rok: 2011/12

Obsah

1. Úvod 3
2. Procesy 3
3. Roly a zodpovednosti 3
4.Životný cyklus plánovania softvéru 4
4.1 Zadefinovanie cieľov 4
4.2 Vymenovanie častí projektu 5
4.3 Vymenovanie zdrojov 5
4.4 Pripravenie harmonogramu 5
4.5 Pripravenie rozpočtu 6
4.6 Vymenovanie rizík 6
5. Spracovanie počiatočných požiadaviek na softvér 7
5.1 Spracovanie požiadaviek na projekt 7
5.2 Analýza podobných projektov 7
5.3 Vytvorenie úloh 8
5.4 Naplánovanie šprintov 9

1. Úvod

Účelom tejto metodiky je definovať procesy potrebné pre úspešné naplánovanie projektu, a teda zadefinovanie výsledku projektu a všetkého, čo je potrebné na dosiahnutie tohto výsledku v stanovenej kvalite, pri dodržaní časového plánu a rozpočtu.

Každý plán má na svojom začiatku zoznam požiadaviek na softvér, ktorý treba správne spracovať a rozbiť na podúlohy a naplánovať na jednotlivé šprinty.

2. Procesy

Tab. 1 Procesy pri spracovaní požiadaviek

Proces	Kapitola
1. Zadefinovanie cieľov	4.1
2. Spracovanie požiadaviek na projekt	5.1
3. Analýza podobných projektov	5.2
4. Vymenovanie častí projektu	4.2
5. Vytvorenie úloh	5.3
6. Naplánovanie šprintov	5.4
7. Vymenovanie zdrojov	4.3
8. Pripravenie harmonogramu	4.4
9. Pripravenie rozpočtu	4.5
10. Vymenovanie rizík	4.6

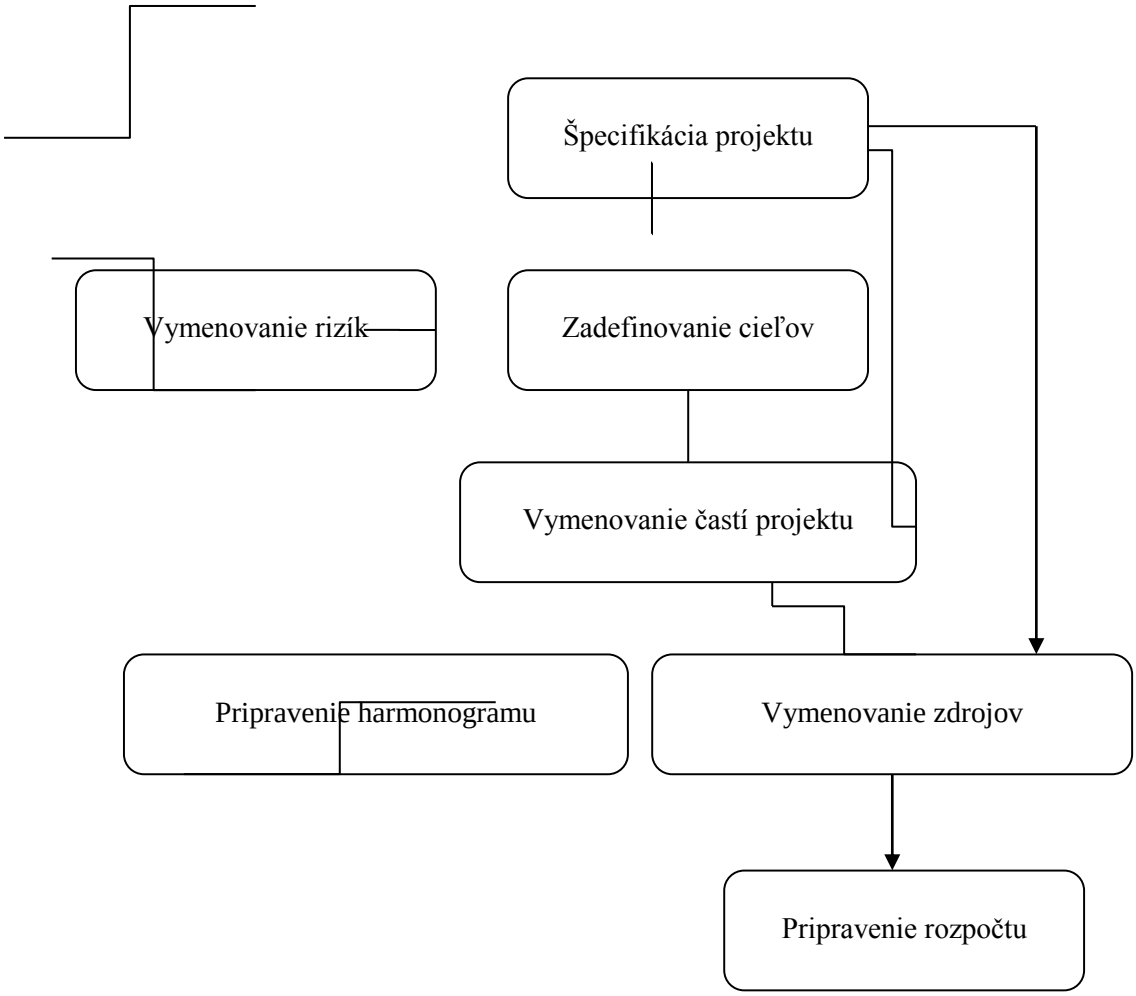
3. Roly a zodpovednosti

Tab. 2 Roly a ich zodpovednosti v procesoch

Rola	Zodpovednosť
projektový manažér	ciele, rozpis prác, plán, rozpočet, riziká
manažér analýzy	rozpis prác
manažér vývoja	rozpis prác
manažér testovania	rozpis prác

Rola	Zodpovednosť
manažér plánovania	plán
manažér návrhu	rozpis prác
manažér rizík	riziká
ekonomické oddelenie	rozpočet

4. Životný cyklus plánovania softvéru



Graf. 1 Životný cyklus plánovania softvéru

4.1 Zadefinovanie cieľov

Vstup: špecifikácia projektu

Výstup: stanovené ciele projektu

Zodpovedná osoba: projektový manažér

Na základe špecifikácie projektu sa stanoví zoznam cieľov, ktoré sa očakávajú dosiahnuť. Musia byť jasné a jednoznačné pre všetky zainteresované strany. Ich definícia má obsahovať kvalitatívne aj kvantitatívne vlastnosti. Ciele projektu majú byť reálne, lebo v opačnom prípade je realizácia projektu neopodstatnená.

4.2 Vymenovanie častí projektu

Vstup: špecifikácia projektu, zadefinované ciele

Výstup: štruktúra rozpisu prác

Zodpovedná osoba: projektový manažér, manažér návrhu, manažér analýzy, manažér vývoja, manažér testovania

Základom projektového plánu je štruktúra rozpisu prác, teda zoznam činností, ktoré treba vykonať pre dosiahnutie cieľa projektu. Tieto činnosti sú komplexné, majú presný začiatok a koniec a ich výsledky sú čiastkovými cieľmi projektu.

4.3 Vymenovanie zdrojov

Vstup: špecifikácia projektu, štruktúra rozpisu prác

Výstup: matica zodpovednosti projektu

Zodpovedná osoba: manažér projektu, manažér návrhu, manažér analýzy, manažér vývoja, manažér testovania

Sú vymenované ľudské zdroje zvlášť pre každú činnosť, t.j. zamestnanci, či iní externí spolupracovníci organizácie, ktorí by sa mohli zapojiť do realizácie projektu na základe ich kvalifikácie a zručností, ktoré si projekt vyžaduje. Na záver je spravená matica zodpovednosti projektu, ktorá slúži na vymedzenie úloh a právomocí členov projektového tímu a externých subjektov.

Sú stanovené ďalšie zdroje potrebné na realizáciu jednotlivých častí projektu, či už je to napr. hardvérové vybavenie, softvér alebo priestory.

4.4 Pripravenie harmonogramu

Vstup: štruktúra rozpisu prác

Výstup: časový plán projektu

Zodpovedná osoba: projektový manažér, manažér plánovania

Je pripravený plán na časový postup jednotlivých činností tak, aby s ohľadom na disponibilné zdroje boli dosiahnuté plánované výsledky projektu v stanovenej kvalite. Časový

plán je zobrazený ako Ganttov diagram, ktorý vyjadruje trvanie jednotlivých činností projektu.

4.5 Pripravenie rozpočtu

Vstup: ľudské a ďalšie zdroje
Výstup: plán cash flow
Zodpovedná osoba: projektový manažér, ekonomické oddelenie

Vymenované zdroje sú ocenené, čím je získaná výška variabilných nákladov na ne. Pripravia sa fixné náklady, ktoré sú rozdelené medzi jednotlivé činnosti. Pripraví sa plán celkových nákladov, ktoré sú zahrnuté v tabuľke položiek a aktivít. Pripraví sa plán výnosov a nakoniec je pripravený plán cash flow.

4.6 Vymenovanie rizík

Vstup: špecifikácia projektu, zadefinované ciele, štruktúra rozpisu prác
Výstup: plán rizík
Zodpovedná osoba: projektový manažér, manažér rizík

Je vytvorený zoznam možných rizík, ktoré zahŕňajú riziká súvisiace so stanovením cieľov, riadením projektu, technickým zabezpečím, financovaním projektu a pod. Plán rizík zahŕňa definíciu, predvídanie, sledovanie a obmedzovanie dôsledkov vzniku rizikových udalostí.

5. Spracovanie počiatočných požiadaviek na softvér

Na začiatku každého projektu je zákazník a jeho požiadavky na finálny softvér, ktoré treba adekvátne spracovať a následne rozdeliť do menších úloh, ktoré majú stanovený svoj začiatok i termín očakávaného konca, ktorých výsledky sú čiastkovými cieľmi projektu a podľa ktorých sa naplánujú samotné šprinty.

5.1 Spracovanie požiadaviek na projekt

Vstup: požiadavky zákazníka na projekt
Výstup: zoznam cieľov
Zodpovedná osoba: projektový manažér

Na základe požiadaviek zákazníka je vytvorený zoznam cieľov, ktoré má projekt splniť, aby bol vyhodnotený ako úspešný. Tento zoznam vznikne rokovaním a následnou dohodou všetkých zainteresovaných strán na projekte.

V rámci Tímového projektu RoboCup-u je touto požiadavkou vylepšenie robota schopného zúčastniť sa turnaja. Po analýze požiadaviek a dostupných možnostiach je vytvorený zoznam cieľov, ktorých úspešné dosiahnutie bude vyústené do splnenia požiadaviek zákazníka. Týmito cieľmi sú najmä vylepšenie existujúceho robota a jeho následné nasadenie na turnaj.

5.2 Analýza podobných projektov

Vstup: dokumentácie k podobným projektom
Výstup: vypracovaná analýza projektov
Zodpovedná osoba: projektový manažér, manažér analýzy

V rámci spracovania počiatočných požiadaviek je potrebné spraviť aj analýzu podobných dostupných projektov, poučiť sa z ich chýb a inšpirovať sa z ich úspešných krokov a riešení.

Počas Tímového projektu je vykonaná analýza niekoľkých tímov zaoberajúcich sa RoboCupom – zahraničné i slovenské. Je zameraná na ich nápady a inovácie pri vylepšovaní robota, na podporné nástroje a dôvody ich úspechov. Výsledkom je vypracovaná analýza, v ktorej sa nachádzajú najpodstatnejšie inovácie, postupy a nápady každého tímu.

5.3 Vytvorenie úloh

Vstup: zoznam cieľov, analýza projektov

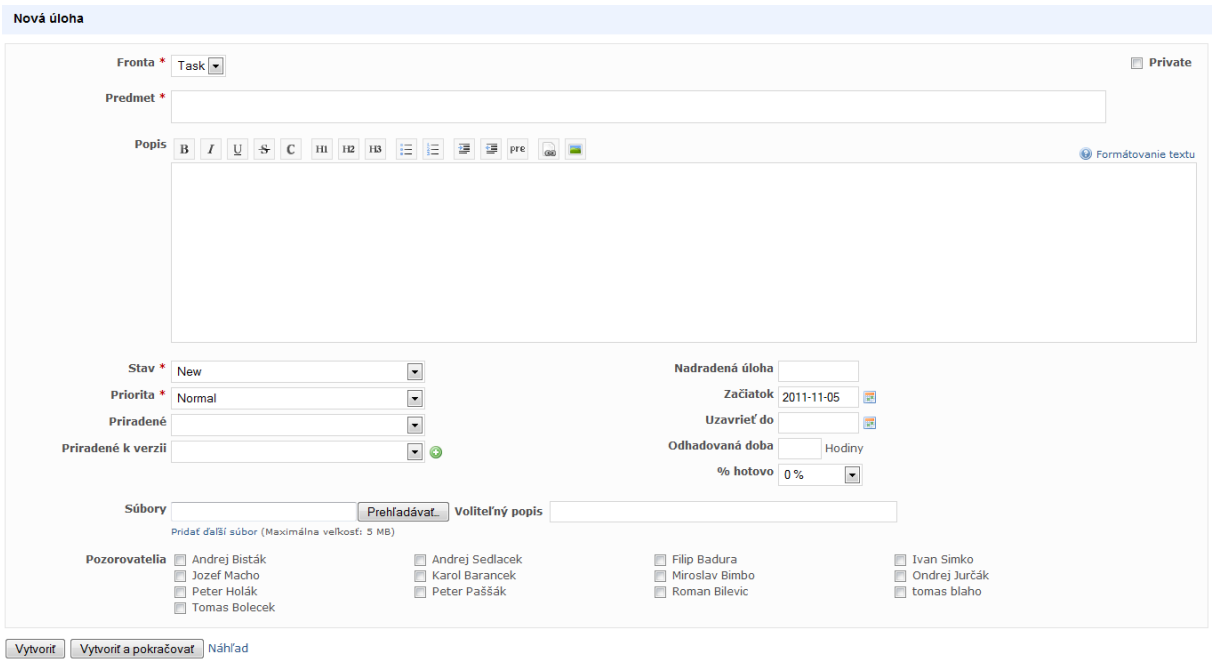
Výstup: zoznam čiastkových úloh

Zodpovedná osoba: projektový manažér

Zo zoznamu zadefinovaných hlavných cieľov a z vykonanej analýzy sú vytvorené čiastkové úlohy, ktorých úspešné zvládnutie vedie k úspešnému projektu.

Počas Tímového projektu sú pre prvý semester stanovené čiastkové úlohy – nainštalovanie potrebných nástrojov pre prácu s robotom a systému na manažment projektu, analýza iných projektov, návrh možných vylepšení a ich implementácia.

Správa úloh je vykonávaná v prostredí Redmine. Umožňuje jednoduché a prehľadné vytvorenie úloh vyplnením polí ako typ, predmet, popis, či vybraním konkrétnych riešiteľov (viď obr. 1).



Obr. 1 Vytvorenie novej úlohy v prostredí Redmine

5.4 Naplánovanie šprintov

Vstup: zoznam čiastkových úloh, zoznam cieľov

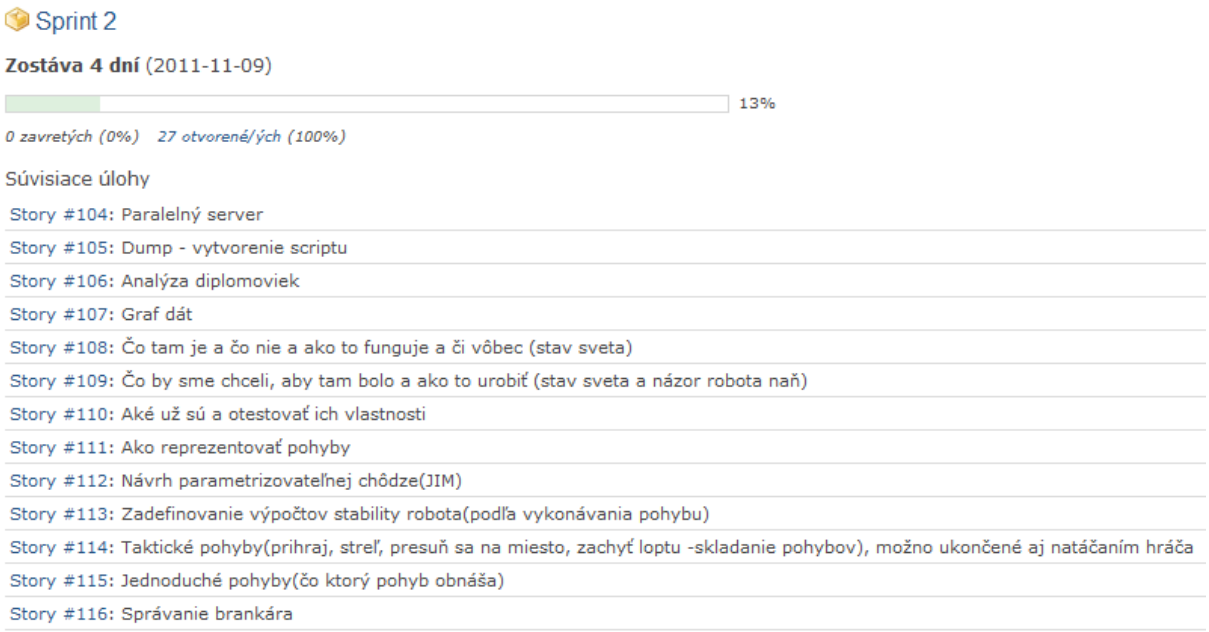
Výstup: plán šprintov

Zodpovedná osoba: projektový manažér, manažér plánovania

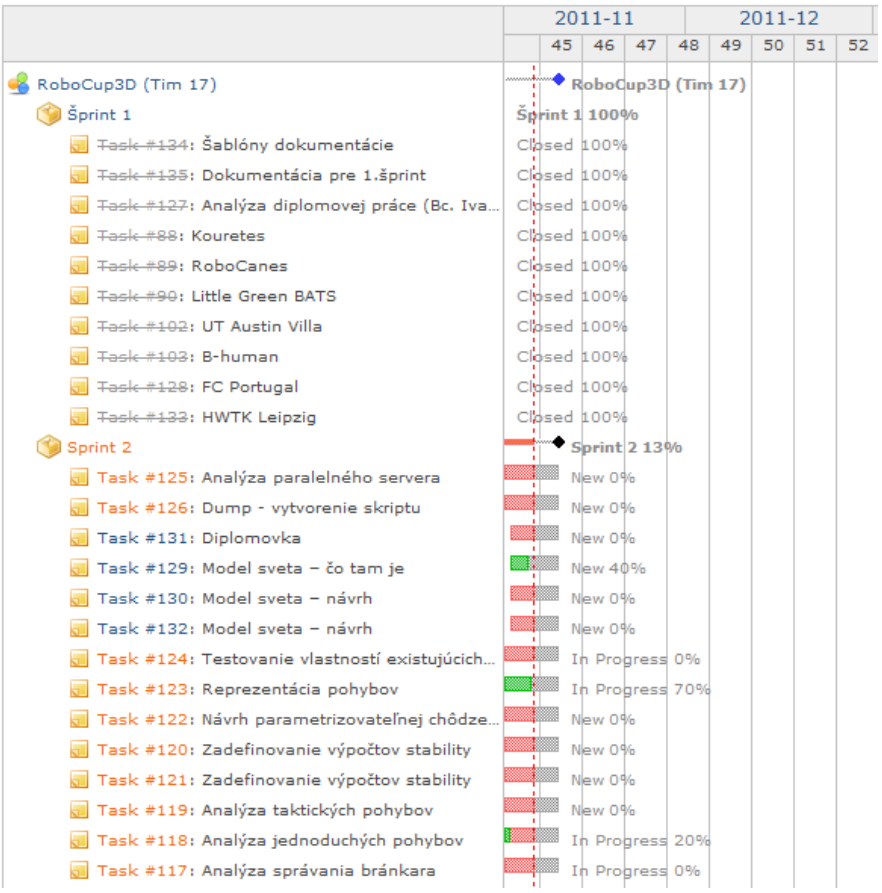
Zo vzniknutého zoznamu hlavných cieľov projektu a následného zoznamu čiastkových úloh sú tieto úlohy rozvrhnuté do šprintov.

V rámci prvého semestra je určených predbežne 6 šprintov, ktoré sú priebežne naplňované úlohami. Do prvého šprintu boli dané základné úlohy, ako nainštalovanie potrebných vecí a analýza existujúcich riešení. V druhom šprinte sú úlohy na bližšiu analýzu fungovania simulačného prostredia a návrh možných vylepšení. Do tretieho šprintu sú zaradené úlohy na detailnejší návrh vylepšení a ich čiastočná implementácia. Úlohy do ďalších šprintov sú navrhované priebežne na spoločných stretnutiach tímu a vedúceho.

Redmine poskytuje prehľadné zobrazenie jednotlivých šprintov s ich úlohami a ich plnením (viď obr. 2) ako aj prehľadné zobrazenie plnenia úloh vo forme Ganttovho grafu (viď obr. 3).



Obr. 2 Prehľad šprintu



Obr. 1 Vytvorenie novej úlohy v prostredí Redmine

Manažment verzií

Manažment verzií – správa viacerých vetiev projektu

Metodika

Úvod

Táto metodika stanovuje pravidlá a procesy pre správu verzií projektu vyvíjanom tímom 17 na predmete tímový projekt. Riadia sa ňou všetci členovia tímu pri činnostiach týkajúcich sa sprístupňovania zmien vykonaných nimi na projekte pre ostatných členov tímu. Podrobne popisuje procesy súvisiace s vytváraním a používaním viacerých vetiev, ako aj pravidlá, pomocou ktorých sa má určovať kedy je vhodné vytvoriť novú vetvu, rozhodovanie o tom, kedy do ktorej vetvy zapisovať a prípadný prenos obsahu medzi jednotlivými vetvami.

Pojmy

- SVN – Subversion, systém na správu verzií. Tento dokument predpokladá používanie verzie 1.7.1
- repozitár (repository) – úložisko všetkých súborov s ich kompletnou históriou, umiestnené na serveri
- vetva (branch) – oddelená skupina objektov, ktoré už raz v projekte existujú, nad ktorými súčasne môže prebiehať vývoj odlišným smerom
- konflikt – prípad, keď rôzni členovia tímu vykonali zmeny nad tým istým objektom a systém tieto zmeny nedokáže navzájom zlúčiť[1]
- pracovná kópia (working copy) – lokálna kópia súborov z repozitára v špecifickom čase alebo verzii. Všetka práca vykonaná na súboroch v repozitári je najprv vykonaná na ich pracovnej kópii[1]
- commit – akcia zápisu alebo zlúčenia zmien vykonaných nad pracovnou kópiou späť do repozitára [1]
- kmeň (trunk) – skupina objektov ktorá nie je vetva, prebieha nad ňou hlavná časť vývoja
- tag – označenie vybranej významnej revízie projektu
- hlava (head) – najnovšia verzia daného objektu

Roly a zodpovednosti

Prehľad

Rola	Zodpovednosti
Správca repozitára	- vytvorenie repozitára - zabezpečenie prevádzky servera - vytvorenie vetvy na základe požiadavky od projektového manažéra
Projektový manažér	- rozhodnutie kedy sa vytvorí nová vetva - rozhodnutie čo musí splnené pre pridanie tagu k určitej revízii
Vývojár	- commituje zmeny vykonané na svojej pracovnej kópií - rieši konflikty vzniknuté pri commitoch

Správca repozitára

Má na starosti riešenie všetkých situácií súvisiacich s prevádzkou SVN servera, zabezpečenie jeho chodu a dostupnosti, odstraňovanie problémov, pravidelné zálohovanie dát a plnenie požiadaviek projektového manažéra týkajúcich sa vytvárania a správy vetiev

Projektový manažér

Robí rozhodnutia týkajúce sa verziovania určitých revízií projektu. Určuje plán vydávania významných verzií. Takisto rozhoduje o tom, kedy je potrebné vytvoriť novú vetvu projektu, čo v nej bude vyvíjané a prípadný následný presun hotovej časti do kmeňa.

Vývojár

Člen tímu pracujúci na objektoch spravovaných pomocou SVN, najčastejšie zdrojovom kóde. Každý z vývojárov má u seba pracovnú kópiu, nad ktorou vykonáva zmeny. Túto pravidelne aktualizuje a zapisuje svoje zmeny späť do repozitára. Ak v prípade konfliktu nedokáže tento konflikt vyriešiť sám (úpravou svojich zmien tak, aby boli zmenami relatívne k novej verzii objektu), kontaktuje autora konfliktnej časti a vykoná kroky potrebné na vyriešenie (úprava kódu, prispôsobenie zvyšných častí.

Podrobný opis krokov

Táto kapitola obsahuje podrobný opis krokov procesov týkajúcich sa správy viacerých vetiev projektu. Presne popisuje pravidlá, ktorými sa riadi rozhodovanie o tvorbe vetiev a ich používaní, jednotlivé kroky ktorými sa toto dosahuje a samotné vykonanie týchto krokov v prostredí nástroja Subversion.

Vytvorenie vetvy

Prehľad

1. Požiadavka na vytvorenie novej vetvy
2. Rozhodnutie o vytvorení novej vetvy
3. Špecifikácia vlastností vetvy
4. Realizácia tvorby vetvy na serveri

Požiadavka na vytvorenie novej vetvy

Zodpovedný: vývojár, projektový manažér

Výstup: špecifikovaný dôvod prečo je potrebné vytvoriť novú vetvu

Hlavnou požiadavkou pre vytvorenie samostatnej vetvy je špecifikovanie potreby jej existencie – určenie dôvodu, prečo by mala byť vytvorená. Takýto podnet sa predpokladá v dvoch hlavných prípadoch:

- podnet od vývojára – v prípade že vyvíjaná súčasť by mohla narušiť vývoj iných častí projektu (dočasným znefunkčnením)
- podnet od projektového manažéra – v prípade že je potrebná alternatívna verzia projektu, ktorá sa vlastnosťami líši od hlavnej – napríklad obsahuje niektoré experimentálne vlastnosti

Požiadavka je zadaná vytvorením úlohy v systéme Redmine.

Rozhodnutie o vytvorení novej vetvy

Zodpovedný: projektový manažér

Vstup: požiadavka na vytvorenie novej vetvy v systéme Redmine

Výstup: požiadavka s nastaveným stavom na schválený alebo zamietnutý a uvedeným dôvodom

Projektový manažér posúdi vhodnosť existencie samostatnej vetvy na základe požiadavky. Ak sa rozhodne požiadavku zamietnuť, tento proces ďalej nepokračuje. Inak sa pokračuje krokom 3.

Špecifikácia vlastností vetvy

Zodpovedný: vývojár, projektový manažér

Vstup: schválená požiadavka na vytvorenie novej vetvy

Výstup: dokument špecifikujúci názov vetvy, dôvod jej existencie a pokyny pre vývojárov

Autor požiadavky na vytvorenie vetvy musí určiť jej názov a popísať čo má byť výsledkom

práce na tejto vetve. Takisto sa po ňom požaduje spísať čo patrí a nepatrí do tejto vetvy. Na základe tohto sa bude neskôr rozhodovať o tom, do ktorej vetvy patria ktoré zmeny. Tento dokument musí byť umiestnený na projektovej wiki.

Realizácia tvorby vetvy na serveri

Zodpovedný: správca repozitára
Vstup: dokument špecifikujúci názov vetvy, dôvod jej existencie a pokyny pre vývojárov
Výstup: uzavretie požiadavky v systéme Redmine, nastavenie jej stavu na vyriešený, existencia vytvorenej vetvy v repozitári na serveri

Správca repozitáru vytvorí na serveri vetvu pomocou príkazu

```
svn copy http://vm08.ucebne.fiit.stuba.sk/repo/robocup/trunk \
  http://vm08.ucebne.fiit.stuba.sk/repo/robocup/branches/[názov-vetvy] \
  -m "Vytvorena vetva [názov-vetvy]"
```

S dosadeným názvom vetvy

Výber správnych vetiev

Prehľad

1. Preskúmanie plánovaných zmien
2. Rozhodnutie

Preskúmanie plánovaných zmien

Zodpovedný: vývojár
Vstup: popis úlohy v systéme Redmine
Výstup: približný prehľad zmien ktoré bude potrebné vykonať pre splnenie úlohy
Pred začatím práce na úlohe, ktorá bola danému vývojárovi pridelená v systéme Redmine vytvorí vývojár zoznam častí, v ktorých predpokladá potrebu zmien pre splnenie úlohy.

Rozhodnutie

Zodpovedný: vývojár
Vstup: približný prehľad zmien ktoré bude potrebné vykonať pre splnenie úlohy
Výstup: zoznam vetiev pre ktoré budú vykonané zmeny
Vývojár si prezrie stav jednotlivých vetiev a posúdi, či je vhodné a možné zmenu integrovať do danej vetvy. Rozhodne sa kladne v prípade ak sú splnené všetky nasledujúce podmienky:

- úloha sa buď priamo týka zamerania danej vetvy, alebo sa jedná o opravu chyby, ktorá sa okrem kmeňa vyskytuje aj v danej vetve

- ak sa úloha netýka priamo zamerania danej vetvy, časť ktorá bude menená sa nesmie líšiť od kmeňa natoľko, aby si to splnenie úlohy vyžadovalo významne množstvo práce navyše

Spojenie vetiev

Prehľad

1. Rozhodnutie o skončení práce na vetve
2. Synchronizácia vetvy s kmeňom
3. Riešenie prípadných konfliktov
4. Spojenie vetvy s kmeňom
5. Zmazanie vetvy

Rozhodnutie o skončení práce na vetve

Zodpovedný: vývojár
Vstup: vetva so splnenými všetkými úlohami
Výstup: rozhodnutie o spojení vetvy s kmeňom
Po naplnení dôvodu existencie vetvy – dokončením časti projektu kvôli ktorej bola vytvorená – sa vývojári pracujúci na vetve rozhodnú skončiť prácu na nej a vrátiť všetky vykonané zmeny do kmeňa. Pre toto rozhodnutie je potrebné, aby boli v systéme Redmine uzavreté všetky úlohy týkajúce sa tejto vetvy. Vývojár ktorý požadoval vytvorenie vetvy má za úlohu vykonať všetky nasledujúce kroky vedúce k jej spojeniu.

Synchronizácia vetvy s kmeňom

Zodpovedný: vývojár
Vstup: rozhodnutie o spojení vetvy s kmeňom
Výstup: vetva obsahujúca zmeny vykonané v kmeni od jej vytvorenia, pracovná kópia pripravená na spojenie

Do vetvy je treba integrovať zmeny vykonané v kmeni pomocou príkazu

```
svn merge ^/robocup/trunk
```

Okrem toho sa treba ubezpečiť, že kmeň pracovnej kópie v sebe neobsahuje žiadne dodatočné zmeny oproti aktuálnej revízii. Ak toto nie je splnené, treba aktualizovať pracovnú kópiu. V prípade že sa nevyskytnú konflikty sa pokračuje krokom 4. popísaným v kapitole 5.3.5.

Riešenie prípadných konfliktov

Zodpovedný: vývojár

Vstup: vetva obsahujúca konflikty pre ktoré nie je možné integrovať zmeny vykonané v kmeni

Výstup: pracovná kópia pripravená na spojenie

Riešenie konfliktov sa riadi procesom z kapitoly 4.5

Spojenie vetvy s kmeňom

Zodpovedný: vývojár

Vstup: pracovná kópia pripravená na spojenie

Výstup: vetva spojená s kmeňom, požiadavka na zmazanie vetvy v systéme Redmine

Vývojár spustí nad svojou pracovnou kópiou v adresári kmeňa príkazy

```
svn update
svn merge --reintegrate ^/robocup/branches/[názov-vetvy]
svn commit -m "Vetva [názov-vetvy] spojena s kmenom"
```

Následne zadá do systému Redmine požiadavku na zmazanie vetvy

Zmazanie vetvy

Zodpovedný: správca repozitára

Vstup: požiadavka na zmazanie vetvy v systéme Redmine

Výstup: nastavenie stavu požiadavky v systéme Redmine na vyriešený

Správca repozitára spustí na serveri príkaz:

```
svn delete ^/robocup/branches/[názov-vetvy] \
  -m "Vetva spojena s kmenom"
```

Vetva následne prestane existovať, no jej kompletná história ostane zachovaná

Použitá literatúra

[1] B. Collins-Sussman, B. W. Fitzpatrick, C. M. Pilato: Version Control with Subversion, http://svnbook.red-bean.com/en/1.7/svn-book.pdf