SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

TÍMOVÝ PROJEKT

Monitorovanie programátora v IDE

Autori:
Bc.Michal JURANIY
Bc.Ivan KOŠDY
Bc.Jozef MARCIN
Bc.Tomáš MARTINKOVIČ
Bc.Matej NOGA
Bc.Ján PODMAJERSKÝNO
Bc.Juraj RABČAN

Školiteľ:
Doc. Mgr. Daniela CHUDÁ, PhD.

2013/2014

Obsah

1 Úvod

Tento dokument predstavuje dokumentáciu k riadeniu projektu Monitorovanie programátora v IDE. Cieľom projektu je vytvrenie webovej aplikácie, ktorá umožní monitorovanie porogramátora tým, že loguje jeho aktivitu. Tento projekt prebieha na Slovenská technickej univerzite v Bratislave konkrétne na Fakulte informatiky a informačných technológií v rámci predmetu Tímový projekt I.

2 Ponuka

2.1 Tím

Náš tím pozostáva najmä zo študentov, ktorí absolvovali bakalárske štúdium na FIIT STU, ale nové pohľady vnáša absolvent bakalárskeho štúdia na FIT ČVUT v Prahe. Snažíme sa pracovať vždy naplno nielen v škole, ale aj pri mimoškolských aktivitách.

Michal Juranyi

Absolvent bakalárskeho študijného programu Informatika na FIIT STU. V bakalárskej práci som sa venoval generovaniu dokumentácie zo zdrojových kódov, pri čom som sa stretol so syntaktickou analýzou kódu a softvérovými metrikami. Popri štúdiu sa venujem vývoju webových aplikácií v jazykoch Python (Django) a PHP oživených JavaScriptom s použitím databázových systémov MySQL (MariaDB) a PostgreSQL.

Ivan Košdy

Absolvent bakalárskeho štúdia na FIT ČVUT v Prahe, odbor Web a multimédia. Zaujímajú ma moderné webové technológie a vývoj mobilných aplikácií. Medzi praktické skúsenosti získané pri štúdiu a zamestnaní môžem zaradiť napríklad programovanie pre platformu Android, Java framework Netty, Node.js či MongoDB.

Jozef Marcin

Absolvent bakalárskeho študijného programu Informatika na FIIT STU. Obsahom mojej bakalárskej práce bolo spracovanie požiadaviek pre vytvorenie rozvrhu v systéme Moodle. Pracoval som s technológiami PHP, Pearl, Moodle, WAMP. Ďalšie skúsenosti, ktoré som získal počas štúdia - Java, C, C++, Assembler, MySQL, Postgre.

Tomáš Martinkovič

Bakalárske štúdium som absolvoval na FIIT STU v odbore informatika. V bakalárskej práci s názvom Vplyv kúskovania textových dokumentov na ich podobnosť som sa venoval porovnávaniu metód kúskovania textových dokumentov a výslednému vyhodnoteniu ich podobnosti. Počas môjho štúdia som nadobudol skúsenosti s programovacími jazykmi Java, C, Lisp, Prolog. Taktiež mám skúsenosti s Assembler, UML, MySQL.

Matej Noga

Bakalárske štúdium som absolvoval na FIIT STU v odbore informatika. V bakalárskej práci som pracoval na personalizovanom odporúčacom systéme. V práci sa venujem programovaniu v Jave.

Ján Podmajerský

Absolvoval som bakalárske štúdium na FIIT v odbore Informatika. Počas štúdia som sa naučil orientovať sa v najnovších technológiách a najmä som sa zlepšil v programovaní. Získal som skúsenosti s Asemblerom, C, C++, Java, PHP, MySQL. V bakalárskej práci som sa venoval spracovaniu obrazu, v aplikácii interaktívnej tabule, ktorá bola naprogramovaná v C++ za pomoci OpenCV.

Juraj Rabčan

Bakalárske štúdium som ukončil na FIIT STU v odbore informatika. Počas štúdia som získal skúsenosti s prácou v programovacích jazykoch C, Assembler a Java. Okrem týchto mám skúsenosti databázovými systémami MySQL a Mongo. V bakalárskej práci som sa venoval vyhľadávamiu motívov v reči. Zo systémov na riadenie verzií mám skúsenosti s Git a s TFS.

3 Úlohy členov tímu

3.1 Manažérske úlohy členov tímu

Meno	Rola
Michal Jurayi	Manažér podpory vývoja
Ivan Košdy	Manažér rizík
Jozef Marcin	Manažér monitorovania projektu
Tomáš Martinkovič	Manažér rozvrhu a plánovania
Matej Noga	Manažér kvality
Ján Podmajerský	Manažér tímu
Juraj Rabčan	Manažér tvorby dokumentácie

3.2 Krátkodobé úlohy

- *Bc. Jozef Marcin*
  - Anlýza logrov
  - Analýza metód rozpoznávania
  - Analýza Eclipse pluginov pre SW metriky
  - Analýza jednotlivých prvkov vektora
  - Imlementácia porovnávania vektorov
- *Bc. Ivan Košdy*
  - Web stránka
  - Konfigurácia Springu
  - Registrácia používateľov
  - Volanie webovej služby Gratexu
  - Spring security
  - Debug okno na prezeranie údajov
- *Bc. Ján Podmajerský*

  - Prihláška na TP Cup
  - Analýza možnosti vizualizovať dáta
  - Implementácia grafu

- *Bc. Matej Noga*
  - Implementácia vytvorenia modelu používateľa
  - Vytvorenie agregovaného vektora pre používateľa
  - Analýza PerCinika a Uacy
  - Analýza a príprava jUnit testov

- *Bc. Michal Juranyi*
  - Inštalácia RedMine
  - Správa servera
  - Inštalácia databázy na server
  - Inštalácia Glassfishu na server a nasadenie projeku na server

- *Bc. Tomáš Martinkovič*
  - Analýza metrík
  - Implementácia LOC metriky
  - Prihláška na TP Cup
  - Vytvorenie šablóny pre písanie zápisníc

- *Bc. Juraj Rabčan*
  - Vytvorenie šablóny pre písanie inžinierskeho diela
  - Vytvorenie šablóny pre písanie dokumentácir riadenia
  - Debug okno na prezeranie user modelov
  - Implentácia filtra pre zobrazenie logov a vektorov
  - Analýza PerConika a UACY

3.3 Podiel členov na jednotlivých fázach dokumentácie

3.3.1 Dokumentácia k inžinierskemu dielu

- *Bc. Jozef Marcin*
  - kapitola 3.1
  - kapitola 3.2
  - kapitola 2.1.1
- *Bc. Ivan Košdy*
- kapitola 2.3
- *Bc. Ján Podmajerský*
  - kapitola 4.1
  - kapitola 4.3
- *Bc. Matej Noga*
  - kapitola 4.1
- *Bc. Michal Juranyi*
  - kapitola 4.4
  - kapitola 4.8
- *Bc. Tomáš Martinkovič*
  - kapitola 2.1.1
- *Bc. Juraj Rabčan*
  - kapitola 1
  - kapitola 2.1.3
  - kapitola 2.2
  - kapitola 3.3

  - kapitola 3.4
  - kapitola 3.5
  - kapitola 3.6
  - kapitola 3.7
  - kapitola 4.2
  - kapitola 4.5
  - kapitola 4.6
  - kapitola 4.7

3.3.2 Dokumentácia k riadeniu

- *Bc. Jozef Marcin*
  - Metodika pre reportovanie postupu prác na projekte (kapitola 9)
  - Manažment monitorovania projektu (kapitola 13)
  - Prílohy: Zápis zo 7. stretnutia tímu č. 8
  - Ponuka (kapitola 2)
- *Bc. Ivan Košdy*
  - Metodika pre prácu s komponentami frameworku Spring a šablónami JSP Dedikácia (kapitola 8)
  - Manažment rizík (kapitola 18)
  - Ponuka (kapitola 2)
- *Bc. Ján Podmajerský*
  - Ponuka (kapitola 2)
  - Prílohy: Zápis zo 4. stretnutia tímu č. 8
  - Téma: Monitorovanie programátora v IDE (kapitola 4)
  - Metodika komunikácie (kapitola 7)
  - Manažment komunikácie (kapitola 16)

  - Ponuka (kapitola 2)

- *Bc. Matej Noga*
  - Prílohy: Zápis z 3. stretnutia tímu č. 8
  - Metodika manažmentu testovania pomocou nástroja JUnit Test(kapitola 10)
  - Manažment kvality (kapitola 19)
  - Ponuka (kapitola 2)

- *Bc. Michal Juranyi*
  - Prílohy: Zápis zo 6. stretnutia tímu č. 8
  - Metodika manažmentu verzií (kapitola 12)
  - Manažment podpory vývoja (kapitola 14)
  - Ponuka (kapitola 2)

- *Bc. Tomáš Martinkovič*
  - Prílohy: Zápis z 2. stretnutia tímu č. 8
  - Ponuka (kapitola 2)
  - Téma: Monitorovanie programátora v IDE (kapitola 4)
  - Metodika pre priebeh tímového stretnutia a pridanie noveh úlohy do Redmine (kapitola 11)
  - Manažment rozvrhu a plánovania
  - Ponuka (kapitola 2)

- *Bc. Juraj Rabčan*
  - Plán projektu (kapitola 5)
  - Metodika pre tvorbu dokumentácie (kapitola 6)
  - Editácie metodík
  - Editácie zápisníc

- Úvod (kapitola 1)
- Ponuka (kapitola 2)
- Úlohy členov tímu (kapitola 3)
- Manažment tvorby dokumentácie (kapitola 15)
- Prílohy: Zápis z 5. stretnutia tímu č. 8

3.4 Preferencie tímu

Poradie	Názov témy
1	Interaktívne hry na mobile s multimediálnym obsahom (mobiHRA)
2	Digital SweatShop (SweatShop)
3	Analýza výsledkov výskumu (Research RANK)
4	Webový komunitný systém otázok a odpovedí (CQA)
5	Zábavný systém pre spolucestujúcich v automobile (AUTO Interakcia)
6	Prehliadka kódov v tímových projektoch (CodeReview)
7	Virtuálna FIIT na mobile (Virtualna FIIT)
8	Distribuované počítanie na FIIT (FIIT grid)
9	Monitor programátora v IDE (IDE Monitor)
10	Vizualizácia informácií v obohatenej realite (Vizualizacia)
11	Sledovanie pohľadu pri používaní aplikácií (GAZE usability)
12	Trojdimenzionálne UML (3D UML)
13	Robotický futbal (RoboCup)

Obr. 1: *Preferencie tímu*

Napriek odovzdaným preferenciáam sa náš tím v Yonbanne uchádzal len o nasledovné témy:

- Interaktívne hry na mobile s multimediálnym obsahom
- Analýza výsledkov výskumu (Research RANK)
- Monitor programátora v IDE (IDE Monitor)

4 Téma: Monitorovanie programátora v IDE

V súčasnosti veľmi veľa programátorov píše zdrojové kódy bez toho, aby si uvedomovali, či ich naprogramovaný zdrojový kód je kvalitný. Nemajú prehľad o tom ako sú pri písaní zdrojového kódu efektívni a viackrát nedostatočne využívajú znovupoužiteľnosť zdrojových kódov. Ak by sme tieto činnosti správne merali a objektívne vyhodnocovali, dokázali by sme vhodne motivovať programátorov k lepším výkonom pri písaní zdrojových kódov.

Rovnako by aj manažéri IT firiem ocenili, ak by mali prehľad o výkone svojich zamestnancov v pracovnom prostredí, o ich kvalite a efektívnosti programovania. Následne by ich na základe týchto vedomostí, o ich spôsobe programovania, dokázali patrične ohodnotiť. Vedeli by lepšie vybrať vhodných programátorov na vývoj daného softvérového produktu, čím by dokázali efektívnejšie a kvalitnejšie pracovať v rámci softvérového projektu.

Programátor pracujúci v IDE vykonáva množstvo rôznych operácií a akcií, ktoré sú pre neho charakteristické ako napríklad orientácia v systéme, štruktúra zdrojového kódu, pomenovávanie premenných, metód, tried, balíkov, využívanie akcií, a pod. Z týchto charakteristík sa dá vytvoriť model konkrétneho používateľa. Aby bolo možné vytvárať modely používateľov a následne vyhodnocovať kvalitu a efektívnosť programátora je potrebné, aby sme najskôr identifikovali používateľa na základe práce s myšou a klávesnicou.

Zdrojové kódy, vzory kódov, či ich časti, je možné automaticky vyhľadávať rôznymi metódami. Ich vyhľadávanie je možné využiť na identifikáciu znovupoužiteľnosti zdrojového kódu, určenie podobných zdrojových kódov, na hľadanie vzorov, na prieskum pokrytia tematiky, či iné spôsoby. V mnohých softvérových projektoch sa veľakrát zbytočne nachádzajú časti zdrojových kódov z webových zdrojov. Takéto využívanie a preberanie zdrojových kódov z webu neprispieva ku kvalite programátora.

5 Plán projektu

Práca na projekte je realizovaná agilnou metódou SCRUM. SCRUM je inkrementálny a iteratívny prístup k vývoju softvéru. Podstatou SCRUMU sú šprinty. Šprinty sú vopred určené časové jednotky, počas ktorých sa pracuje na pridelených úlohách, ktoré sa majú vykonať počas prebiehajúceho šprintu. Úlohy sa prideľujú na začiatku šprintu. Naše šprinty majú dĺžku dva týždne. Vyhodnotenie úloh sa robí na tímových stretnutiach a tiež sa tam definujú nové úlohy.

Harmonogram môžeme vidieť na nasledujúcej tabuľke.

Týždeň semestra	Plán
1	Odovzdanie zoznamu kompetencií, pridelenie témy
2	oboznámenie sa s projektom, plánovanie prvého šprintu
3	práca na prvom šprinte
4	ukončenie prvého šprintu, stanovenie úloh do druhého šprintu
5	práca na druhom šprinte
6	ukončenie druhého šprintu, stanovenie úloh do tretieho šprintu
7	práca na tretiom šprinte
8	ukončenie tretirho šprintu, stanovenie úloh do štvrtého šprintu
9	práca na štvrtom šprinte, priebežné odovzdanie dokumentácie
10	koniec štvrtého šprintu, stanovenie úloh do ďalšieho šprintu
11	práca na piatom šprinte
12	odovzdanie projektu

6 Metodika pre tvorbu dokumentácie

6.1 Úvod

Úlohou metodiky je stanovenie postupu tvorby dokumentácie inžinierského diela. Dokumentácia sa píše v typografickom systéme Latexe. Pre zjednodušenie tvorby dokumentácie bola vytvorená šablóna, ktorá mnohé veci uľahčuje a automatizuje no stále je potrebné zadefinovať postupy, ktoré nie sú úplne automatizované a tie, ktoré sa automatizovať nedajú. Táto dokumentácia je určená pre všetkých členov tímu, ktorý akýmkoľvek spôsobom prispievajú do dokumentácie. Metodika pokrýva manažment tvorby dokumentácie inžinierskeho diela na spodnej úrovni, ciže dáva presný návod ako postupovať pri tvorbe inžinierskeho diela.

6.2 Editácia šablóny

Po nainštalovaní latexu máme k dispozícií základné balíčky. Naša šablóna vyžaduje balíčky, ktoré nie sú súčasťou základnej inštalácie. Vo Windowse je nutné urobiť update balíčkov Miktexu. Následne je potrebné spustiť TexMaker editor a v ňom otvoriť súbor *main.tex* a spustiť quick build nad týmto súborom. Po vykonaní update Miktex balíčkov nám TexMaker stiahne všetky potrebné balikíčky. V Linuxe (Ubuntu/Mint) je potrebné spustiť príkaz:

```
sudo apt-get install texlive-latex3 texlive-lang-czechslovak
texlive-latex-recommended lmodern texlive-latex-extra
latex-xcolor texlive-fonts-extra texlive-fonts-recommended
texlive-generic-recommended
```

Obr 6.1. Inštalácia potrebných balíčkov v Linuxe

Následne sputíme príkaz:

```
Sudo apt-get install texmaker
```

Obr 6.2. Inštalácia TexMakeru v Linuxe

6.3 Pridanie kapitoly do dokumentácie

V našej dokumentácii sa každá kapitola píše do osobitného súboru. Súbor je pomenovaný podľa názvu kapitoly. Tím, že sa každá kapitola píše to samostatného súboru, môže viacero ľudí editovať dokumentáciu a následne ju ľahko spojiť dokopy. Každá kapitola môže mať najviac podkapitoly druhej úrovne. Súbory budú umiestnené v adresári, kde sa nachádza súbor *main.tex*. V Latexe existuje viacero príkazov na prácu s kapitolami. My budeme používať len tieto:

- \section{Názov kapitoly}
- \subsection{Názov podkapitoly prvej úrovne}
- \subsubsection{Názov podkapitoly druhej úrovne}

6.4 Spájanie kapitol do komplexného dokumentu

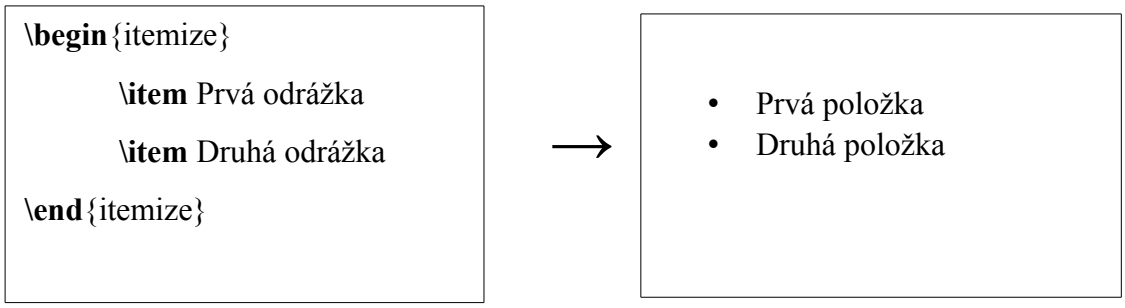
Potom ako vytvoríme súbor, kde sa bude písať kapitola musíme tento súbor začleniť do dokumentu. Do dokumentu kapitolu začleníme tak, že ju pridáme do súboru *main.tex*. Kapitolu pridáme nasledujúcim príkazom: \input{súbor s kapitolou.tex}. Kapitoly sa v dokumente objavia v tom poradí ako sú začlenené v súbore *main.tex*.

```
\begin{document}
      \input{uvod.tex}
      \input{analyza.tex}
      \input{zaver.tex}
\end{document}
```

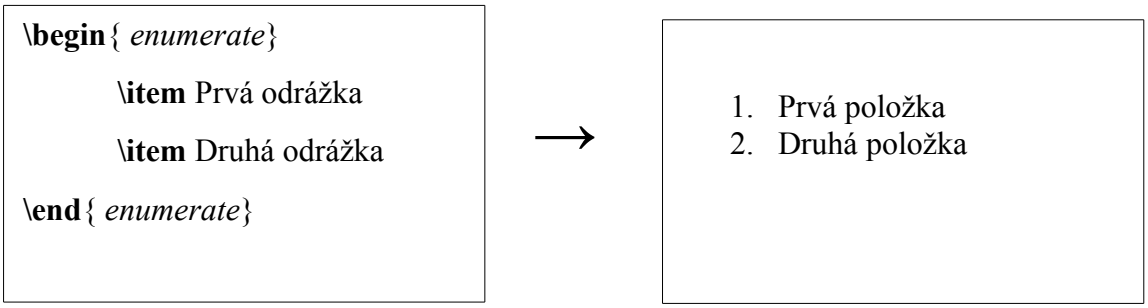
Obr 6.3. Ukážka dokumentu, ktorý obsahuje kapitoly: Úvod, Analýza a záver.

6.4 Odrážky a číslovanie

V Latexe existuje viacero spôsobov pre pridanie odrážok a číslovania. My v našej dokumentácii pre zachovanie konzistencie budem používať len jeden spôsob. Pre odrážky použijeme prostredie *itemize* a pre číslovanie použijeme prostredie *enumerate*. V prostredí budeme používať len tag *item*. Ako sa čísluje a odrážkuje v dokumentácii môžeme vidieť na nasledujúcich obrázkoch.



Obr 6.4. Vzorový príklad odrážok.



Obr 6.5. Vzorový príklad číslovania.

6.5 Aktualizovanie obsahu

Vo Windowse ak chceme aktualizovať obsah, zoznamu obrázkov a zoznam ukážok je potrebné spustiť v editore od Miktexu nad súborom main.tex:

- 2x pdflatex
- 2x bibtex
- 2x pdflatex

Pri písaní dokumentácie v Linuxe stačí vykonať nasledujúcu sériu príkazov:

- cd cesta_dokumentácii
- pdflatex main.tex; pdflatex main.tex; bibtex main; bibtex main; pdflatex main.tex; pdflatex main.tex;

Obr 6.6 aktualizovanie obsahu v Linuxe

6.6 Listing zdrojových kódov

V Latexe existuje množstvo spôsobov pre listing zdrojových kódov. My budeme používať len jeden spôsob a to prostredie *lstlisting*. Pre automatické zvýraznenie syntaxe zdrojového kódu použijeme príkaz \lstset{language=Java}. Zdrojový kód vložíme medzi príkazy *\begin{lstlisting}[frame=single]* a *\end{lstlisting}*.

```
\lstset{language=Java}
\begin{lstlisting}[frame=single]
        package sk.stuba.fiit.idem.controller;
        public class Class {
                public String printString(String output) {
                        int i = 0;
                        for(int i = 0; i<10; i++){
                                System.out.println(output);
                        }
                }
        }
```

Obr. 6.7 Ukážka ako vyzerá zdrojový kód v Latex súbore našej dokuemntácie.

```
package sk.stuba.fiit.idem.controller;

public class Class {
        public String printString(String output) {
                int i = 0;
                for(int i = 0; i<10; i++){
                        System.out.println(output);
                }
        }
}
```

Obr. 6.8 *Ukážka ako vyzerá zdrojový kód vo vygenerovanom pdf súbore.*

6.7 Číslovanie strán

Jednotlivé strany dokumentácie musia byť očíslované. Nastavenie šablóny čísluje strany automaticky, avšak číslovanie strán v prílohe je potrebné dodatočné nastavenie. Nastavenie číslovania sa výkonná nasledovne:

- Pred každú prílohu vložíme nasledujúci kód:

```
\newpage
\setcounter{page}{1}
\renewcommand{\thepage}{PREFIX-\arabic{page}}
\newpage
```

Obr.6.9 Nastavenie číslovania v prílohe.

- Kód zabezpečí číslovanie pre prvú stranu PREFIX -1, druhú stranu PREFIX -2 a pod.
- Pre zmenu prefixu číslovanie je potrebné prepísať "PREFIX" na požadovaný tvar.

6.8 Vkladanie obrázkov

Všetky obrázky, ktoré sa nachádzajú v dokumentácií sa ukladajú do adresára */figures*. Obrázky vždy vkladáme nasledujúcim kódom, pričom vždy zvolíme vhodnú scale, tak aby sa obrázok zmestil na požadovanú pozíciu. Text, ktorý vložíme do *caption* bude zobrazený ako popis v zozname obrázkov.

```
\begin{figure}[H]
        \begin{center}
                \includegraphics[scale=1]{cesta_k_obrázku}
                \caption{Popis obrázku}
                \label{label obrázka}
        \end{center}
\end{figure}
```

Obr.6.*10* Vkladanie obrázka.

Následne sputíme príkaz:

Sudo apt-get install texmaker

Obr 6.2. Inštalácia TexMakeru v Linuxe

6.3 Pridanie kapitoly do dokumentácie

V našej dokumentácii sa každá kapitola píše do osobitného súboru. Súbor je pomenovaný podľa názvu kapitoly. Tím, že sa každá kapitola píše to samostatného súboru, môže viacero ľudí editovať dokumentáciu a následne ju ľahko spojiť dokopy. Každá kapitola môže mať najviac podkapitoly druhej úrovne. Súbory budú umiestnené v adresári, kde sa nachádza súbor *main.tex*. V Latexe existuje viacero príkazov na prácu s kapitolami. My budeme používať len tieto:

- \section{Názov kapitoly}
- \subsection{Názov podkapitoly prvej úrovne}
- \subsubsection{Názov podkapitoly druhej úrovne}

6.4 Spájanie kapitol do komplexného dokumentu

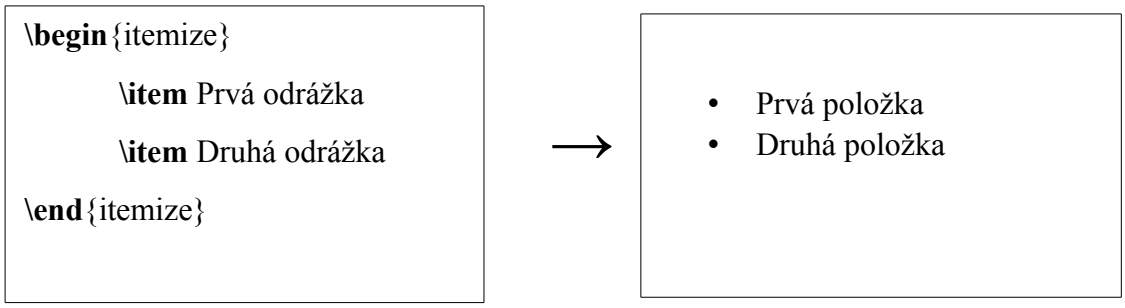
Potom ako vytvoríme súbor, kde sa bude písať kapitola musíme tento súbor začleniť do dokumentu. Do dokumentu kapitolu začleníme tak, že ju pridáme do súboru *main.tex*. Kapitolu pridáme nasledujúcim príkazom: \input{súbor s kapitolou.tex}. Kapitoly sa v dokumente objavia v tom poradí ako sú začlenené v súbore *main.tex*.

```
\begin{document}
      \input{uvod.tex}
      \input{analyza.tex}
      \input{zaver.tex}
\end{document}
```

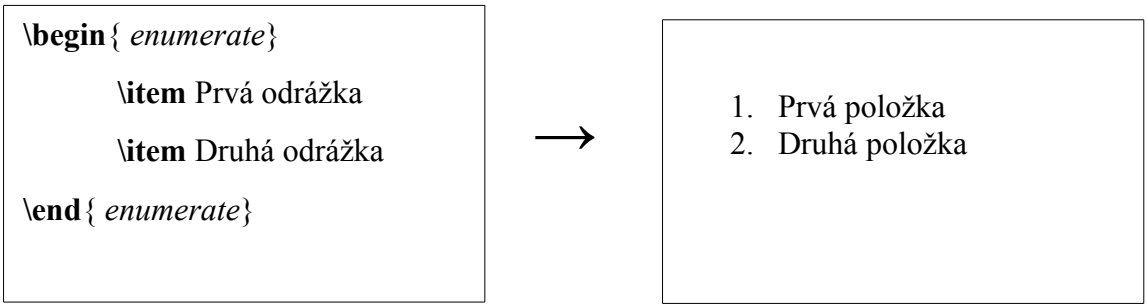
Obr 6.3. Ukážka dokumentu, ktorý obsahuje kapitoly: Úvod, Analýza a záver.

6.4 Odrážky a číslovanie

V Latexe existuje viacero spôsobov pre pridanie odrážok a číslovania. My v našej dokumentácii pre zachovanie konzistencie budem používať len jeden spôsob. Pre odrážky použijeme prostredie *itemize* a pre číslovanie použijeme prostredie *enumerate*. V prostredí budeme používať len tag *item*. Ako sa čísluje a odrážkuje v dokumentácii môžeme vidieť na nasledujúcich obrázkoch.



Obr 6.4. Vzorový príklad odrážok.



Obr 6.5. Vzorový príklad číslovania.

6.5 Aktualizovanie obsahu

Vo Windowse ak chceme aktualizovať obsah, zoznamu obrázkov a zoznam ukážok je potrebné spustiť v editore od Miktexu nad súborom main.tex:

- 2x pdflatex
- 2x bibtex
- 2x pdflatex

Pri písaní dokumentácie v Linuxe stačí vykonať nasledujúcu sériu príkazov:

- cd cesta_dokumentácii
- pdflatex main.tex; pdflatex main.tex; bibtex main; bibtex main; pdflatex main.tex; pdflatex main.tex;

Obr 6.6 aktualizovanie obsahu v Linuxe

6.6 Listing zdrojových kódov

V Latexe existuje množstvo spôsobov pre listing zdrojových kódov. My budeme používať len jeden spôsob a to prostredie *lstlisting*. Pre automatické zvýraznenie syntaxe zdrojového kódu použijeme príkaz \lstset{language=Java}. Zdrojový kód vložíme medzi príkazy *\begin{lstlisting}[frame=single]* a *\end{lstlisting}*.

```
\lstset{language=Java}
\begin{lstlisting}[frame=single]
        package sk.stuba.fiit.idem.controller;
        public class Class {
                public String printString(String output) {
                        int i = 0;
                        for(int i = 0; i<10; i++){
                                System.out.println(output);
                        }
                }
        }
```

Obr. 6.7 Ukážka ako vyzerá zdrojový kód v Latex súbore našej dokuemntácie.

```
package sk.stuba.fiit.idem.controller;

public class Class {
        public String printString(String output) {
                int i = 0;
                for(int i = 0; i<10; i++){
                        System.out.println(output);
                }
        }
}
```

Obr. 6.8 *Ukážka ako vyzerá zdrojový kód vo vygenerovanom pdf súbore.*

6.7 Číslovanie strán

Jednotlivé strany dokumentácie musia byť očíslované. Nastavenie šablóny čísluje strany automaticky, avšak číslovanie strán v prílohe je potrebné dodatočné nastavenie. Nastavenie číslovania sa výkonná nasledovne:

- Pred každú prílohu vložíme nasledujúci kód:

```
\newpage
\setcounter{page}{1}
\renewcommand{\thepage}{PREFIX-\arabic{page}}
\newpage
```

Obr.6.9 Nastavenie číslovania v prílohe.

- Kód zabezpečí číslovanie pre prvú stranu PREFIX -1, druhú stranu PREFIX -2 a pod.
- Pre zmenu prefixu číslovanie je potrebné prepísať "PREFIX" na požadovaný tvar.

6.8 Vkladanie obrázkov

Všetky obrázky, ktoré sa nachádzajú v dokumentácií sa ukladajú do adresára */figures*. Obrázky vždy vkladáme nasledujúcim kódom, pričom vždy zvolíme vhodnú scale, tak aby sa obrázok zmestil na požadovanú pozíciu. Text, ktorý vložíme do *caption* bude zobrazený ako popis v zozname obrázkov.

```
\begin{figure}[H]
        \begin{center}
                \includegraphics[scale=1]{cesta_k_obrázku}
                \caption{Popis obrázku}
                \label{label obrázka}
        \end{center}
\end{figure}
```

Obr.6.*10* Vkladanie obrázka.

7. Metodika komunikácie

7.1 Úvod

Cieľom tohto dokumentu je oboznámiť čitateľa so spôsobom komunikácie v rámci tímu. Výsledkom by malo byť uľahčenie spolupráce a presné určenie aký kanál na čo použiť, kde hľadať veci týkajúce sa úloh. Každý člen tímu bude mať jasný prehľad ako efekívne komunikovať a kde hľadať záznamy komunikácie so zákazníkom, so spolupracovníkmi, inak sa nedá pracovať. Smeruje to k čo najlepšiemu zjednodušeniu a sprehľadneniu komunikácie v rámci tímu.

V tomto dokumente sú definované všetky štandardy, ktoré musí každý člen tímu dodržať na to aby sa dosiahlo požadované, plynulé riešenie problému.

7.2 Vzájomná komunikácia v tíme

V tíme používame komunikačné kanály, ktoré sa nachádzajú v prvom riadku nasledujúcej tabuľky. Riadky tabuľky opisujú situácie, v ktorých je vhodné použiť daný kanál, označený krížikom v bunke.

	Email	Chat	Redmine fórum	Redmine wiki	Dropbox	Google dokumenty	Osobné stretntia
Vybavovanie záležitostí s 3. stranou	x						

Riešenie malých problémov	x	x					
Riešenie rozsiahlych problémov			x				
Opis postupu				x			
Pripomienky pre ostatných členov tímu	x	x					
Navrhovanie nových funkcií			x				x
Upozornenie na chyby	x		x				
Dohody o použitých technológiách			x				x
Zápisnice					x		
Dokumentácia					x		
Výnimočné dokumenty						x	
Komunikácia so zákazníkom	x						

7.3 Detailné spôsoby komunikácie

Každý komunikačný kanál má svoje určité špecifiká, na dosiahnutie efektívnosti využívania je nutné používať ho vo všetkých vyššie spomenutých situáciách podľa dohodnutých pravidiel.

E-mail

- komunikácia vedená v slovenčine, možná aj bez diakritiky
- pri písaní emailu umiestniť do predmetu predponu [TP]
- meno predmetu začína veľkým písmenom a vždy obsahuje podstatné meno
- dĺžka emailu by nemala prekročiť 10 viet, pretože to môže viesť k ignorancii
- vyjadrovať sa stručne, vecne, technicky
- pozor na odbočovanie od témy
- pri písaní ohľadom nejakej úlohy, ktorá sa týka len jedného človeka, napísať správu iba jemu, vyhýbať sa zbytočnému otravovanie celého tímu
- e-mail nejakej tretej strane vždy píše manažér komunikácie, do kópie pridá celú skupinu
- prijatý email z tretej strany na skupinový email vybavuje manažér komunikácie
    - odpovie cez svoje konto, tak, že do kópie pridá celú skupinu
    - ak nevie daný problém riešiť, pridá úlohu do redmine, ktorá bude začínať reťazcom „Odpovedzte na email“ a popíše problém, ostatný členovia tímu budú upozornení a niekto si úlohu vyberie
    - vedený v jazyku akým bol písaný pôvodný e-mail, čiže netreba striktne dodržiavať slovenčinu
- ak sa píše o úlohách v redmine vždy pridávať linky na ne
- prílohy dávať vždy aj do dropboxu, do priečinku *Prílohy* a v ňom do nového priečinka s názvom problému – rovnakým štýlom ako predmet emailu

7.4 Chat

- používame akýkoľvek dostupný kanál: gTalk, Facebook,...
- vždy iba jednoduchý chat, nie skupinový
- komunikácia vedená v slovenčine, možná aj bez diakritiky
- komunikácia musí prebiehať rýchlo
- jednotlivé správy nesmú prekročiť dĺžku 3 viet
- vyjadrovať sa vecne
- ak komunikácia prejde k diskusii o konkrétnej úlohe, presunúť komunikáciu do redmine fóra

- tam sa nakopíruje celý chat, musí byť s toho zjavné, kedy chat prebiehal, dátum a čas

7.5 Redmine fóra

- komunikácia vedená v slovenčine, možná aj bez diakritiky
- pozor na podobné vlákna, pred vytvorením nového vždy prečítať všetky už existujúce
- ak sa jedná o diskusiu o návrhu novej úlohy pridať do názvu vlákna predponu [PreTask] a názov úlohy začínajúcim veľkým písmenom
- ak z navrhovaného tasku nakoniec task nevznikne, takéto vlákno zmazať
- ak z diskusie vznikne nová úloha, vytvoriť novú úlohu v systéme a toto diskusné vlákno uzatvoriť
- ak sa komunikácia týka úlohy pridať do názvu vlákna predponu [Task] a názov úlohy začínajúcim veľkým písmenom
- vždy diskutovať iba o problémoch týkajúcich sa témy, ak po dlhšej diskusii prirodzene vznikne nové vlákno (úlohy), vytvoriť ho

7.6 Redmine wiki

- názov vždy začína veľkým písmenom, obsahuje podstatné meno
- vedená v slovenčine, možná aj bez diakritiky
- krátky opis je spravidla jedna veta
- nadpis H1, ktorým dokument začína, je totožný s názvom stránky
- využívať odrážky, zoznamy
- vyhnúť sa dlhým, súvislým textom

7.7 Dropbox

- zápisnica z každého stretnutia musí byť vložená jej autorom do priečinku *Zapisnice*
- všetky prílohy emailov sú v priečinku *Prilohy*
- akýkoľvek softvér vkladať do priečinku *Software*
- akékoľvek pomocné texty, pomôcky, vkladať do priečinka *Pomocky*
    - tam do nových priečinkov s príhodným názvom začínajúcim veľkým písmenom, zoskupovať logicky súvisiace pomôcky

7.8 Google dokumenty

- názov dokumentu je zhodný s predpísaným názvom od zadávateľa, avšak neobsahuje medzery ani diakritiku
- každý dokument je zdieľaný všetkým členom tímu s plnými právami
- ak je dokument už hotový, všetci členovia odsúhlasili jeho obsah, tento súbor je publikovaný a uložený na dropboxe

7.9 Osobné stretnutia

- stretnutie vždy vedie jeden člen tímu, ktorý bude na budúce písať zápisnicu
- pri vedení sa strieda po týždňoch, podľa abecedy
- stretnutí sa zúčastňujú vždy všetci členovia tímu
- všetci členovia tímu otvorene diskutujú o navrhovaných problémoch
- každá navrhnutá úloha je ohodnotená scrum kartičkami
    - najnižšia a najvyššia hodnota je okomentovaná
    - náročnosť úlohy je stanovená po dohode, príp. viacnásobnom hlasovaní
- s každého stretnutia je výstup v podobe zápisnice, v dohodnutom formáte
- vládne demokracia, väčšina vyhráva

8 Metodika pre prácu s komponentami frameworku Spring a šablónami JSP Dedikácia

8.1 Úvod

Metodika určuje postupy pre programátora vytvárajúceho a modifikujúceho zdrojový kód aplikácie založenej na frameworku Spring a šablón v jazyku JSP.

Touto metodikou sa budú riadiť zainteresované osoby pri:

- vytváraní a úprave komponentov
- vytváraní, úprave a integrácií šablón

8.2 Slovník pojmov

JSP - Java Server Pages
MVC - architektúra/štýl Model-View-Controller
Cesta - relatívna URL vzhľadom ku context root
Package - balíček v jazyku Java
Controller - súčasť MVC, nazývaný aj radič
Layout - základné rozloženie stránky, ktoré obsahuje zobrazenia (views)
View - zobrazenie obsahujúce dáta pre konkrétnu funkciu

Použité značenie v príkladoch:
text - nový/modifikovaný text
text - text z už uvedeného príkladu

8.3 Postupy metodiky

8.3.1 Controllery

Tvorba nového controlleru

1. Zvolenie vhodného názvu problémovej domény, ktorú má daný controller riešiť
2. Vytvorenie novej triedy s názvom *<Názov prob. domény>Controller* v balíku *sk.stuba.fiit.idem.controllers*
3. Vytvorenie anotácií pre Spring MVC:

a. @Controller
b. @RequestMapping(value="<Názov prob. domény>")

4. Vytvorenie základného javadoc komentáru

```
package sk.stuba.fiit.idem.controller;
/**
 * Routes:
 */
@Controller
@RequestMapping("/user")
public class User {
}
```

Obr. 8.1 Ukážka kódu

8.3.2 Cesty

Tvorba novej cesty

1. Vytvorenie metódy s názvom funkcie, ktorú má vykonať
2. Vytvorenie anotácie Spring MVC @RequestMapping s potrebnými parametrami
3. Vytvorenie javadoc komentáru s informáciou o ceste s popisom RequestMappingu

```
/**
 * Routes:
 * / [GET] index() (3)
 */
...
public class User {
	@RequestMapping(method =
RequestMethod.GET) (2)
	public String index() { (1)
		return "user"; (1)
	} (1)
}
```

Obr. 8.2 Ukážka kódu

Úprava cesty
Ak nastane zmena v štruktúre cesty je potrebné túto zmenu zaznamenať

1. Upravenie anotácie
2. Upravenie javadocu metódy
3. Upravenie záznamu o ceste v javadocu controlleru
4. Vyhľadanie pôvodných výskytov cesty v šablónach
5. Upravenie všetkých výskytov pôvodných ciest na nové

8.3.3 Šablóny

Šablóny môžeme rozdeliť na tri typy:

a) základné rozvrhnutie (layout) skupiny stránok
b) zobrazenie (view) s funkciou zdieľanou viacerými stránkami
c) zobrazenie (view) vytvorené pre jedinú funkciu využitú konkrétnou stránkou

Umiestnenie a názov súborov šablón podľa typu:

a) src/main/webapp/WEB-INF/layouts/<skupina>.tag
b) src/main/webapp/WEB-INF/views/shared/<funkcia>.tag
c) src/main/webapp/WEB-INF/views/<názov controlleru>/<funkcia>.jsp

Tvorba novej šablóny:

1. Zvolíme typ použitia novej šablóny

2. Vytvoríme v zodpovedajúcom priečinku súbor s názvom funkcie, alebo skupiny stránok podľa daného typu šablón.
3. Doplnenie šablóny o povinné prvky

Rozvrhnutia

Rozvrhnutia musia obsahovať:

1. časti pre vkladanie voliteľných CSS štýlov v hlavičke
2. časti pre vkladanie JS skriptov na konci šablóny
3. potrebné komponenty frameworku Bootstrap
4. potrebné komponenty frameworku jQuery
5. hlavný súbor CSS štýlov *main.css*
6. telo rozvrhnutia

```
<%@tag description="Default page template" pageEncoding="UTF-8"%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@attribute name="footer" fragment="true"%>
<html>
<head>
	<link href="..." rel="stylesheet">
	<c:forEach items="${css}" var="cssfile">
		<link rel="stylesheet"
			href="<c:url value="/resources/css/${cssfile}" />">
	</c:forEach>
</head>
<body>
	<jsp:doBody>
	<script type="text/javascript" src="..."></script>
	<c:forEach items="${js}" var="jsscript">
		<script type="text/javascript"
			src="<c:url value="/resources/js/${jsscript}" />">
	</c:forEach>
</body>
</html>
```

Obr.8.3 Ukážka základného rozvrhnutia

Zdieľané zobrazenia

Hlavné menu

Zobrazenie hlavného menu je zdieľané zobrazenie, ktoré zjednocuje navigáciu v aplikácií.

Pri úprave hlavného menu je potrebné zachovať zoskupenie a oddelenie odkazov pre jednotlivé prístupové zóny:

a) verejná
b) privátna
c) administračná

Pätička

Zobrazenie pätičky musí obsahovať:

a) názov projektu a tímu
b) označenie aktuálnej verzie
c) kontakt na tím

8.4 Zdroje

Štruktúra umiestnenia zdrojov je daná typom zdroja.

Typ zdroja	Umiestnenie
CSS súbory	*src/main/webapp/resources/css/*
JavaScript súbory	*src/main/webapp/resources/js/*
Obrazové dáta	*src/main/webapp/resources/img/*

Hlavným CSS súborom je *main.css* (src/main/webapp/resources/css/main.css)

Pre optimalizáciu času načítavania stránok je nutné použiť CDN a zdroje Google Web Fonts:

- *Bootstrap*
  http://www.bootstrapcdn.com/

- jQuery + UI
  http://jquery.com

- Google Web Fonts
  https://www.google.com/fonts

9 Metodika pre reportovanie postupu prác na projekte

9.1 Úvod

Táto metodika je určená najmä členom tímu 08 v rámci tímového projektu. Avšak čerpať z nej môže ktorýkoľvek tím, jednotlivec, alebo spoločnosť, ktorá pracuje na softvérovom diele a využíva agilný prístup scrum a Redmine. Riadiť sa budú jednotlivci pri vytváraní reportov o úlohách v tíme.

9.2 Slovník pojmov

- Scrum – Agilný prístup k vývoju softvéru. Zameriava sa na komplexné stratégie vývoja produktov, kde vývojový tím funguje ako celok na dosiahnutie spoločného cieľa.
- Úloha –predstavuje súbor činností potrebných na dosiahnutie špecifického cieľa, ktorý reprezentuje malú časť funkcionality, analýzu, dizajn, odstraňovanie chýb atď.
- Redmine – Softvérový nástroj pre riadenie manažmentu tímu z hľadiska komunikácie, reportovania a vytvárania úloh atď.
- Šprint – je základnou jednotkou vývoja scrume. Zvyčajne týždňový časový horizont určený na vypracovanie úloh.
- Scrum master – Osoba zodpovedná za scrum, zabezpečuje, že je scrum správne používaný a snaží sa maximalizovať jeho výhody.
- Člen tímu – jednotlivec pracujúci v tíme na softvérovom produkte.
- Watcher – reprezentuje jednotlivca, ktorému je v Redmine posielaný mailom stav úlohy.

9.3 Reportovanie postupu prác na projekte

9.3.1 Reportovanie dokončenej časti úlohy

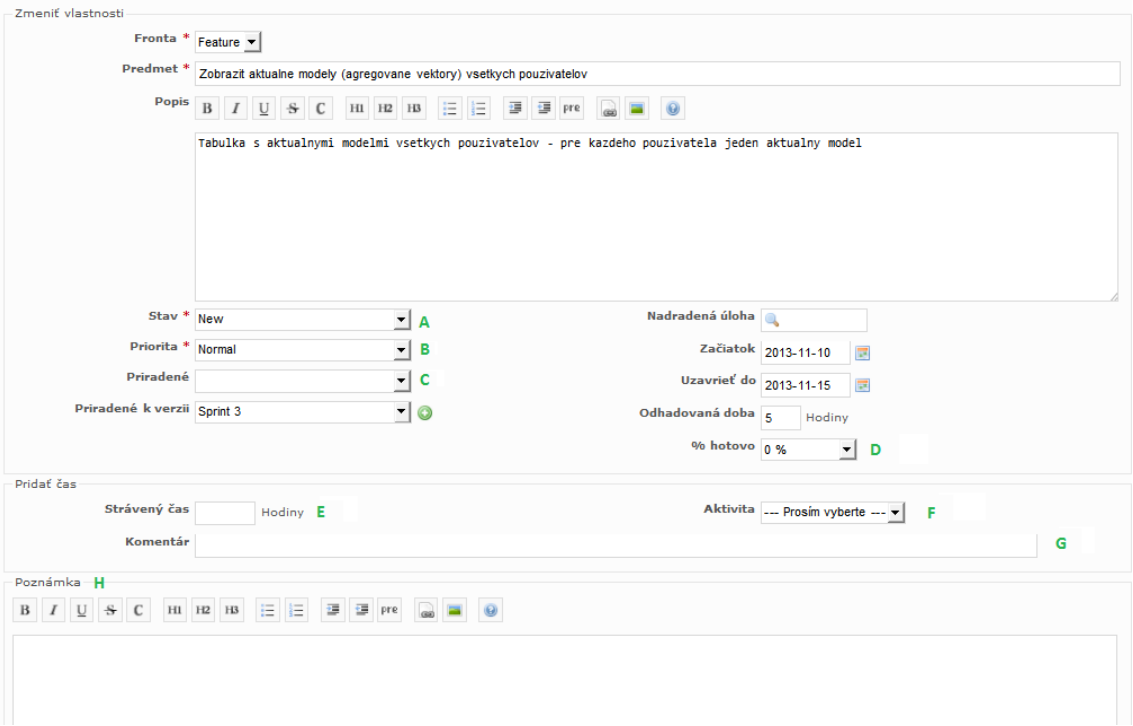
Po dokončení práce na úlohe je člen tímu povinný zapísať do systému Redmine v daný deň stav, v ktorom sa jeho úloha nachádza.

V Redmine člen tímu edituje úlohu nasledovne:

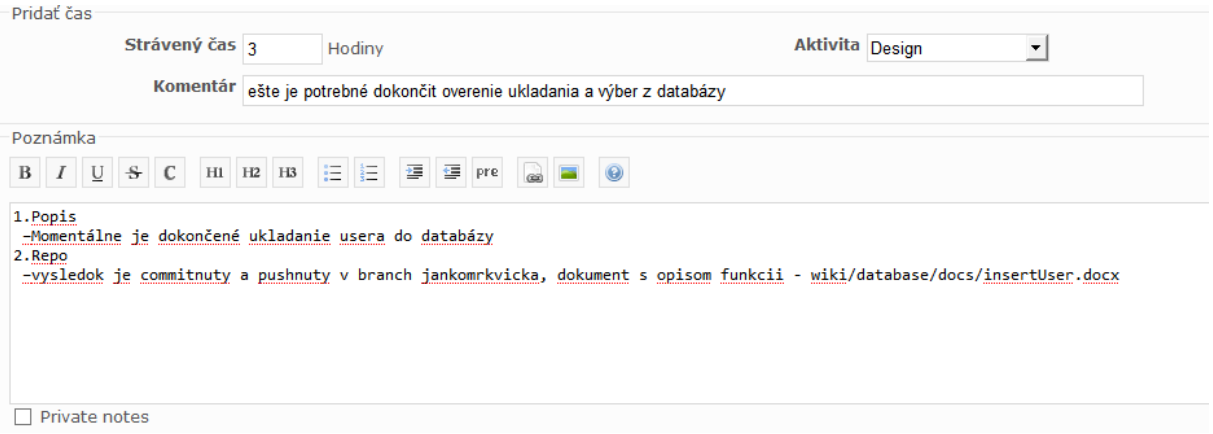
1 Vyberie úlohu z ponuky úloh, na ktorej pracuje
2 Klikne na možnosť editácie úlohy
3 Riadok A(stav)(viď obr. 1) – člen tímu vloží stav z ponuky podľa momentálneho stavu. Keďže úloha ešte nie je dokončená, vyberie možnosť: „in progress“
4 Riadok B(priorita) - člen tímu neupravuje, pretože priorita bola nastavená na začiatku šprintu.
5 Riadok C(priradené) - člen tímu taktiež neupravuje, úlohy sú už pridelené.
6 Riadok D(% hotovo) - člen tímu upraví nasledovne: z ponuky vyberie možnosť, ktorá percentuálne popisuje momentálny stav úlohy.
7 Riadok E(strávený čas) – člen tímu zadá číslo, ktoré reprezentuje čas, ktorý venoval úlohe počas tohto sedenia.
8 Riadok F(aktivita) – člen tímu vyberie z možností tú aktivitu, ktorá najviac korešponduje s jeho doterajšou prácou.
9 Riadok G(komentár) – do komentára člen tímu napíše krátky slovný popis(max 10 slov), čo ešte treba dokončiť.
10 Riadok H(poznámka) – člen tímu vloží dlhší opis zatiaľ dokončenej práce vo formáte : (viď obr. 2)

- Popis – Stručne, ale výstižne, čo je hotové(max 20 slov) (samostatný riadok)

- Ak je parciálny výsledok odovzdaný(commit ak šlo o kóderskú úlohu, súbor, ak išlo o analytickú úlohu atď.) je potrebné napísať, kde a ako je možné si tento výsledok pozrieť. Ak parciálny výsledok odovzdaný nie je, tento odsek ostane prázdny.(samostatný riadok)



Obr.9.1



Obr. 9.2

9.3.2 Reportovanie dokončenej celej úlohy

Po dokončení úlohy je nutné, aby člen tímu zadokumentoval ako úloha dopadla. Postup tvorby reportu je takmer totožný s reportovaním dokončenej časti úlohy(viď 2.2.1.), preto opíšeme iba odlišné kroky.

1 Riadok A(stav) – člen tímu zmení stav na „resolved“
1 Riadok D(% hotovo) - člen tímu zmení na: 100%

1. Riadok E(strávený čas) – člen tímu zadá číslo, ktoré reprezentuje čas, ktorý venoval úlohe počas posledného sedenia na ktorom úlohu dokončil.
2. Riadok F(aktivita) – člen tímu vyberie z možností tú aktivitu, ktorá najviac korešponduje s jeho hotovou prácou.
3. Riadok G(komentár) – člen tímu pole zmaže a napíše „hotovo“
4. Riadok H(poznámka) – člen tímu vloží dlhší opis dokončenej práce vo formáte : (viď obr. 2)

4.1. Popis – Stručne, ale výstižne, čo je hotové(max 20 slov)(samostatný riadok)

4.2. Repo – člen tímu napíše, ako sa ostatní členovia môžu dostať k jeho výsledku. (samostatný riadok)

5. Ak Redmine podporuje možnosť pridať watchera, člen tímu pridá scrum mastra k watcherom. Ak táto možnosť nie je, člen tímu je povinný poslať mail scrum mastrovi vo formáte:

5.1. Príjemca – mailová adresa scrum mastra

5.2. Predmet – [tp08][názov úlohy][stav]

5.3. Správa – skopírovaná z riadku H(poznámka)

9.3.3 Reportovanie problémov pri úlohe

Postup opisu problému pri vypracovávaní úlohy pozostáva z podobných krokov ako reportovanie dokončenej časti úlohy(viď 2.2.1.), preto opíšeme iba rozdielne kroky.

1 Riadok D(% hotovo) - člen tímu upraví nasledovne: z ponuky vyberie možnosť, ktorá percentuálne popisuje momentálny stav úlohy.

2 Riadok E(strávený čas) – člen tímu zadá číslo, ktoré reprezentuje čas, ktorý venoval úlohe počas toho sedenia v ktorom vznikol problém.

3 Riadok F(aktivita) – člen tímu vyberie z možností tú aktivitu, ktorá najviac korešponduje s jeho prácou.

4 Riadok G(komentár) – do komentára člen tímu napíše: „problem“.

5 Riadok H(poznámka) – člen tímu vloží dlhší opis problému v pevne definovanom formáte na samostatné riadky: (viď obr. 3)

1 Popis problému–Stručný popis problému(max 20 slov) pozostávajúci z:
- kde problém vznikol(samostatný riadok)
- akého je problém charakteru(databáza, program, server, analýza) (samostatný riadok)

1.1. Dôvod – prečo problém vznikol(samostatný riadok)

1.2. Dopad pozostávajúci z:
- opis následkov neopravenia chyby(max 15 slov) (samostatný riadok)
- ak na úlohu nadväzujú iné úlohy – člen tímu napíše úlohy, ktoré nutne potrebujú výstup jeho úlohy(samostatný riadok) vo formáte: „úlohy:“ a nasledujú názvy nadväzujúcich úloh oddelené „/“. V prípade ak úloha nie je prepojená s inými úlohami, tento riadok ostane prázdny.

2. Člen tímu oboznámi s problémom ostatných členov tímu nasledovne:

2.1. Ak ide o samostatnú úlohu: Člen tímu pridá k watcherom v Redmine scrum mastera. Ak túto možnosť Redmine nepodporuje, člen tímu pošle scrum masterovi mail vo formáte:

- Príjemca – mailová adresa scrum mastra
- Predmet – [tp08][názov úlohy][problem]
- Správa – skopírovaná z riadku H(poznámka)

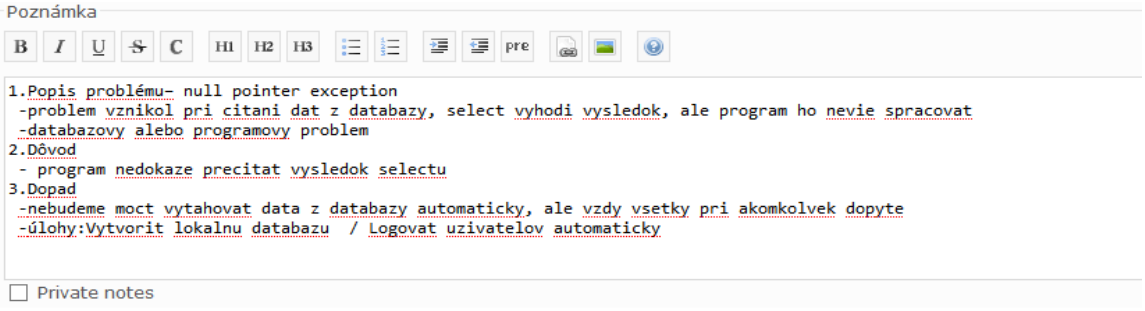
2.2. Ak ide o prepojenú úlohu(úloha nadväzuje, alebo naopak), Člen tímu pridá k watcherom v Redmine scrum mastera a členov tímu, ktorých úlohy sa dotýkajú problému. Ak túto možnosť Redmine nepodporuje, člen tímu pošle jednotlivcom mail vo formáte rovnakom ako v kroku 11.1, avšak k príjemcom zadá maily jednotlivcov, ktorých sa problém týka.

3. Člen tímu zapíše problém na fórum nasledovne:

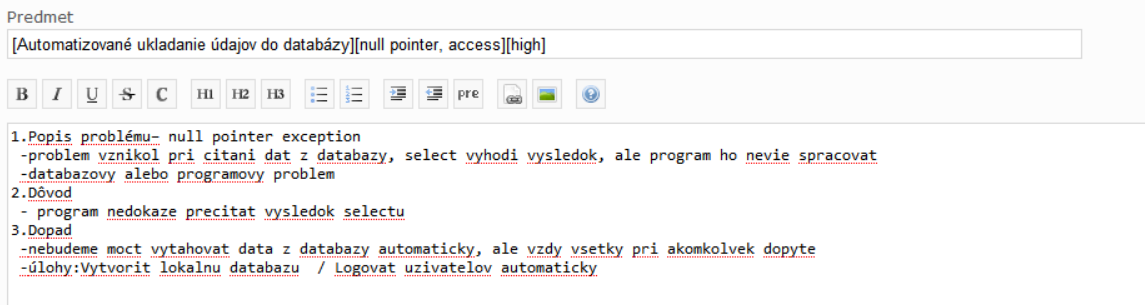
3.1. Vytvorí na fóre novú tému v položke „Problémy s úlohami“

3.2. Tému pomenuje: [úloha][problem][priorita], kde úloha reprezentuje názov úlohy v ktorej vznikol problém, problem reprezentuje kľúčové slová pre problém a priorita reprezentuje súrnosť riešenia problému a môže pozostávať z normal, high, alebo urgent.

3.3. Do správy skopíruje opis problému, ktorý vznikol v bode 10(viď obr. 4)



Obr. 9.3



Obr. 9.4

9.3.4 Reportovanie práce tímu po ukončení šprintu

Po ukončení šprintu scrum master vytvorí:

1 dokument .pdf, v ktorom bude nasledovná tabuľka:

Šprint č.:xx						
Úloha č.:	Priorita	Názov úlohy	Pridelená	Začiatok	Stav	Poznámka
1.	Normal / High / Urgent	Názov úlohy	Meno člena tímu	21.8.1998	Hotová / Nedokončená / Prenesená do ďalšieho šprintu	Ľubovoľná poznámka k úlohe
...	...	...	...	...	...	...

1. Formát uloženého súboru bude ulohy_sprint_xx.pdf, kde xx reprezentuje dvojciferné čísla od 00 až po 99.
2. Tento dokument rozpošle všetkým členom tímu a formát mailu bude:

2.1. Príjemca – mailové adresy všetkých členov tímu
2.2. Predmet – [tp08][ulohy_sprint_xx]
2.3. Správa – „V prílohe posielam tabuľkový prehľad úloh minulého šprintu.“
2.4. Príloha – súbor ulohy_sprint_xx.pdf

10 Metodika manažmentu testovania pomocou nástroja JUnit Test

10.1 Úvod

Účelom tejto metodiky je určenie správnej tvorby testov pre potreby nášho tímového projektu. Metodika je určená pre všetkých programátorov, ktorí vytvárajú novú funkcionalitu alebo upravujú starú. Primárne je určená pre sekciu **Testovania** a **Kvality**.

Predmetom metodiky dolnej úrovne je spôsob vytvorenia testu JUnit vo vývojovom prostredí Eclipse.

10.2 Súvisiace manuály

Táto metodika sa odkazuje na nasledovné manuály:

- manuál k nástroju JUnit verzia 4

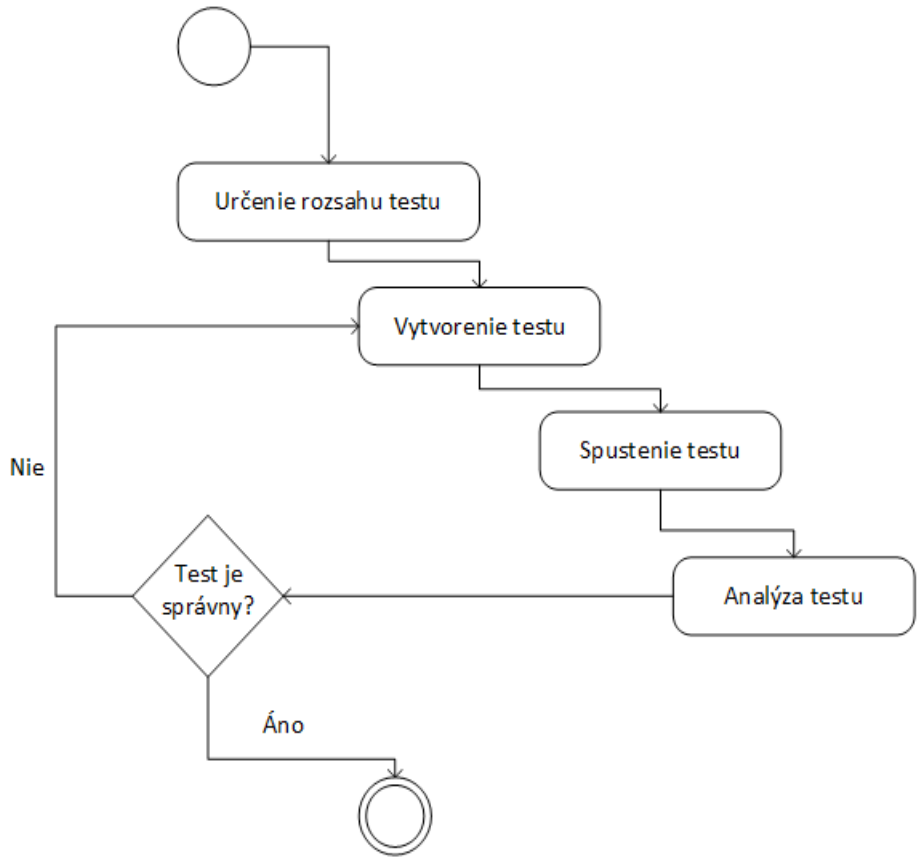
10.3 Použité pojmy

- jednotkový test – nízkoúrovňový test, ktorý testuje kód, zväčša triedy a metódy
- JUnit – nástroj na testovanie zdrojových kódov v jazyku Java
- framework – softvérová štruktúra, ktorá slúži na podporu pri programovaní a organizovaní softvérových projektov.
- Eclipse–vyvojové prostredie pre jazyk Java
- UML - grafický jazyk na vizualizáciu, špecifikáciu, navrhovanie a dokumentáciu programových systémov.

10.4 Vytvorenie jednotkového testu

Táto časť metodiky pokrýva proces vytvorenia jednotkového testu v nástroji JUnit Test vo vývojovom prostredí Eclipse.

Vytvorenie testu podľa krokov opísaných nižšie možno vyjadriť aj nasledovným UML diagramom:



Obr 10.1 *:* Postupnosť krokov vytvorenia testu

10.4.1 Rozsah testu

Test by mal pokrývať jednotlivé časti funkcionality systému. Po pridaní novej funkcionality do systému, je programátor, ktorý novú funkcionalitu vytvoril povinný vytvoriť JUnit test, na následnú kontrolu tejto funkcionality.

10.4.2 Vytvorenie testu:

Vytvorenie testu začína vytvorením súboru *JUnit Test Case*verzie 4 vo vývojovom prostredí Eclipse. Názov súboru začína slovom *Test*. Z názvu súboru by malo byť programátorovi hneď jasné, akú funkcionalitu test testuje.

Umiestnenie súboru s testom:

Vytvorený test sa umiestni do zdrojového priečinku *test* (nie *src*), kde sa následne umiestni do rovnakého balíka ako trieda, ktorú vytvorený test testuje. Ak sa v jednom teste testujú viaceré triedy z rôznych balíkov, súbor s testom sa uloží do najnižšieho balíka v hierarchii, v ktorom sa všetky testované triedy nachádzajú.

Vytvorenie testu:

Test musí obsahovať aspoň jednu funkciu, nad ktorou je anotácia *@Test*. V tele takejto funkcie je napísaný test, ktorý testuje zvolenú funkcionalitu. Ak sa testuje správny výstup pri zadaných vstupných parametroch, tak na konci tela testovacej funkcie je použitá jedna z funkcií *junit.Assert*, na základe ktorej sa test vyhodnotí ako úspešný alebo neúspešný.

```
Import static org.junit.Assert.*;
import org.junit.Test;
public class TestAverage{
    @Test
public voidtestAverageFunc(){
int a  =10;
int b =8;

assertEquals("Average of 10 and 8 mustbe 9",9,average(10,8));
```

Ak sa testuje vyhodenie výnimky, tak test musí obsahovať rozšírenie anotácie *@Test* o triedu požadovanej výnimky. Test je úspešný, len ak nastala požadovaná výnimka. Telo testovacej metódy neobsahuje *Assert*.

```
@Test(expected=IllegalArgumentException.class)
```

Ak sa testuje rýchlosť systému, tak test musí obsahovať rozšírenie anotácie *@Test* o počet milisekúnd, za ktorý musí test zbehnúť, inak bude vyhodnotený ako neúspešný.

```
@Test(timeout=1000)
```

10.4.3 Spôsob testovania

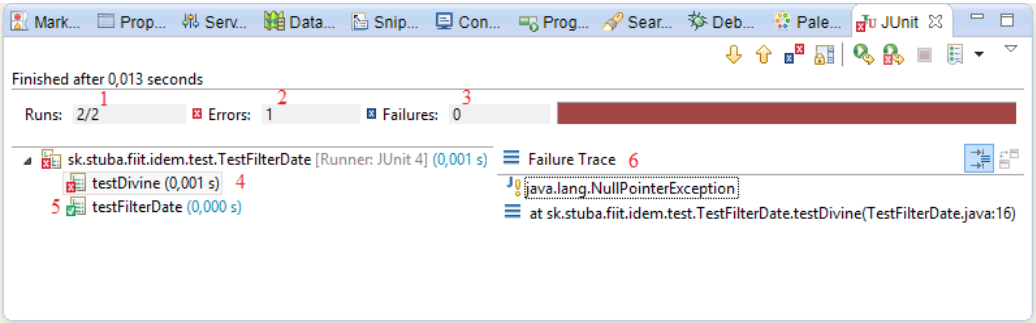
Vytvorený test by mal obsahovať testy, ktoré testujú celú pridanú funkcionalitu, s výnimkou jednoduchého kódu. Každý test musí na vstupe dostať aspoň päť rôznych vstupov. Odporúča sa použiť *Parameterized.class*(bližšie informácie o použití triedy sa nachádzajú v manuáli JUnit 4). Následne je nutné otestovať metódu aj pri nesprávnych vstupných parametroch, aby sa otestovali možné chyby a vyhodenie správnych výnimiek. Test sa považuje za správne vytvorený a metóda za správne otestovanú, ak sú dodržané zadané podmienky.

10.4.4 Spustenie testu

Vytvorené testy je možné spúšťať samostatne alebo naraz. Samostatne sa spúšťajú kliknutím na daný test a následne *Run/Debug As* ➢ *JUnit Test*. Pre pustenie viacerých testov naraz (sady testov) je potrebné vytvoriť *JUnit test suite*, kde sa pridajú všetky jednotlivé testy, ktoré sa majú spustiť. *JUnit test suite* sa odporúča vytvoriť cez prostredie Eclipse: *New* ➢ *JUnit Test Suite* a v zobrazenom okne vybrať z uvedených dostupných testov.

10.4.5 Výsledky testu

Po spustený jednotlivých testov sa v Eclipse na paneli JUnit zobrazia výsledky testov. Podrobný popis panelu je na obrázku nižšie.



Obr. 10.2 : Zobrazenie výsledkov testu

Vysvetlivky:

1. Označuje počet vykonaných testov.
2. Označuje počet vzniknutých chýb pri vykonávaní testov.
3. Označuje počet zlyhaní (pri použití funkcie *fail()*).
4. Označuje dĺžku trvania testu.
5. Označuje výsledný stav testu.
6. Vypisuje zachytené chyby/výnimky.

10.5 Záver

Navrhnutá metodika dojnej úrovne prebrala vytváranie jednotkových testov pre potreby nášho tímového projektu. Je dostatočne konkrétna aby viedla pri členov tímu správnou cestou pri správnom vytvorení testov. Pre bližšie informácie o nástroji JUnit je potrebné preštudovať manuál, na ktorý táto metodika odkazuje.

11 Metodika pre priebeh tímového stretnutia a pridanie novej úlohy do Redmine

11.1 Úvod

Tento dokument špecifikuje metodiku pre potreby tímového projektu tímu číslo 8. Táto metodika slúži na správny priebeh tímového stretnutia v rámci jedného týždňa. Výsledkom tohto stretnutia je pridanie nových úloh v Redmine pre splnenie do nasledovného tímového stretnutia.

11.2 Dedikácia metodiky

Touto metodikou sa riadia všetci členovia tímu. Je nevyhnutné sa ňou riadiť počas tímového stretnutia vo štvrtok medzi 12:00 – 15:00 v Jobsovom softvérovom štúdiu. Používať pri pridávaní nových úloh do Redmine.

11.3 Nadväzujúce metodiky a dokumenty

Nadväzujúcou metodikou je Reportovanie postupu práce v Redmine.

11.4 Skratky a pojmy

Redmine – softvérový nástroj pre manažment v tíme
Scrum – agilná metodika vývoja softvérového produktu
Sprint backlog - naplánované príbehy s prioritizáciou pre daný sprint
Sprint – vývoj softvéru metodikou Scrum v dvojtýždňových intervaloch
Tímové stretnutie – skratka „TS“
Jobsové softvérové štúdio – skratka „JSŠ“

11.5 Roly a zodpovednosti účastníkov

Rola:

- zapisovateľ – každé TS iný člen tímu, kde striedanie je v abecednom poradí

Zodpovednosť:

- vyhotoviť zápis z TS
- pridať nové úlohy do Redmine

Rola:

- vedúci stretnutia (Scrum master) – každé TS iný člen tímu, kde striedanie je v abecednom poradí

Zodpovednosť:

- viesť TS
- oboznámiť s progresom na projekte zákazníka (Scrum owner)

Rola:

- člen tímu – všetci členovia tímu

Zodpovednosť:

- ohodnotenie jednotlivých úloh
- priradenie úlohy

11.6 Definované procesy

Nasledovné procesy je nutné dodržať v stanovenom poradí:

1. Proces - vyhodnotiť aktuálny stav projektu
2. Proces - vybrať naplánované príbehy zo Sprint backlog
3. Proces - ohodnotiť úlohy vyplývajúce z naplánovaných príbehov v Sprint backlog
4. Proces - vytvoriť nové úlohy v Redmine

11.7 Opis definovaných procesov

V tejto časti je podrobný opis jednotlivých definovaných procesov v kapitole 6. Proces je definovaný postupnosťou krokov, kde pred príslušnými krokmi je zodpovedná osoba za danú časť procesu.

11.7.1 Proces - vyhodnotiť aktuálny stav projektu

Člen tímu:

1. Prítomnosť na TS vo štvrtok o 12:00 v JSŠ.

Zapisovateľ:

2. Vytvoriť zápis z TS od jeho začiatku až do jeho konca

Vedúci stretnutia:

3. Oboznámiť zákazníka o aktuálnom stave produktu
4. Postupne prečítať názvy všetkých úloh z posledného TS v Redmine
5. Postupne vyzvať členov tímu k vyjadreniu sa o splnení/nesplnení prečítaných úloh

Člen tímu:

6. Oboznámiť ostatných členov tímu o stave úlohy, ktorú prečítal vedúci stretnutia

11.7.2 Proces - vybrať naplánované príbehy zo Sprint backlog

Vedúci stretnutia:

1. Vybrať naplánované príbehy zo Sprint backlog s najvyššou prioritou
2. Vybrané naplánované príbehy rozbiť na jednotlivé úlohy
3. Počet úloh je minimálne 1 pre každého člena tímu z vybratých naplánovaných príbehov

Členovia tímu:

4. Stanoviť prioritu, dobu riešenia a dobu splnenia jednotlivých úloh na základe vzájomnej dohody

11.7.3 Proces - ohodnotiť úlohy vyplývajúce z naplánovaných príbehov v Sprint backlog

Členovia tímu:

1. Postupne ohodnotiť každú úlohu pomocou scrum kariet
2. Každý člen tímu si podľa svojho uváženia sám ohodnotí danú úlohu
3. Následne ukázať pred všetkými členmi tímu kartu pre danú úlohu
4. Vyjadrenie členov tímu ohľadom ich zvolenej náročnosti s maximálnou alebo minimálnou hodnotou karty k danej úlohe
5. Vyjadrenie ostatných členov tímu k danej úlohe
6. Na základe spoločnej dohody a stanoviť odhadovaný čas pre riešenie úlohy

11.7.4 Proces - vytvoriť nové úlohy v Redmine

Zapisovateľ:

1. Po skončení TS pridať nové úlohy do Redmine
2. Dodržať predpísanú formu pre pridanie nových úloh do Redmine (Obrázok 6.5.) podľa kapitoly 7.4.1
3. Oboznámiť vedúceho stretnutia prostredníctvom emailu tp-tim-8@googlegroups.com o pridaní všetkých nových úloh do Redmine
4. Vykonať najneskôr vo štvrtok do 20:00

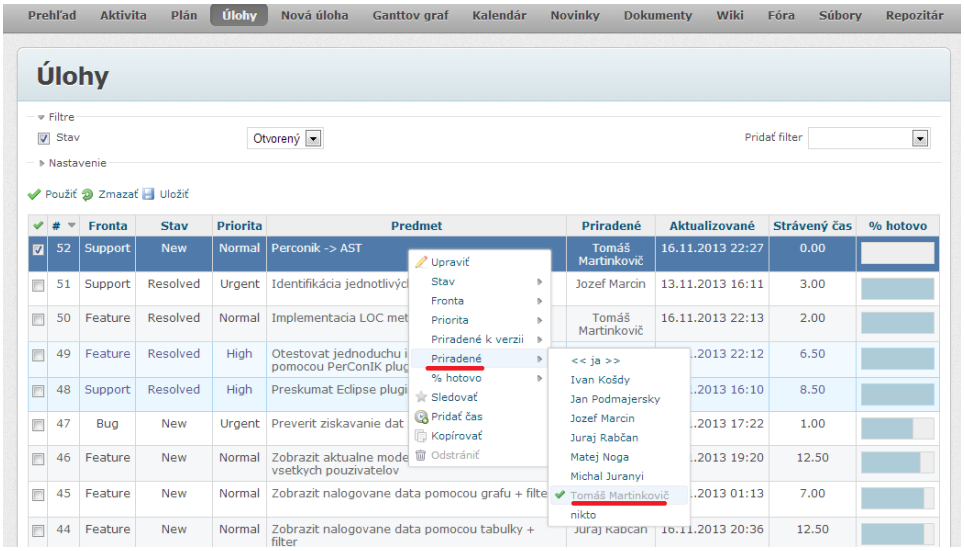
Vedúci stretnutia:

5. Po prijatí emailu od zapisovateľa skontrolovať správnosť pridaných nových úloh
6. Nesprávne zadefinovanú pridanú novú úlohu opraviť do požadovaného stavu ako bola dohodnutá na TS
7. Oboznámiť všetkých členov tímu prostredníctvom emailu tp-tim-8@googlegroups.com o priraďovaní nových úloh v Redmine
8. Vykonať najneskôr vo štvrtok do 23:59

Členovia tímu:

9. Po prijatí oznámenia o začatí priraďovania úloh v Redmine, si konkrétny člen tímu musí vybrať minimálne 1 ľubovoľnú úlohu a to nasledovne (Obrázok 11.):
    - V záložke „Úlohy“ kliknúť pravým tlačidlom na danú úlohu

- V položke „Priradené“ nastaviť svoje meno



Obrázok 11. Nastavenie zodpovedného člena tímu k danej úlohe.

Položky vo formulári „Nová úloha“ vyplniť nasledovne:

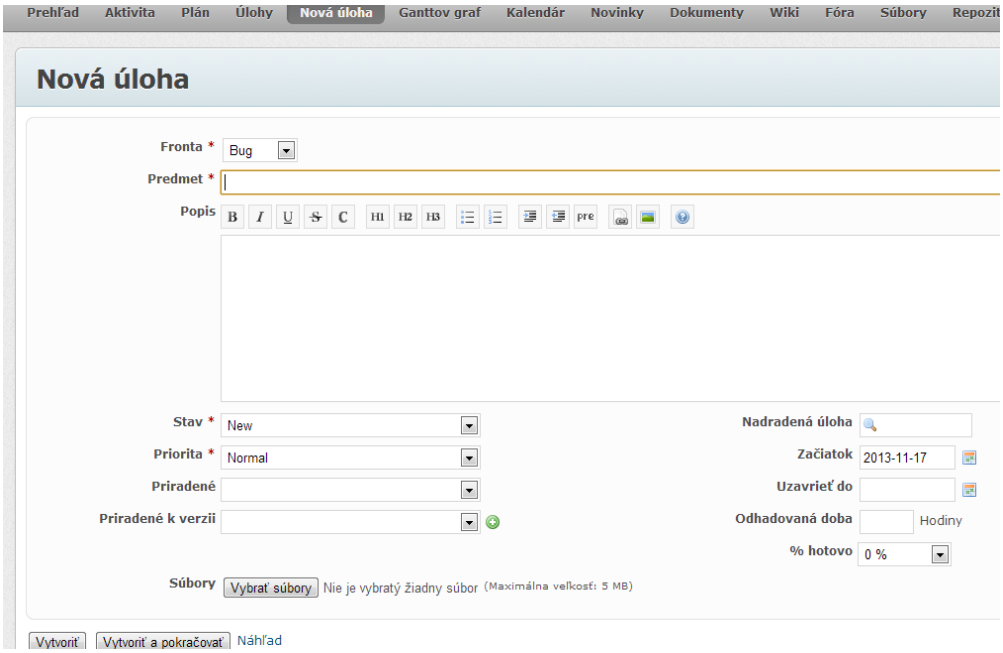
1. **Fronta** – vybrať na základe typu úlohy jednu z možností:
   - Bug – nájdená chyba v existujúcej úlohe
   - Support – podporná úloha pre implementáciu riešenia
   - Feature – implementačná úloha
2. **Predmet** – výstižne pomenovať novú úlohu
3. **Popis** – podrobne opísať zadanie úlohy
4. **Stav** – nastaviť na položku New
5. **Priorita** – nastaviť na základe dohodnutých priorít úloh, nadväznosti úloh a termínov ich splnenia na TS a to nasledovne:
   - Immediate:
     - úloha s maximálnou prioritou
     - bez vyriešenia nie je možná práca na žiadnej úlohe
     - stanovená doba na TS na vyriešenie je max. 1 deň
   - Urgent:
     - úloha s maximálnou prioritou
     - bez vyriešenia nie je možná práca na väčšine úloh
     - stanovená doba na TS na vyriešenie sú max.2 dni
   - High:
     - úloha s vysokou prioritou

- bez vyriešenia nie je možná práca na minimálne 1 úlohe a maximálne 2 úlohách
- stanovená doba na TS na vyriešenie sú max.3 dni

o Normal:
- úloha s normálnou prioritou
- žiadna iná úloha nenadväzuje na danú úlohu
- stanovená doba na TS na vyriešenie je do nasledujúceho TS

o Low:
- úloha s minimálnou prioritou
- žiadna iná úloha nenadväzuje na danú úlohu
- stanovená doba na TS na vyriešenie je do nasledujúceho TS

6. **Priradené** – nenastavovať na žiadneho konkrétneho člena tímu
7. **Nadradená úloha –** napísať predmet z existujúcej úlohy nasledovne:
   o ak je nová úloha podúlohou existujúcej úlohy
   o ak je nová úloha nastavená vo fronte typu Bug
   o inak nenastavovať
8. **Priradené k verzii** – vybrať na základe aktuálneho bežiaceho šprintu:
   o ak začína nový šprint:
   - Tlačidlo „+“ otvoriť formulár *Nová verzia*
   - *Názov* vyplniť v tvare „Sprint x“, kde „x“ predstavuje poradové celé číslo šprintu – napríklad „Sprint 5“
   - ostatné položky nechať v štandardnom stave
9. **Začiatok –** nastaviť dátum podľa dňa, kedy sú nové úlohy pridávané do Redmine
10. **Uzavrieť do –** nastaviť ako súčet položky „Začiatok“ a „Priorita“
    o napríklad: „Začiatok je 14.11.2013“ + „Priorita je Urgent (max. 2 dni)“ = 16.11.2013
11. **Odhadovaná doba –** napísať číslom počet odhadovaných hodín pre prácu na novej úlohe
12. **% hotovo** – vždy nastaviť na 0%
13. **Súbory** – vybrať len súbory, ktoré bezprostredne súvisia s novou úlohou
14. Na vytvorenie novej úlohy je potrebné potvrdiť vyplnený formulár tlačidlom **Vytvoriť.**



Obrázok 11.2 Formulár „Nová úloha“.

12 Metodika manažmentu verzií

12.1 Zameranie

Metodika je určená všetkým členom tímu, ktorí pracujú so zdrojovými kódmi vytváraného projektu.

12.2 Používaný nástroj

Pre správu verzií je používaný systém Git. Na klientských počítačoch je používaný oficiálny Git klient a v Eclipse IDE plug-in EGit.

12.3Inicializácia

12.3.1 Autentifikácia

Komunikácia a autentifikácia prebieha prostredníctvom SSH, pomocou kľúčov. Pred začatím práce je preto nutné, aby si programátor vygeneroval privátny a verejný kľúč, pokiaľ zatiaľ žiadny nemá a odoslal verejný kľúč poverenému členovi tímu. Správca servera nahrá verejný kľúč používateľa na server, čím umožní programátorovi plný prístup k repozitárom.

SSH kľúč sa generuje pomocou nástroja Git Bash, ktorý je súčasťou oficiálneho balíčka Git pre OS Windows, nasledovne:

1. Spustiť program Git Bash
2. ssh-keygen -t rsa
3. Program sa spýta na cestu ku kľúču. Túto položku nevypĺňať, nechať východziu hodnotu. V odôvodnených prípadoch možno zmenu konzultovať s kompetentným členom tímu.
4. Program sa v dvoch krokoch spýta na heslo a overenie hesla. Vyplnenie hesla nie je nevyhnutné, ale odporúča sa.
5. Po úspešnom vygenerovaní kľúča program vypíše jeho odtlačok (fingerprint), čím je tento proces ukončený. Program vytvoril 2 súbory s kľúčmi umiestnené podľa uvedenej cesty – id_rsa (súkromný kľúč) a id_rsa.pub (verejný kľúč)
6. Súbor s verejným kľúčom alebo jeho obsah odoslať poverenému členovi tímu.

12.3.2 Získanie zdrojových súborov projektu

Eclipse EGit:

Otvoriť pohľad Git Repositories
Kliknúť na „Clone a Git repository“
URI:
git@team08-13.ucebne.fiit.stuba.sk:repos/<*nazov_repozitara*>.git
Next
Ak je vyžiadané, zadať heslo k vygenerovanému kľúču
Označiť požadované vetvy (odporúčajú sa všetky)
Next
Directory: miesto uloženia súborov (napr. Eclipse workspace)
Initial branch: develop
Zaškrtnúť „Import all existing projects after clone finishes“

12.4 Používané vetvy

12.4.1 Master

Hlavná produkčná vetva, obsahuje najnovšiu stabilnú verziu projektu. Táto vetva je aktualizovaná z vetvy *develop* po každom šprinte, ktorého výsledkom je funkčná verzia projektu.

12.4.2 Develop

Hlavná vývojová vetva, obsahuje kód vytváraný v priebehu šprintu. Do tejto vetvy sa počas šprintu pridávajú ucelené implementácie jednotlivých požiadaviek.

12.4.3 Feature / bugfix

Vývojové vetvy vytvárané jednotlivými programátormi za účelom práce na im pridelenej úlohe. Po dokončení je každá z týchto vetiev pripojená k *develop* vetve a zmazaná.

12.5 Práca s verziami

12.5.1 Tvorba a úprava kódu

Postup pri tvorbe alebo zmenách kódu v zmysle zapracovania novej funkcionality, úpravy existujúcej funkcionality alebo opravy chyby.

1. Stiahnutie aktuálnej verzie projektu z repozitára
   Eclipse EGit:
   pravý klik na projekt->Team->Pull
   Git Bash:

git pull

2. Vytvorenie novej vetvy (branch) s názvom v tvare *feature/<nazov_funkcionality>,* v prípade implementácie novej alebo úpravy existujúcej funkcionality, alebo *bugfix/<nazov_bugu>,* ak ide o opravu chyby

   Eclipse EGit:
   pravý klik na projekt->Team->Switch To->New Branch...
   Branch name: vyplniť podľa uvedenej konvencie
   Pull strategy: None
   Checkout new branch: zaškrtnuté
   Finish
   Git Bash:
   git branch <nazov_vetvy> && git checkout <nazov_vetvy>

3. Práca prebieha nad vytvorenou vetvou, čím sa obmedzí množstvo konfliktov

4. Súčasťou jednotlivých commitov je správa opisujúca obsah daného commitu v tvare:

[fix] kratky popis commitu – pre commity poskytujúce opravu chyby
[feature] kratky popis commitu – pre commity poskytujúce novú funkcionalitu

Eclipse EGit:
pravý klik na projekt->Team->Commit...
Commit message: vyplniť podľa uvedenej konvencie
Commit
Git Bash:
git commit -a -m "<sprava podla uvedenej konvencie>"

5. V prípade dlhšej (viacdennej) práce na vetve, nahrať (push) najmenej raz denne lokálne zmeny do spoločného repozitára

Eclipse EGit:
pravý klik na projekt->Team->Remote->Push...
Zvoliť Configured remote repository
Next
Source ref: zvoliť aktuálnu vetvu na ktorej pracujem
Destination ref: zvoliť rovnako ako Source ref (pokiaľ sa nezvolilo automaticky)
Add All Branches Spec
Finish
Git Bash:

git push

6. Po dokončení práce a otestovaní vytvoreného riešenia nastane fáza spojenia s hlavnou vývojovou vetvou nasledovne (najprv stiahnúť aktuálnu verziu projektu z repozitára – viď bod 1.):
   a) Ak od momentu vytvorenia vlastnej pracovnej vetvy *nastali* v hlavnej vetve zmeny:
      i. pripojenie (rebase) hlavnej vetvy k vlastnej pracovnej vetve

Eclipse EGit:

pravý klik na projekt ->Team-> Rebase...
Zvoliť Local->develop

Git Bash:

git rebase develop

      ii. otestovanie funkčnosti
      iii. spojenie (rebase) pracovnej a hlavnej vetvy

Eclipse EGit:

pravý klik na projekt->Team->Switch To->develop
pravý klik na projekt->Team->Rebase...
Zvoliť Local-><nazov_pracovnej_vetvy>

Git Bash:

git checkout develop
git rebase <nazov_pracovnej_vetvy>

   b) Ak od momentu vytvorenia vlastnej pracovnej vetvy *nenastali* v hlavnej vetve zmeny:
      i. spojenie (rebase) pracovnej a hlavnej vetvy – viď bod a.iii
7. Ak zaniká potreba vytvorenej pracovnej vetvy (požadovaná vlastnosť je implementovaná podľa očakávaní, resp. zaznamenaný problém je opravený v zistenom rozsahu), je táto vetva zmazaná

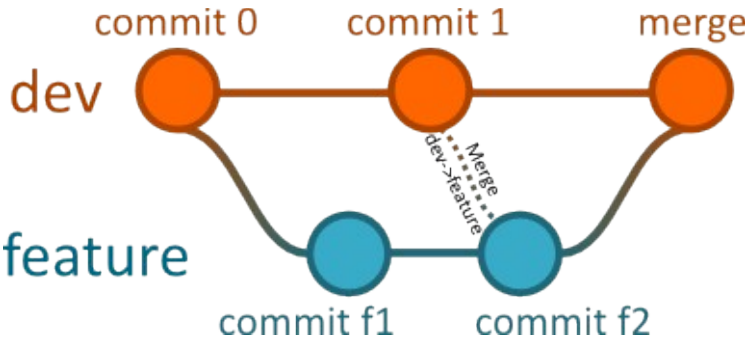
Eclipse EGit:

pravý klik na projekt->Team->Advanced->Delete Branch...
Zvoliť Local-><nepotrebna_pracovna_vetva>
OK

Git Bash:

git checkout develop (pokiaľ je aktívna pracovná vetva)
git branch -d <nepotrebna_pracovna_vetva>

8. Nahrať zmeny do spoločného repozitára (push) – viď bod 5.



Obr. 12.1

12.5.2 Vrátenie vykonaných zmien

Postup pre vrátenie zmien vykonaných v zdrojových súboroch. V prípade zmien vo vlastnej vývojovej vetve alebo v prípade, že zmeny zatiaľ neboli nahrané do spoločného repozitára môže takýto návrat vykonať ktorýkoľvek člen tímu. Pokiaľ ide o vrátenie zmien v hlavnej vývojovej vetve a ďalších spoločných vetvách, musí byť takýto návrat opodstatnený a po dohode v tíme ho vykoná poverený člen.

a) Pokiaľ nebol vykonaný push:

Eclipse EGit:

pravý klik na projekt->Team->Reset...
Zvoliť References->HEAD (ak nebol vykonaný commit)
Zvoliť References->ORIG_HEAD (ak bol vykonaný commit – odstráni všetky zmeny od posledného vykonania pull)
Reset type: Hard
Reset

Git Bash:

git reset --hard HEAD (ak nebol vykonaný commit)
git reset --hard ORIG_HEAD (ak bol vykonaný commit – odstráni všetky zmeny od posledného vykonania pull)

b) Pokiaľ bol vykonaný push:

Eclipse EGit:

otvoriť pohľad Git Repositories
pravý klik na repozitár->Show In->History
pravý klik na požadovaný commit->Reset->Hard

Git Bash:

git reset --hard <commit-id>

12.5.3 Vydanie novej verzie

Postup pri uvoľňovaní nových produkčných verzií projektu.

1. Kontrola funkčnosti a konzistencie požadovaného stavu v hlavnej vývojovej vetve
2. Spojenie (merge) hlavnej vývojovej a produkčnej vetvy v požadovanom mieste
   a) Ak má byť nová verzia vytvorená z aktuálnej verzie hlavnej vývojovej vetvy:

Git Bash:

```
git checkout master
git merge develop
```

   b) Ak má byť nová verzia vytvorená z inej ako aktuálnej verzie hlavnej vývojovej vetvy:

Git Bash:

```
git merge <commit_id>
```

3. Označenie (tag) uvoľnenej verzie v hlavnej produkčnej vetve

Git Bash:

```
git tag <cislo_verzie>
```

4. Nahrať zmeny do spoločného repozitára (push) – viď bod 5.1.5

13 Manažment monitorovania projektu

Manažment monitorovania projektu je veľmi dôležitou súčasťou každého väčšieho softvérového projektu. Hlavným dôvodom pre toto tvrdenie je fakt, že projekty bez kontroly postupov práce, kontroly cieľov a charakteristík v projekte, kontroly termínov odovzdávania a najmä kontroly a prehliadky stavu projektu často krát nedosahujú dostatočne kvalitné výsledky a riešenia. Manažment monitorovania pozostáva najmä z kontroly postupu práce na projekte, ktorý zaručuje aby sa úlohy na projekte dokončovali v čas a najmä kontinuálne.

13.1 Monitorovanie postupu práce na projekte

Pre tento projekt sme ako primárny systém pre manažment úloh ale aj komunikáciu zvolili systém Redmine. Tento systém podporuje sofistikovaný spôsob tvorby úloh, sledovanie aktivity členov, sledovanie postupu práce jednotlivých úloh, celkový pohľad na úlohy v šprintoch, Gantov graf, ktorý opisuje snahu celého tímu v rámci časového obdobia a mnoho iných. Vďaka tomuto nástroju je monitorovanie postupu práce oveľa jednoduchšie a prínosné pre obe strany. Manažéra monitorovania a aj jednotlivca v tíme, pretože dokáže rýchlo zistiť, ako stojí so svojimi úlohami a vďaka dobrému reportovaniu problémov rýchlo opravovať chyby, ktoré vznikli. Hlavnou úlohou manažéra monitorovania pri monitorovaní postupu práce teda je:

- Sledovať korektný spôsob popisu úloh
- Kontrolovať stav úloh a problémy vznikajúce s nedostatkom času na úlohu
- Kontrolovať problémy vznikajúce z chýb pri vypracovávaní úlohy
- Kontrolovať témy vo fóre zaoberajúce sa jednotlivými úlohami
- Sledovať, či sú všetky úlohy pridelené
- Sledovať ukončovanie úloh a termíny úloh

14 Manažment podpory vývoja

Tímová práca na väčších projektoch si vyžaduje použitie širokej palety nástrojov pre podporu tímovej práce. Projekty zamerané na vývoj softvéru si navyše často vyžadujú použitie rôznych nástrojov pre vývoj a mnoho knižníc, ktorých množstvo spravidla rastie s rozsahom projektu a jeho zložitosťou.
V tejto časti sú opísané nástroje a knižnice používané pri spolupráci a vývoji v našom tíme.

14.1 Používané nástroje

14.1.1 Spolupráca v tíme

Pre podporu spolupráce v tíme sme sa rozhodli nasadiť často odporúčanú webovú aplikáciu Redmine, ktorú sme nasadili na náš tímový server. Nasadenie na vlastnom serveri so sebou nesie nutnosť vynaloženia väčšieho úsilia pri nasadzovaní, ale prináša nezávislosť a možnosť používania rozšírení podľa vlastných potrieb. V súvislosti s používaním SCRUMu pre tímovú prácu, používame plugin Agile Dwarf, ktorý prináša podporu delenia úloh do šprintov a vlastnú implementáciu burndown chartu. Pre prehľadné zdieľanie dokumentov používame službu Dropbox. Tá zabezpečuje, že každý člen má dostupné aktuálne verzie dokumentov a samotná služba zabezpečuje aj určitú zálohu a správu verzií zdieľaných súborov. Vďaka dostupnosti desktopového klienta pre rôzne platformy umožňuje pohodlnú prácu pre každého člena bez ohľadu na používanú platformu.

14.1.2 Správa verzií zdrojových kódov

Správu verzií zdrojových kódov zabezpečujeme pomocou systému Git. Tento systém máme nasadený na vlastnom tímovom serveri, čím ho máme plne pod kontrolou. Klientskú časť systému tvoria oficiálne aplikácie pre operačné systémy Linux a Windows a plugin pre vývojové prostredie Eclipse. Git repozitáre nášho projektu je možné tiež prehliadať v aplikácii Redmine.

14.1.3 Vývojové prostredie

Vzhľadom na povahu aplikácie a zloženie tímu používame multiplatformové integrované vývojové prostredie Eclipse, ktoré nám poskytuje komfort a nástroje potrebné pre vývoj webových aplikácií v jazyku Java.

14.1.4 Databázové systémy

Pre systém Redmine používame relačnú databázu PostgreSQL. Samotná vytváraná aplikácia využíva NoSQL databázu MongoDB, ktorá vzhľadom na štruktúru a charakter spracovávaných dát poskytuje lepší výkon.

14.1.5 Monitorovanie používateľov

Zbieranie údajov o práci používateľov (biometrických - práca s klávesnicou a myšou – a údajov o práci v niektorých programoch) zabezpečuje Windows aplikácia UACA (User Activity Client Application). Táto aplikácia pomocou svojich modulov zaznamenáva a hromadne odosiela získane údaje o práci používateľov na server s databázou. Z tejto databázy potom získavame údaje na analýzu pre náš projekt.

14.1.6 Používané jazyky, knižnice a frameworky

Celý projekt je vyvíjaný v jazyku Java. Webová aplikácia okrem toho využíva JavaScript pre rozšírenie funkcionality používateľského rozhrania.

14.1.7 Webová aplikácia

Základom webovej aplikácie je MVC framework Spring, uľahčujúci tvorbu webových aplikácií. Pre komunikáciu s databázou používame knižnice pre prácu s MongoDB. Údaje pre analýzu získavame z databázy projektu PerConIK, ktoré získavame pomocou webovej služby, pri čom sa používajú knižnice vytvorené firmou Gratex. Pre vytváranie grafov z analyzovaných používateľských dát je použitá JS knižnica D3.

14.1.8 Eclipse plugin

Klientskou časťou aplikácie zodpovednej za získavanie dát z Eclipse IDE je plugin, ktorý zaznamenáva a čiastočne analyzuje pomocou Gratex knižníc činnosti programátora vrámci IDE. Základom je PerConIK plugin pre Eclipse, ktorý pomocou Eclipse API umožňuje zachytávať udalosti vykonané používateľom. V našom projekte ho preto použijeme ako framework na ktorom si možme vytvoriť vlastnú potrebnú funkcionalitu. Všetky takto získané údaje sa odosielajú pomocou Gratex knižníc na ďalšie spracovanie mimo Eclipse IDE.

15 Manažment tvorby dokokumentácie

Manažment tvorby dokumentácie sa zaoberá tvorbou dokumentácie a tvorbou ľablón. Dokumentácia sa vytvára priebežne. Dokumenty sa v neskorších fázach dávaju dokopy v Latexe. Dokumentácia sa zdieľa cez git aj cez DropBox. Obrázky sa ukladajú vo formáte png do adresára /figures. Manažér dokumentácie má na starosti:

- vytváranie šablón pre dokumentáciu
- integráciu dokumentov do finálnych dokumentov
- úpravu štýlov
- Výber vhodných nástrojov pre tvorbu dokumentácie

16 Manažment komunikácie

V našom tíme sa kladie dôraz na osobnú komunikácia. Stretáme sa na vopred dohodnutých stretnutiach v jobsovom štúdiu, účasť je povinná pre všetkých členov tímu. Na väčšine stretnutí sa navrhuje aj nová funkcionalita a spôsoby riešenia. Vždy zhodnotíme čo sme dosiahli od minulého stretnutia. Na týchto stretnutiach tím konzultuje výsledky aj s vedúcim projektu. Projekt riešime agilným prístupom SCRUMom. Na lepšiu efektivitu práce používame tabuľu, kde prilepujeme úlohy

napísané na malých papierikoch, pre lepšiu motiváciu a predstavivosť. Každú úlohu ohodntíme hlasovaním cez scrum kartičky, tak že najnižšiu a najvyššiu hodnotu vylúčime a s ostatných spravíme priemer. Šprinty plánujeme na 2 týždne.

Úlohy manažéra komunikácie:

- dohliadanie na regulérnosť komunikácie
- komunikácia so zákazníkom
- informovanosť celého tímu
- návrhy komunikácie

17 Manažment rozvrhu a plánovania

Manažment rozvrhu a plánovania je dôležitou súčasťou každého vývoja softvérového produktu. Je dôležitý najmä pre vyhodnocovanie splnenia stanovených termínov, dohliadanie na ukončenie projektu včas, prípadne na zapracovanie zmien do rozvrhu. Tvorba rozvrhu a plánovanie nášho tímového projektu je ovplyvnené používaním agilnej metodiky SCRUM. Táto metodika nám rozdeľuje vývoj softvérového produktu do dvojtýždňových intervalov predstavujúcich šprinty. Vývoj prebieha v iteratívnom a inkrementálnom postupe, a preto nie je možné vopred zostaviť detailný rozvrh a plán tímového projektu. Ucelený hrubý plán projektu je vopred zostavený z dvojtýždňových šprintov a míľnikov predstavujúcich kontrolné body odovzdania počas celého zimného a letného semestra. Tento hrubý plán sa vždy postupne pre začatím nového šprintu zjemňuje pre daný nasledujúci šprint. Hlavnými úlohami manažéra rozvrhu a plánovania sú:

- Zostavenie hrubého plánu pre tím na daný semester
- Dozeranie na postupné zjemňovanie hrubého plánu v šprintoch
- Vyhodnotenie naplnenia plánu, prípadné návrhy pre korekciu plánu
- Sledovanie míľnikov v harmonograme na zimný semester
- Kontrola a zaistenie, aby sa tímový projekt ukončil včas

18 Manažment kvality projektu

Manažér kvality zastáva v tíme dôležitú úlohu, pretože zodpovedá za kvalitu výsledného produktu ako celku. S týmto procesom sa spájajú úlohy ako dohliadanie nad zdrojovým kódom, ktorý musí spĺňať primeranú úroveň kvality. S projektom sa spájajú isté funkcionálne a nefunkcionálne požiadavky, vzhľadom na zdrojový kód, ktoré musí manažér kvality zabezpečovať a sledovať.

Jednou z hlavných nefunkcionálnych požiadaviek je spôsob písania zdrojového kódu vzhľadom na metodiku, ktorá bola pre tieto účely vytvorená. Dodržiavanie tejto metodiky je dôležité z hľadiska prehľadnosti zdrojového kódu a jeho následného pochopenia. Okrem zdrojového kódu, manažér kvality dohliada tiež na kvalitu ostatných aspektov tvorby projektu ako dokumentácia, prehliadky kódov a tvorba testov.

Z funkcionálnych požiadaviek manažér kvality sleduje kvalitu projektu z hľadiska testovania. Vyvára a sleduje vytvorené testy, na základe ktorých vyhodnocuje odolnosť projektu voči chybám. Ďalej na základe vytvorených jednotkových testov vytvára testovacie scenáre, ktoré slúžia na overenie ucelenej funkcionality. Na základe výsledkov týchto testov sa rozhoduje o nasadení verzie do produkcie.

Hlavnou úlohou manažéra kvality je:

- sledovanie dodržiavania jednotlivých metodík členmi tímu
- sledovanie konvencií písania zdrojových kódov členmi tímu
- refaktorovanie zdrojového kódu
- odstraňovanie redundancie kódu
- dohliada na testovanie

19 Manažment rizík

V úvodných fázach projektu je potrebné reagovať na nové vznikajúce riziká, ktoré súvisia s tvorbou strategických rozhodnutí, alebo úvodným formovaním tímu. Projekt vyvíjaný na vysokej škole, ako súčasť vyučovaného predmetu Tímový

projekt, má svoje špecifické riziká, na ktoré je taktiež potrebné v rámci manažmentu rizík patrične reagovať.

19.1 Riziká nákladov

Špecifickou skupinou rizík sú riziká nákladov, ktoré sú eliminované povahou projektu, nakoľko je vyvíjaný na akademickej pôde. Z tohto dôvodu neexistujú žiadne finančné náklady na mzdy a taktiež nám sú poskytnuté vyhradené priestory pre prácu na projekte.

19.2 Riziká rozvrhu

Ku rizikám rozvrhu môžeme z interného hľadiska zaradiť úvodný studený štart, zaškolenie do nových technológií, pracovné vyťaženie a rozdielnosť rozvrhov členov tímu. Ku externým rizikám patrí nutnosť komunikácie a spolupráce s externými tímami. Pre vyššiu efektivitu a prevenciu rizík boli vypracované metodiky určujúce priebeh tímových stretnutí, reportovania stavu projektu a metodika programovania pre používané technológie. Dôležitým nástrojom pre tímovú prácu a jej efektivitu sa venuje podpora vývoja, v rámci ktorej bola vypracovaná metodika manažmentu verzií. Monitorovanie rizík rozvrhu prebieha pomocou tímového nástroja Redmine a jeho rozšírenia pre agilný vývoj, poskytujúci vlastnú formu burn-down chartu, Gantov graf, alebo sledovanie stavu jednotlivých úloh v projekte. Prevencia externých rizík je v stave riešenia.

19.3 Riziká splnenia požiadaviek

Eliminovanie rizík spojených s plnením požiadaviek pomáhajú riešiť princípy agilného vývoja v rámci metodiky SCRUM a taktiež vypracované metodiky pre vytváranie dokumentácie a testovania.

20 Prílohy

Zápis z 2. stretnutia tímu č. 8

Téma	Úvodné pokyny k tímovému projektu
Dátum	3.10.2013
Čas:	10:00 – 12:00
Miestnosť:	Jobsovo softvérové štúdio, FIIT STU
Členovia:	Bc. Ivan Košdy
	Bc. Jozef Marcin
	Bc. Michal Juranyi
	Bc. Tomáš Martinkovič
	Bc. Matej Noga
	Bc. Ján Podmajerský
	Bc. Juraj Rabčan
Stretnutie viedol:	doc. Mgr. Daniela Chudá, PhD.
Zápis vypracoval:	Tomáš Martinkovič
Nasledujúci zápis vyhotoví:	Matej Noga

Priebeh stretnutia

- Úvodné pokyny k písaniu dokumentácie k projektu
  - pozostáva z 2 častí - inžinierske dielo, riadenie projektu
  - odovzdávanie 2-krát za semester
  - zápisy sú súčasťou dokumentácie
- Budeme pracovať na projekte metodikou SCRUM
  - dvojtýždňové šprinty
- Pohľad na tému zo strany zákazníka
  - Logovanie – dynamika klávesnice, myši
  - Modelovanie - efektivita
  - Rozpoznanie
  - Kvalita – softvérové metriky, podobnosť, efektivita

- Vybrať si nástroj pre manažment projektu, verziovanie dokumentov
- Úlohou projektu nie je naprogramovať nový logger, môžeme použiť existujúci
- Každý tím musí mať svoju webovú stránku - priblíženie témy projektu, o fotky a charakteristiky jednotlivých členov tímu, o udržiavanie aktuálnej dokumentácie na stránke
- Otázka ohľadom prihlásenia na TP cup sa rozoberie na dlhšom stretnutí
- Vyriešené prvotné problémy ohľadom rozvrhu - čas stretnutia je štvrtok 12:00 a zákazník bude prítomný na začiatku a konci stretnutia
- Predstavenie sa zákazníkovi, v akej roli budú jednotlivý členovia tímu vystupovať
  - prerozdelenie rolí je možné v LS
  - ako by mala prebiehať diskusia v tíme
  - aká osobnosť by mal byť vedúci tímu (vie dať dokopy celý tím, komunikatívny, dobrý programátor, má rešpekt, vie postrážiť termíny)

Stanovenie úloh do ďalšieho stretnutia:

ID	Zadanie úlohy	Pridelenie úlohy	Dátum pridelenia	Dátum splnenia
1.0	Analýza existujúcich loggerov	Marcin	03.10.2013	10.10.2013
1.1	Analýza modelov používateľa	Noga	03.10.2013	10.10.2013
1.2	Analýza hodnotenia kvality	Martinkovič, Rabčan	03.10.2013	10.10.2013
1.3	Prezrieť súvisiace DP, BP	Martinkovič	03.10.2013	10.10.2013
1.4	Vybrať najvhodnejší nástroj pre verziovanie dokumentov	Juranyi	03.10.2013	10.10.2013
1.5	Vytvorenie webstránky tímu	Košdy	03.10.2013	10.10.2013
1.6	Analýza EmLog, PerConIK	Podmajerský	03.10.2013	10.10.2013
1.7	Vybrať najvhodnejší nástroj pre manažment projektu	Juranyi, Košdy	03.10.2013	10.10.2013
1.8	Vytvoriť šablónu pre dokumentáciu v LaTeX	Rabčan	03.10.2013	10.10.2013

Obr. 2: *Zoznam úloh, ktoré treba vypracovať počas šprintu*

Zápis z 3. stretnutia tímu č. 8

Téma	Úvodné pokyny k tímovému projektu
Dátum	10.10.2013
Čas:	12:00 – 15:00
Miestnosť:	Jobsovo softvérové štúdio, FIIT STU
Členovia:	Bc. Ivan Košdy
	Bc. Jozef Marcin
	Bc. Michal Juranyi
	Bc. Tomáš Martinkovič
	Bc. Matej Noga
	Bc. Ján Podmajerský
	Bc. Juraj Rabčan
Stretnutie viedol:	doc. Mgr. Daniela Chudá, PhD.
Zápis vypracoval:	Matej Noga
Nasledujúci zápis vyhotoví:	Ján Podmajerský

Vyhodnotenie úloh z predchádzajúceho stretnutia:

Vyhodnotenie úloh z predchádzajúceho stretnutia:

ID	Zadanie úlohy	Pridelenie úlohy	Dátum pridelenia	Dátum splnenia	Stav
1.0	Analýza existujúcich loggerov	Jozef Marcin	03.10.2013	10.10.2013	splnená
1.1	Analýza modelov používateľa	Matej Noga	03.10.2013	10.10.2013	splnená
1.2	Analýza hodnotenia kvality	Tomáš Martinkovič, Juraj Rabčan	03.10.2013	10.10.2013	splnená
1.3	Prezrieť súvisiace DP, BP	Tomáš Martinkovič	03.10.2013	10.10.2013	splnená
1.4	Vybrať najvhodnejší nástroj pre správu verzií dokumentov	Michal Juranyi	03.10.2013	10.10.2013	splnená
1.5	Vytvorenie webstránky tímu	Ivan Košdy	03.10.2013	10.10.2013	splnená
1.6	Analýza EmLog, PerConIK	Ján Podmajerský	03.10.2013	10.10.2013	splnená
1.7	Vybrať najvhodnejší nástroj pre manažment projektu	Michal Juranyi, Ivan Košdy	03.10.2013	10.10.2013	splnená
1.8	Vytvoriť šablónu pre dokumentáciu v LaTeX	Juraj Rabčan	03.10.2013	10.10.2013	splnená

Obr. 3: *Zoznam úloh, z minulého stretnutia*

Priebeh stretnutia

- Rozbor metodiky SCRUM. Tím pozrel krátke inštruktážne video o metodike SCRUM a následne prebiehal rozbor pozretého videa v rámci tímu.
- Naplánoval sa postup práce v Jobsonovom štúdiu. Tím sa dohodol na používaní tabule a nalepovaní popísaných papierových štítkov rôznej farby na tabuľu.
- Stručný prehľad a analýza loggerov použiteľných vo vývojovom prostredí Eclipse – Fluoride, Perconik.
- Tím sa zhodol, že pre správne identifikovanie používateľa je potrebné nie len zachytávať akcie vykonané klávesnicou a myšou, ale aj následná analýza používateľom napísaného kódu. Nutnosť hlbšieho preskúmania softvérových metrík vyhodnocujúcich kvalitu kódu.
- Tím sa dohodol na pravidelných sedeniach v dĺžke pár minút na začiatku každého stretnutia, kde každý člen povie, čo od posledného stretnutia mal urobiť, čo urobil, prípadne prečo to neurobil a čo do ďalšieho stretnutia

plánuje urobiť. Na konci každého sedenia sa zhodnotí vykonaná práca.

- Krátky rozbor plagiátorských techník.
- Návrh na tvorbu spoločnej tímovej fotografie.
- Dohodnuté vlastnosti programu:
  - modelovanie používateľa
  - Rozpoznanie používateľa
  - meranie používateľovej aktivity
  - vyhodnocovanie nazbieraných údajov
- Tím sa zhodol, že jedným z cieľov je marketing. Musíme vedieť presvedčiť ľudí, že používaním nášho programu o nich nezbierame citlivé osobné údaje, ale pomáhame zamestnávateľovi/používateľovi zlepšiť výkonnosť a efektívnosť tvorby softvéru.
- Tím sa zhodol na zadávaní malých úloh jednotlivým členom, kde je možné takéto úlohy aj rýchlo plniť a spraviť ich viacej. Zložité úlohy sa preto budú rozdeľovať medzi viacerých členov tímu.

Stanovenie úloh do d'alšieho stretnutia:

ID	Zadanie úlohy	Pridelenie úlohy	Dátum pridelenia	Dátum splnenia
2.1	Analýza loggerov pre prostedie Eclipse, Analýza metód rozpoznávania modelov/používateľov	Jozef Marcin	15.10.2013	17.10.2013
2.2	Rozpoznanie	Ján Podmajerský	14.10.2013	17.10.2013
2.3	Komunikácia Eclipse pluginu (Perconik) s tray agentom	Juraj Rabčan	15.10.2013	17.10.2013
2.4	Analýza, rozbehanie a konzultácia s tvorcom Eclipse pluginom	Matej Noga	13.10.2013	17.10.2013
2.5	Analýza softvérových metrík	Tomáš Martinkovič	14.10.2013	17.10.2013
2.6	Analýza objektovo-orientovaných metrík	Michal Juranyi	14.10.2013	17.10.2013
2.7	WSDL technológie	Ivan Košdy	14.10.2013	17.10.2013

Obr. 4: *Zoznam úloh, ktoré treba vypracovať počas šprintu*

Zápis zo 4. stretnutia tímu č. 8

Téma	Perconik
Dátum	17.10.2013
Čas:	12:00 – 15:00
Miestnosť:	Jobsovo softvérové štúdio, FIIT STU
Členovia:	Bc. Ivan Košdy
	Bc. Jozef Marcin
	Bc. Michal Juranyi
	Bc. Tomáš Martinkovič
	Bc. Matej Noga
	Bc. Ján Podmajerský
	Bc. Juraj Rabčan
Pedagóg	doc. Mgr. Daniela Chudá, PhD.
Stretnutie viedol:	Juraj Rabčan
Zápis vypracoval:	Ján Podmajerský
Nasledujúci zápis vyhotoví:	Juraj Rabčan

Vyhodnotenie úloh z predchádzajúceho stretnutia:

ID	Zadanie úlohy	Pridelenie úlohy	Dátum pridelenia	Dátum splnenia	Stav
2.1	Analýza loggerov pre prostedie Eclipse, Analýza metód rozpoznávania modelov/používateľov	Jozef Marcin	14.10.2013	17.10.2013	hotovo
2.2	Rozpoznávanie modelov	Ján Podmajerský	14.10.2013		nevyriešené
2.3	Komunikácia Eclipse pluginu (Perconik) s tray agentom	Juraj Rabčan	14.10.2013	17.10.2013	hotovo
2.4	Analýza, rozbehanie a konzultácia s tvorcom Eclipse pluginom	Matej Noga	14.10.2013	17.10.2013	hotovo
2.5	Analýza softvérových metrík	Tomáš Martinkovič	14.10.2013	17.10.2013	hotovo
2.6	Analýza objektovo-orientovaných metrík	Michal Juranyi	14.10.2013	17.10.2013	hotovo
2.7	WSDL technológie	Ivan Košdy	14.10.2013		začaté

Obr. 5: *Zoznam úloh, z minulého stretnutia*

Priebeh stretnutia

- Treba určiť požiadavky a ich priority a následne prebiehal rozbor pozretého videa v rámci tímu.
- Vytvoriť už konkrétne úlohy a priradiť ich jednotlivým členom tímu a nalepovaní popísaných papierových štítkov rôznej farby na tabuľu.
- Na konci šprintu spraviť šprint review - zhodnotenie splnenia požiadaviek, prípadne urobiť scrum korekcie, aby sme na záver mali všeky požiadavky splnené.
- Následne nám pani docentka vysvetlila, naše pozície v šprintoch. Poukázala tiež na nesprávne formulovanie niektorých papierikov, ktoré máme dosť nejasné.
- Následne Jozef Marcin povedal o rozpoznávaní užívateľa cez rytmiku písania, key stroke Dynamics:
  - Časové hodnotenie, ako rýchlo používateľ píše trojicu, štvoricu písmen

- Ďalšie metriky:
  - Používa ľavý, alebo pravý shift, čí si zapne capslock
- Z týchto metód sú spravené testy buď pomocou časového diagramu (nízka úspešnosť) alebpo pomocou euklidovej vzdialenosti (vyššia úspešnosť cca. 80 %)
- Logovanie
  - logujem používateľa
  - zbieram dáta o ňom
  - analýza dát
  - určenie modelu používateľa
- Ďalej Michal Juranyi povedal niečo o objektovo orientovaných metrikách:
  - zameriava sa na počty metód, ich prepojenia a objekty
  - ukazujú ako kvalitné kódy ľudia píšu.
  - závisia tieto metriky od programátora alebo od projektu? Treba spoznať človeka sa dá podľa toho ako píše, softvérové metriky hovoria o kvalite práce
- Charakteristiky
  - hustota chýb - určí sa kvalita programovania
  - dokumentácia k veľkosti kódu
  - počet warningov
  - čas strávený nad kódom (riadok za čas)
  - komentár za čas
  - ak nevieme koho kód analyzujeme, nebudeme z neho získavať údaje, iba cez logovanie používateľa
- Na záver nám Pavol Zbell vysvetli fungovanie perconika a čo tam dorobil, aký plugin na sledovanie v eclipse.

- komunikácia prebieha z pluginu eclipsu, cez uaca do databázy
- plugin je v Jave a komunikuje s ostatnými časťami perconika
- môžeme celý ten systém a posielať tie zozbierane dáta na našu databázu
- klávesnica a myš, je to jeden watcher, ktorý je priamo napojený na uacu.

Stanovenie úloh do ďalšieho stretnutia:

ID	Zadanie úlohy	Pridelenie úlohy	Dátum pridelenia	Dátum splnenia
3.1	Rozbehanie uacy, úvod do dokumentácie	Juraj Rabčan	17.10.2013	24.10.2013
3.2	Získať prístup ku gratex databáze	Ján Podmajerský	17.10.2013	24.10.2013
3.3	Nájsť jedinečné charakteristiky	Jozef Marcin	17.10.2013	24.10.2013
3.4	Analyzovať zlogované dáta	Matej Noga	17.10.2013	24.10.2013
3.5	Ukladať nalogované dáta	Ivan Košdy	17.10.2013	24.10.2013
3.6	Základné logovanie	Tomáš Martinkovič	17.10.2013	24.10.2013
2.2	Rozpoznanie	Ján Podmajerský	17.10.2013	24.10.2013

Obr. 6: *Zoznam úloh, ktoré treba vypracovať počas šprintu*

Zápis z 5. stretnutia tímu č. 8

Téma	Spracovanie dát z UACY
Dátum	24.10.2013
Čas:	12:00 – 15:00
Miestnosť:	Jobsovo softvérové štúdio, FIIT STU
Členovia:	Bc. Ivan Košdy
	Bc. Jozef Marcin
	Bc. Michal Juranyi
	Bc. Tomáš Martinkovič
	Bc. Matej Noga
	Bc. Ján Podmajerský
	Bc. Juraj Rabčan
Pedagóg	doc. Mgr. Daniela Chudá, PhD.
Stretnutie viedol:	Bc. Michal Juranyi
Zápis vypracoval:	Bc. Juraj Rabčan
Nasledujúci zápis vyhotoví:	Bc. Michal Juranyi

Vyhodnotenie úloh z predchádzajúceho stretnutia:

ID	Zadanie úlohy	Pridelenie úlohy	Dátum pridelenia	Stav
2.1	Rozbehanie uacy, úvod do dokumentácie	Juraj Rabčan	24.10.2013	Hotovo
2.2	Získať prístup ku gratex databáze	Ján Podmajerský	24.10.2013	Hotovo
2.3	Nájsť jedinečné charakteristiky	Jozef Marcin	24.10.2013	Hotovo
2.4	Analyzovať nalogované dáta	Matej Noga	24.10.2013	Hotovo
2.5	Ukladať nalogované dáta	Ivan Košdy	24.10.2013	hotovo
2.6	Základné logovanie	Tomáš Martinkovič	24.10.2013	Hotovo
2.7	Rozpoznanie	Ján Podmajerský	24.10.2013	Hotovo
2.8	Umiestnenie projektu na gitovy repozitar	Michal Juranyi	24.10.2013	Hotovo

Obr. 7: *Zoznam úloh, z minulého stretnutia*

Priebeh stretnutia

- Na úvod sme sa rozprávali o TP-Cupe. Rozprávali sme sa o tom ako prebieha táto súťaž a aké výhody prináša účastníkom. Dohodli sme sa, že pošleme prihlášku a zúčastnime sa súťaže.
- Po to ako sme prebrali TP-Cup všetci členovia tímu povedali či splnili a ako splnili úlohy z minulého šprintu. Úlohy sa podarilo splniť každému členovi tímu.
- Následne nasledovala porada o ďalšej práci, ktorá bude prebiehať na projekte. Dohodli sme sa, že údaje, ktoré získame z UACY budeme logovať do databázy MongoDB. Mongo bude nainštalované na serveri, ktorý bol pridelený nášmu tímu. Z tejto databázy sa budú dáta priebežne odosielať do databázy na školskom servery.

Stanovenie úloh do ďalšieho stretnutia:

ID	Zadanie úlohy	Pridelenie úlohy	Dátum pridelenia
3.1	Rozchodenie Monga lokálne, rozšírenie Eclipse o git plugin, študovanie Springu, zápisnica	Juraj Rabčan	24.10.2013
3.2	TP CUP prihláška, nainštalovanie Gitu, Eclipse a Mongo lokálne	Ján Podmajerský	24.10.2013
3.3	Porovnanie vektorov, ktoré reprezentujú model používateľa	Jozef Marcin	24.10.2013
3.4	Model používateľa z nalogovaných dát	Matej Noga	24.10.2013
3.5	Ukladať nalogované dáta na server, JSP, nastavenie Springu	Ivan Košdy	24.10.2013
3.6	Rozchodenie Monga lokálne, rozšírenie Eclipse o git plugin, študovanie Springu, prihláška na TP cup	Tomáš Martinkovič	24.10.2013
3.7	Inštalácia Monga na server, príprava prezentácie na prednášku.	Michal Juranyi	24.10.2013

Obr. 8: *Zoznam úloh, ktoré treba vypracovať počas šprintu*

Zápis zo 6. stretnutia tímu č. 8

Téma	Analýza a vizualizácia získaných dát
Dátum	7.11.2013
Čas:	12:00 – 15:00
Miestnosť:	Jobsovo softvérové štúdio, FIIT STU
Členovia:	Bc. Ivan Košdy
	Bc. Jozef Marcin
	Bc. Michal Juranyi
	Bc. Tomáš Martinkovič
	Bc. Matej Noga
	Bc. Ján Podmajerský
	Bc. Juraj Rabčan
Pedagóg	doc. Mgr. Daniela Chudá, PhD.
Stretnutie viedol:	Bc. Ivan Košdy
Zápis vypracoval:	Bc. Michal Juranyi
Nasledujúci zápis vyhotoví:	Bc. Ivan Košdy

Vyhodnotenie úloh z predchádzajúceho stretnutia:

ID	Zadanie úlohy	Pridelenie úlohy	Dátum pridelenia	Stav
3.1	Rozchodenie Monga lokálne, rozšírenie Eclipse o git plugin, študovanie Springu, zápisnica	Juraj Rabčan	24.10.2013	Hotovo
3.2	TP CUP prihláška, nainštalovanie Gitu, Eclipse a Mongo lokálne	Ján Podmajerský	24.10.2013	Hotovo
3.3	Porovnanie vektorov, ktoré reprezentujú model používateľa	Jozef Marcin	24.10.2013	Hotovo
3.4	Model používateľa z nalogovaných dát	Matej Noga	24.10.2013	Hotovo
3.5	Ukladať nalogované dáta na server, JSP, nastavenie Springu	Ivan Košdy	24.10.2013	Hotovo
3.6	Rozchodenie Monga lokálne, rozšírenie Eclipse o git plugin, študovanie Springu, prihláška na TP cup	Tomáš Martinkovič	24.10.2013	Hotovo
3.7	Inštalácia Monga na server, príprava prezentácie na prednášku.	Michal Juranyi	24.10.2013	Hotovo

Obr. 9: *Zoznam úloh, z minulého stretnutia*

Priebeh stretnutia

- Úvod stretnutia bol venovaný zhodnoteniu práce na zadaných úlohách. Tie boli vyhodnotené ako úspešne splnené.
- V ďalšej časti stretnutia sme identifikovali problémy na ktoré sme narazili. Išlo najmä o to, že sme z databázy nezískavali žiadne dáta o práci v IDE. V tejto súvislosti sme preverovali, či máme nainštalované najnovšie verzie programov (vyšli nové verzie PerConIK programov).
- Hlavnú časť stretnutia sme venovali identifikácii nových úloh v súvislosti so spracovaním doposiaľ získaných dát. Hlavnou úlohou bolo získané údaje zobraziť a vizualizovať pomocou grafov. Ďalšou témou bolo zefektívnenie vytvárania modelov používateľov s čím súvisí otázka periodického získavania a spracovania dát z databázy PerConIK a vzorkovania (časový úsek, za aký sa budú údaje agregovať a analyzovať).
- V závere sme ešte identifikovali bočné úlohy súvisiace s nasadzovaním

projektu na tímový server a rozdelili sme si všetky identifikované úlohy.

Stanovenie úloh do ďalšieho stretnutia:

ID	Zadanie úlohy	Pridelenie úlohy	Dátum pridelenia
3.1	Preskúmanie použitia Eclipse pluginov pre výpočet SW metrík	Jozef Marcin	7.11.2013
3.2	Zobraziť nalogované dáta pomocou grafu s filtrom	Ján Podmajerský	7.11.2013
3.3	Zobraziť nalogované dáta pomocou tabuľky s filtrom	Juraj Rabčan	7.11.2013
3.4	Zobraziť aktuálne modely používateľov	Matej Noga	7.11.2013
3.5	Periodické získavanie dát registrovaných používateľov	Ivan Košdy	7.11.2013
3.6	Otestovať prácu s PerConIK Eclipse pluginom a implementovať jednoduchú LOC metriku	Tomáš Martinkovič	7.11.2013
3.7	Inštalácia GlassFish na tímový server	Michal Juranyi	7.11.2013
3.8	Autodeployment pomocou Git a Maven	Ivan Košdy	7.11.2013
3.9	Preveriť získavanie dát z Eclipse	Michal Juranyi	7.11.2013
3.10	Identifikácia jednotlivých prvkov vektora	Jozef Marcin	7.11.2013

Obr. 10: *Zoznam úloh, ktoré treba vypracovať počas šprintu*

Zápis zo 7. stretnutia tímu č. 8

Téma	Dopracovanie úloh zadaných na začiatku šprintu 3
Dátum	14.11.2013
Čas:	12:00 – 15:00
Miestnosť:	Jobsovo softvérové štúdio, FIIT STU
Členovia:	Bc. Ivan Košdy
	Bc. Jozef Marcin
	Bc. Michal Juranyi
	Bc. Tomáš Martinkovič
	Bc. Matej Noga
	Bc. Ján Podmajerský
	Bc. Juraj Rabčan
Pedagóg	doc. Mgr. Daniela Chudá, PhD.
Stretnutie viedol:	Bc. Michal Juranyi
Zápis vypracoval:	Bc. Jozef Marcin
Nasledujúci zápis vyhotoví:	Bc. Michal Juranyi

Vyhodnotenie úloh z predchádzajúceho stretnutia:

ID	Zadanie úlohy	Pridelenie úlohy	Dátum pridelenia	Stav
3.1	Rozchodenie Monga lokálne, rozšírenie Eclipse o git plugin, študovanie Springu, zápisnica	Juraj Rabčan	24.10.2013	Hotovo
3.2	TP CUP prihláška, nainštalovanie Gitu, Eclipse a Mongo lokálne	Ján Podmajerský	24.10.2013	Hotovo
3.3	Porovnanie vektorov, ktoré reprezentujú model používateľa	Jozef Marcin	24.10.2013	Hotovo
3.4	Model používateľa z nalogovaných dát	Matej Noga	24.10.2013	Hotovo
3.5	Ukladať nalogované dáta na server, JSP, nastavenie Springu	Ivan Košdy	24.10.2013	Hotovo
3.6	Rozchodenie Monga lokálne, rozšírenie Eclipse o git plugin, študovanie Springu, prihláška na TP cup	Tomáš Martinkovič	24.10.2013	Hotovo
3.7	Inštalácia Monga na server, príprava prezentácie na prednášku.	Michal Juranyi	24.10.2013	Hotovo

Obr. 11: *Zoznam úloh, z minulého stretnutia*

Priebeh stretnutia

- V úvode stretnutia sme s Bc. Hudákom diskutovali o riešení ich tímového projektu, ktorý sa venoval emocionálnej stránke programátora.
- V ďalšej časti stretnutia sme hodnotili úlohy zadané na predošlom stretnutí z čoho vyplynula potreba niektoré úlohy prepracovať.
- Podstatnú časť stretnutia tvorila rozprava o odovzdávaní výsledkov po prvých troch šprintoch, kde nám doc. Chudá vysvetlila všetky potrebné náležitosti odovzdávania. Následne jej bola predvedená funkcionalita momentálneho stavu nášho prototypu.
- Zvyšok stretnutia bol venovaný identifikácií nových úloh, respektíve prepracovaniu úloh zadaných na začiatku šprintu a bol dohodnutý scenár odovzdávaného prototypu po prvých troch šprintoch.
- V závere sme pridelili úlohy jednotlivcom a dohodli sa na ďalšom stretnutí s doc. Chudou, na ktorom jej predvedieme hotový výsledok po troch šprintoch.