SLOVENSKÁ TECHNICKÁ
UNIVERZITA V BRATISLAVE
FAKULTA INFORMATIKY
A INFORMAČNÝCH TECHNOLÓGIÍ

Tím 15

DOKUMENTÁCIA RIADENIA

Tímový projekt

Študijný odbor: Informačné systémy, Softvérové inžinierstvo
Miesto vypracovania: Ústav informatiky a softvérového inžinierstva, FIIT STU Bratislava
Vedúca tímu: Ing. Nadežda Andrejčíková, PhD.

December 2014

Obsah

Kapitola 1

Big picture

Úvod

Dokument obsahuje dokumentáciu k riadeniu projektu Big data kvality a ich využitie pri validácii a obohacovaní existujúcich dát. Obsah dokumentu zodpovedá aktuálnemu stavu projektu v danej etape. Projekt vypracúva tím 15 s názvom Nitrogen Family.

Zloženie tímu:

- Bc. Marek Černák
- Bc. Martin Šustek
- Bc. Ján Krivý
- Bc. Vladimír Demčák
- Bc. Martin Cibula
- Bc. Jozef Melko
- Bc. Juraj Michalička

Vedúca tímu: Ing. Nadežda Andrejčíková, PhD.

1.1 Roly členov a podiel práce

- **Vedúci tímu** - Bc. Martin Šustek
  Kompletizácia dokumentu - 30%
  Vytváranie obsahu - 20%
- **Zástupca vedúceho a hlavný architekt** - Bc. Juraj Michalička
  Vytváranie obsahu - 5%
- **Manažér rozvrhu a plánovania** - Bc. Martin Cibula
  Vytváranie obsahu - 10%
- **Manažér rizík** - Bc. Vladimír Demčák
  Vytváranie obsahu - 15%
- **Manažér podporných prostriedkov a vývoja** - Bc. Marek Černák
  Kompletizácia dokumentu - 30%
  Vytváranie obsahu - 20%
- **Manažér kvality a dokumentácie** - Bc. Ján Krivý
  Kompletizácia dokumentu - 30%
  Vytváranie obsahu - 20%
- **Manažér komunikácie a ľudských zdrojov** - Bc. Jozef Melko
  Vytváranie obsahu - 10%

1.2 Aplikácie manažmentov

1.2.1 Plánovanie

Zodpovedná osoba: Martin Cibula

Úlohou tohto manažmentu je zabezpečiť plánovanie úloh počas celého vývoja produktu ako aj dohľad nad ich plnením.

Spôsob plánovania

Pri vývoji využívame ako metodiku vývoja techniku SCRUM. Je to agilná technika vývoja založená na inkrementálno-iteratívnom vývoji. Vývoj softvéru prebieha v iterácia, ktoré sa nazývajú šprinty, pričom výstupom každého šprintu by mal byť nejaký funkčný celok, ktorý sa rozširuje v ďalších šprintoch. Šprinty prebiehajú vo vopred stanovených časových úsekoch, v našom tíme sme si stanovili dva týždne trvajúce šprinty.

Plánovanie šprintu

Na začiatku každého šprintu sa vykonáva retrospektívne zhodnotenie predchádzajúceho šprintu, z ktorého zistíme do akej miera sa plán na predchádzajúci šprint

splnil. Na základe retrospektívy sa následne vytvára plán na nasledujúci šprint. V prvom rade treba prihliadať na úlohy, ktoré sa nestihli spraviť počas predchádzajúceho šprintu. Ideálny stav je ak takéto úlohy nie sú. Následne sa vytvoria veľké ciele pre nasledujúci šprint. Tieto ciele sa rozpracujú a rozbijú do niekoľkých, už konkrétnych úloh, potrebných na splnenie daného cieľa. Veľmi dôležitou časťou plánovanie šprintu je určenie priority vykonávania úloh a čo najpresnejší odhad času potrebného na vykonanie úlohy. Odhadovanie času sa rieši metódou plánovacieho pokra, kde každý člen tímu má karty s časovými hodnotami a pre úlohu, pre ktorú sa odhaduje potrebný čas na vykonanie, vyberie kartu s takou časovou hodnotou, o ktorej si myslí, že určuje čas vykonávania danej úlohy. Výsledný čas je priemerom odhadovaných časov všetkých členov tímu pre danú úlohu.. Na úspešnosti splnenia cieľov má veľký podiel vhodné a rovnomerné rozdelenie úloh medzi členov tímu práve na základe priority a odhadu potrebného času na vykonanie pridelených úloh. Úlohy treba rozdeliť tak aby sa čo najefektívnejšie využili ľudské zdroje v tíme.

Miera plnenia plánu

Na začiatku vývoja manažér plánovania vytvoril približný plán cieľov šprintov na celý zimný semester. V súčasnom časti by sme podľa plánu mali mať vytvorený funkčný servlet , ktorý vyhľadáva informácie o dokumentoch a aj ohlasy z WOS a SCOPUS a buduje zo všetkých informácií databázu. Taktiež by mal ukladať získane fulltexty do mongo databázy. Plán sa nám za prvé dva šprinty darilo plniť avšak v tretom sme časovo trocha zaostali. Rozhodli sme sa na základe zhodnotenia priorít funkcionalít posunúť ukladanie fulltextov do mongoDB až na ďalšie šprinty, pretože sa nestihli implementovať niektoré potrebné časti a rozhodli sme sa implementovať testovacie rozhranie aby sme umožnili testovanie produktu treťou osobou. Taktiež servlet ešte nie je úplne dokončený keďže bolo potrebné zapracovať zmeny workflowu servletu.

Podporné prostriedky

Ako podporný nástroj pre plánovanie a evidenciu úloh sme si vybrali nástroj Jira. Tento nástroj slúži na evidenciu projektov, úloh, zdrojov a ľudí. Veľkou výhodou tohto nástroja je možnosť exportu štatistík formou diagramov, ktoré si na základe práce používateľov, tento nástroj sám vytvára. Tieto štatistiky slúžia ako pohľad na priebeh vývoja a sú užitočné pre ďalšie plánovanie. Jeden z dôvodov prečo sme si tento nástroj zvolili bolo, že podporuje agilné metódy vývoja ako je SCRUM, ktorým sa riadime. Umožňuje radenie úloh do jednotlivých šprintov a zobrazenie úloh na nástenke. Správe úloh v Jire sa venujeme viac v metodike: Životný cyklus issue v Jire.

1.2.2 Manazment vyvoja a podpornych prostriedkov

Zodpovedná osoba: Marek Černák

Pre prácu na našom projekte bola potrebná voľba prostriedkov, ktoré nám spomedzi ponúknutých najviac vyhovujú a sme si istí, že s ich pomocou budeme môcť úspešne dosiahnuť stanovené ciele. Oblasti, pre ktoré používame podporné prostriedky sú:

- Verziovanie
- Priraďovanie úloh
- Písanie zdrojového kódu
- Komunikácia
- Umiestnenie webovej služby
- Ukladanie do databáz
- Tvorba dokumentácie

Verziovanie

Pre verziovanie nášho produktu sme sa rozholi použiť Git, pretože s ním mala väčšina členov tímu dobré skúsenosti. Pre zobrazenie zmien v jednotlivých verziách a zobrazenie obsahu aktuálnej verzie používame webovú službu Bitbucket. Pre aktualizáciu verzií v Gite, ale aj v lokálnych projektoch jednotlivých členov používame prevažne plugin do Eclipsu (pozri časť "Písanie zdrojového kódu") s názvom EGit. Viac informácií o procese verziovania sa nachádza v Metodike verziovania a spravy verzií, ktorej autorom je Vladimír Demčák.

Priraďovanie úloh

V prípade priraďovania sme sa rozhodovali medzi viacerými nástrojmi, no nakoniec sme sa rozhodli pre nástroj Jira, s ktorým už niekoľko členov tímu pracovalo a vedia sa v ňom velmi dobre orientovať. Výhodou bolo aj jednoduché zavedenie nástroja, keďže sa nachádza aj na školskom serveri a o prístup stačilo iba požiadať.

Písanie zdrojového kódu

Keďže pre výtváranie kódu nášho produktu používame programvací jazyk Java, jednohlasne sme súhlasili s používaním prostredia Eclipse, s ktorým má každý člen tímu najlepšie slúsenosti. Eclipse, okrem množstva iných funkcií, je schopný kód aj kompilovať a ponúka aj nástroj na debugovanie.

Komunikácia

Komunikácia v tíme prebieha pomocou komunikačného nástroja Skype, v ktorom máme vytvorený skupinový rozhovor. Pre zdieľanie dokumentov používame cloudovú službu Dropbox.

Umiestnenie webovej služby

Pre umiestnenie webovej služby, ktorú vytvárame používame webový server Tomcat 7. Tento server nám bol odporúčený projektovým vedúcim.

Ukladanie do databáz

Jedna z funkcionalít služby vytváranej v projekte je aj ukladanie do lokálnych databáz. Pre ukladanie objektov bol zriadený Postgresql server na ktorom je vytvorená vždy databáza pre poslednú stable verziu a ďalšia vytvorená pre testovacie účely. Pre ukladanie fulltextov bola vytvorená MongoDB databáza.

Tvorba dokumentácie

Dôležité dokumenty ohľadom chodu projektu (workflovy, dtb modely, zápisnice stretnutí, plány šprintov, konfiguračné parametre, navody, kontakty, dokumentácie používaných služieb) sa nachádzajú na webovej službe Bitbucket, pomocou ktorej sme si vytvorili vlastnú Wikipediu. Dokumentácia zdrojového kódu je generovaná pomocou nástroja JavaDoc.

1.2.3 Analýza rizík

Zodpovedná osoba: Vladimír Demčák

Manažér rizík máza úlohu v rámci projektu identifikovať možné riziká, ktoré môžu počas práce na projekte vzniknúť. Stratégiou manažmentu rizík je predísť nepríjemným situáciám, prekvapeniam alebo iným nepríjemnostiam, ktoré by mohli ohroziť vývoj projektu.

Manažér rizík musí v každom projekte musí pochopiť kontext projektu, predvídať skryté požiadavky zákazníka, reálne zhodnotiť časové vyťaženie členov vývojového tímu a schopnosti jednotlivých členov tímu.

Pre identifikované riziká navrhuje možné riešenia a opatrenia, ako predísť reálnemu naplnenia rizika. Pre riziká definuje možné dosahy na projekt a hodnotu rizika. Hodnota predstavuje pravdepodobnosť nastania rizika.

Manažér rizík identifikované riziká interpretuje všetkým členom tímu. Po tímových stretnutiach a po stretnutiach so zákazníkom sú riziká analyzované a aktualizované.Identifikované riziká sú konzultované s vedúcim tímu.

V projekte boli identifikované riziká, ktoré sú uvedené v tabuľke „Tab. Identifikované riziká“.

Č.	Názov rizika	Scenár	Riešenie a opatrenia	Hodnota rizika	Dosah na projekt
1	Nepochopenie zadania	Nedostatočné pochopená problémová doména projektu a požiadavky zákazníka.	• Konzultácia so zákazníkom • Spoločná analýza problému v rámci tímu • Kontaktovanie tímu z minulého roka	stredná	vysoký
2	Integrácia API tretích strán	Nepodarí sa integrovať API zo systémov, z ktorých získavame ohlasy na diela (citačné indexy Scopus a WOS)	• Kontaktovať J. Klimesa • Testovať API manuálne v klientoch pre REST alebo SOAP	stredná	vysoký
3	Prístup na verejný server	Nepodarí sa získať prístup na server, ktorý má prístup k databázam s citačnými indexmi	• Kontaktovať vedúceho tímu • Dočasne používať VPN STU • Pristupovať k databázam zo siete YNET v netlabe	stredná	vysoký
4	Nesprávne pridelenie manažérskych rolí	Tím si nesprávne rozdelil roly a niektorá z manažérskych oblastí funguje neefektívne alebo nedostatočne.	• Výmena manažérskych pozícií • Podpora neefektívnej manažérskej pozície	stredná	stredný
5	Synchronizácia v rámci tímu	Členovia tímu majú rozdielne rozvrhy a mimoškolské aktivity a je nemožné stretávať sa mimo tímového stretnutia v rámci cvičení.	• Dohodnúť stretnutie na skype • Dohodnúť stretnutie cez víkend	vysoký	vysoký
6	Slabá dokumentácia projektu	Projekt je nedostatočne dokumentovaný a je veľmi slabo komentovaný zdrojový kód.	• Vybrať 2 členov tímu, ktorým sa pridelí review zdrojového kódu a dokumentácia projektu	stredná	vysoký
7	Nastavenie prostredia lokálneho vývoja	Lokálne prostredie (Tomcat, GIT ..) sa nepodarí nastaviť všetkým členom tímu.	• Definovať vývojový tím (s fungujúcim lokálnym nastavením) a tím dokumentácie • Nastaviť lokálny vývoj všetkým členom tímu na spoločnom stretnutí • Vypracovať dokumentovaný postup ako nastaviť lokálne prostredie	vysoká	nízka
8	Nevhodný výber podporných technológií	V rámci analýzy projektu bola/boli nevhodne vybrané technológie (knižnice), ktoré sú používané v projekte.	• Nájsť alternatívnu technológiu s rovnakou funkcionalitou • Implementovať vlastnú technológiu • Obísť problém implementovaním podpornej funkcionality	stredná	stredný
9	Malá produktivita niektorého z členov	Niektorí člen tímu nepracuje dostatočne na svojich úlohách. Nekomunikuje s členmi tímu a neprejavuje záujem pracovať na projekte.	• Oznámiť problém vedúcemu tímu • Porozprávať sa a motivovať problémového člena • Ponúknuť inú pozíciu v tíme	nízka	stredný
10	Privátny projekt	Nezabezpečený prístup k projektovým dokumentom a zdrojovým kódom (projekt má mať uzavretý zdrojový kód)	• Požiadať vedúceho tímu o prístup k privátnemu repozitáru • Uchádzať sa o študentskú licenciu v BitBucket • Nainštalovať vlastné git prostredie na školskom serveri • Zaplatiť Premium účet na githube	vysoká	stredný
11	POST metóda	Požiadavka zákazníka je vytvorenie 3 POST metód. Predpokladalo sa vytvorenie 1 POST metódy.	• preprogramovanie POST metódy na 3 POST metódy • diskusia so zákazníkom sosnahou zachovať aktuálne riešenie	nízka	vysoké

Tab. Identifikované riziká

Každé identifikované riziko je analyzované a má pridelené nižšie uvedené parametre.

- Názov rizika - riziko má pridelený názov, ktorý ho stručne charakterizuje
- Scenár - stručný opis rizika a možnosti jeho nastania
- Dosah na projekt - definovanie, ako vážne dokáže riziko ovplyvniť vývoj a prácu na projekte

- Hodnota rizika - pravdepodobnosť (váha), že sa riziko nastane počas projekt
- Riešenie a opatrenia - definované možné riešenia a opatrenia pre identifikované riziká. Snahou je uviesť viac možných riešení, vďaka ktorým je neskôr možné rýchlejšie reagovať na nepríjemné situácie

1.2.4 Kvalita dokumentácie

Zodpovedná osoba: Ján Krivý

Manažér dokumentácie má v našom tíme na starosti celistvosť a úplnosť dokumentácie. Kontroluje jednotlivé časti dokumentácie vytvorenej inými členmi a zodpovedá za ich kvalitu. Ďalšou úlohou manažéra dokumentácie je kontrola zdokumentovania jednotlivých issue v systéme pre správu úloh - Jira.

Rôzne typy dokumentácie a dokumentov

Počas trvania projektu sa vytvárajú rôzne druhy dokumentácie a dokumentov. Každý typ dokumentácie ma svoje pravidlá, ktoré sa týkajú formy, obsahu, štruktúry a člena tímu za to zodpovedného. Na formu a štruktúru tímových dokumentov bola vytvorená metodika - Písanie a identita tímových dokumentov, podľa ktorej sa vyhotovujú všetky tímové dokumenty.

- **Dokumentácia v kóde**

  Pravidlá pre písanie komentárov boli stanovené v metodike komentovania zdrojového kódu a jeho dokumentovanie pomocou JavaDoc. Veľkú pozornosť venujeme kvalitnému JavaDocu, ktorý vlastne slúži aj ako voľne dostupná dokumentácia API, ktoré vyvíjame. Za vytváranie JavaDocu a komentovanie zdrojového kódu je zodpovedný autor daného kódu, resp. metódy.

- **Dokumentácia vo wiki**

  V tímovej wiki sa nachádza celá dokumentácia k projektu, alebo aspoň odkazy na ňu. Za vytvorenie a správu wiki zodpovedá manažér dokumentácie.

  V tímovej wiki sa nachádzajú tieto typy dokumentov:

  - Kontakty na členov tímu
  - Plán projektu
  - Popisy šprintov
  - Odkazy na zápisnice
  - Roly členov tímu
  - Odkazy na projektové denníky
  - Dokumentácia k inžinierskemu dielu

  - Odkazy na dokumentáciu API citačných indexov
  - Odkazy na metodiky
  - Návody k podporným prostriedkom(PostgreDB,MongoDB,Tomcat)

- **Zápisy zo stretnutí**

  - Zápisnica zo stretnutia - pri každom oficiálnom tímovom stretnutí vzniká zápisnica zo stretnutia. Zápisnicu vytvárajú postupne všetci členovia tímu, neskôr manažér dokumentácie. Na zápisnicu bola vytvorená šablóna, ktorá sa nachádza vo wiki
  - Retrospektíva - retrospektíva vzniká pri stretnutí, kde sa ukončuje a hodnotí šprint. Za jej vyhotovenie je zodpovedný manažér dokumentácie. Na retrospektívu bola vytvorená šablóna, ktorá sa nachádza vo wiki
  - Plán šprintu - po úvodnom stretnutí šprintu je manažér plánovania zodpovedný za vyhotovenie plánu šprintu a jeho následné pridanie do wiki

- **Metodiky**

  V rámci tohto projektu, boli zatiaľ vytvorené tieto metodiky:

  - Písanie a identita tímových dokumentov
  - Komentovanie zdrojového kódu a jeho dokumentovanie pomocou JavaDoc
  - Priebeh šprintu
  - Metodika verziovania a správy verzií
  - Štábna kultúra písania kódu v jazyku Java
  - Životný cyklus issue
  - Spôsob komunikovania a komunikačné kanály

- **Dokument na odovzdanie**

  Počas trvania projektu sa vytvára dokument na odovzdanie, ktorý sa skladá z 2 častí a to: Riadenie projektu a Inžinierske dielo. Tento dokument sa vytvára podľa šablóny v latexu. Tento dokument ma na starosti manažér dokumentácie, ktorý dbá nato, aby všetci členovia tímu doplnili do dokumentu, všetky im prislúchajúce časti.

Zásady tvorby dokumentácie

- Dokumentáciu vytváram počas priebehu jednotlivých udalostí
- Pri vytváraní dokumentov sa riadim metodikou a používam šablóny na to určené

- Každý dokument (okrem JavaDocu) sa musí nachádzať v tímovej wiki (alebo aspoň odkaz naň)
- Dokumentáciu vytváram stručne a jasne, tak aby jej rozumeli všetci členovia tímu

1.2.5 Komunikácia

Zodpovedná osoba: Jozef Melko

Komunikácia v tíme je veľmi dôležitá. Má rôzne podoby od písaného textu, cez obrázky až po hovorené slovo. Spôsob komunikácie v tíme je prostredníctvom komunikačných kanálov.

Skype

Skype používame v prípade, že máme s niečím problém a nevieme s ním pohnúť. Skype môžeme použiť aj v prípade, že s niekým na nejakom riešení spolupracujeme. Nepíšu sa tam správy typu kto má čo urobiť, lebo na to je určená JIRA. Ak s niekým spolupracujeme alebo potrebujeme niečo od konkrétneho člena, píšeme mu správu osobne a nepíšeme to do skupinovej konverzácie. Ak máme nejakú dôležitú správu, ktorá by sa nemalastratiť, tak to nepíšeme do skupinovej konverzácie ale použijeme iný komunikačný kanál. Skype takisto používame na hovor alebo videohovor. Praktizujeme v prípade, že sa vedúca tímového projektu osobne zúčastniť nemôže. Skupinový hovor alebo videohovor uskutočňujeme ak je stretnutie neosobné ale má mať náplň osobného stretnutia, akoby sme boli spolu na rovnakom mieste. Vzhľadom na kvalitnú komunikáciu v tíme je skype veľmi dobrým a využívaným kanálom. V komunikácií zastáva funkciu ako normálna konverzácia. Funguje ako chat, kde sa členovia tímu rozprávajú, radia si ohľadom projektu. V tomto komunikačnom kanály neexistuje žiadna šablóna, podľa ktorej by mali členovia formulovať svoje správy.

Nástenka na facebook-ovej skupine

Na nástenku sa píšu príspevky, ktoré nechceme aby sa rýchlo stratili v skupinovom chate na skype alebo aby sa na ne nezabudlo na stretnutí. Sú to veci ako prihlasovacie údaje k veciam potrebným na bezproblémový beh projektu. Na nástenku facebook stránky sa píšu aj príspevky spojené s informáciami od vedúcej TP alebo iné dôležité informácie. V prípade, že vieme dôležitú informáciu hoci z e-mailu, tak je potrebné ju hodiť na nástenku, lebo nie každý môže byť tak pohotový, aby si tú informáciu na spoločnom maily prečítal včas.Ďalšie príspevky ktoré nechceme, aby sa stratili sú informácie, ktoré sme sa dozvedeli skôr ako budú potrebné. Keďže sa použijú v projekte neskôr tak je dôležité ich dať na miesto kde budú jednoducho prístupné a budú dobre viditeľné. Skupina na facebooku je dobrá aj v prípade, keď sa informácia viaže k nejakému dátumu, ktorý treba striktne dodržať. Ak je potrebné o niečom hlasovať, v tom prípade sa vytvára v skupine príspevok na hlasovanie.

Podobne ako na skype, tak ani na facebooku nie je štruktúra, podľa ktorej je nutné písať príspevky.

Ak máme informáciu k projektu od osoby, je dôležité napísať od koho to je. Takisto je dôležité stručne a jasne formulovať čoho sa to týka, čo sa od nás vyžaduje. Medzi dôležité informácie patria aj tie o konaní stretnutí. Je dôležité sa dohodnúť na vhodnom termíne pre všetkých. Keďže existuje demokracia tak to znamená, že sa tím nebude podriaďovať menšine ohľadom času stretnutia. Keďže chceme mať kvalitnú komunikáciu tak sa vyberá najvhodnejší termín. Následne sa napíše na nástenku skupiny informácie o stretnutí:

- Dátum a čas
- Čo sa má riešiť na stretnutí
  - meno člena
  - čo by chcel riešiť – stručný popis, prípadne s kým by chcel daný problém riešiť

Stretnutia

Stretnutia sú minimálne raz do týždňa. Povinné stretnutie je vo štvrtok. Na tomto stretnutí zhodnocujeme celý týždeň. Čo sme urobili a čo nie. Čo je na koľko percent splnené. Na týchto stretnutiach sme všetci členovia tímu a aj vedúca projektu. Čiže v rámci dobrej komunikácie je dôležité pýtať sa ak niečomu nerozumieme, lebo je to hneď vysvetlené. Ak sa počas týždňa vyskytne viac problémov, ktoré sa dajú riešiť len osobným stretnutím, členovia tímu sa dohodnú na termíne kedy sa stretnutie uskutoční. Keďže dosť komunikujeme na skype aj v škole keď sa stretneme, tak je postačujúci aj ten jeden termín stretnutia.

E-mail

Dôležitým komunikačným kanálom v našom tíme je e-mailová adresa. Do tejto adresy nám chodia dôležité správy od osôb, ktoré v danom odbore pracujú a spolupracujú s nami na projekte. Sú to požiadavky alebo pomocné materiály, ktoré nám výrazne uľahčujú prácu. Takisto sú to správy, ktoré obsahujú informácie, ku ktorým by sme sa nedostali alebo je to literatúram ktorú odporúčajú. Na vhodnú reprezentáciu tímu a komunikovanie s externými osobami je oficiálny mail veľmi dôležitý a môžu ho používať všetci členovia.

Aby bola práca kvalitná je komunikácia veľmi dôležitá a preto je dôležité včas a vhodne komunikovať.

1.2.6 Architektúra produktu

Zodpovedná osoba: Juraj Michalička

Kvalita architektúry je jedna z najdôležitejších častí produktu, pretože odzrkadľuj jeho celkovú kvalitu. Kvalitná architektúra ďalej výrazne uľahčuje neskoršie fázy vývoja ako aj udržiavanie produktu po jeho vydaní.

Používame agilnú metódu, kde sa objavujú nové požiadavky postupne, a preto je dôležité navrhnúť a udržiavať architektúru tak, aby sa do nej nové požiadavky len pridávali a nebolo pritom potrebné výraznejšie zasahovať do už navrhnutých a implementovaných častí. Preto je najdôležitejšie si najprv ujasniť pre koho je náš produktu určený a čo to má byť, nie len aké funkcie má mať. Taktiež by sme mali poznať problematiku, ktorú má náš produkt riešiť.

Náš produkt je určený pre knižnice, ktoré majú nejaký vlastný systém. Tento ich systém má rozširovať o funkcionalitu, konkrétne získavanie ohlasov. Preto sme sa rozhodli vytvoriť jeden Java servlet. Takéto riešenie umožňuje rozšíriť existujúci systém bez priameho zásahu doň. Taktiež je toto riešenie multiplatformové, čo zvyšuje jeho univerzálnosť a tým aj znižuje nároky na zákazníka.

Napriek tomu, že pod získavaním ohlasov sa skrýva viacero častí, rozhodli sme sa je jeden servlet. Hlavne preto, že tieto jednotlivé časti na seba nadväzujú a pre požadovaný celkový výsledok sa musia využiť všetky.

Na znázornenie nadväzností používame „work flow“ diagram, ktorý sprehľadňuje fungovanie systému a podáva jednoduchý celkový náhľad na systém. Tento diagram aktualizujeme pri každom šprinte s tým ako pribúdajú nové požiadavky a dostávame sa k funkcionalite, ktorá predtým mala menšiu prioritu ako už implementované časti systému.

Pri vytváraní a udržiavaní kvalitnej architektúry nám výrazne pomáhajú diskusie na tímových stretnutiach so zadávateľom, kde sme si vydiskutovali celkový význam systému. Ďalej ako má systém fungovať aktuálne fungovať, čiže aktuálny „work flow“. Od toho sa priamo odvíjajú časti, ktoré majú momentálne najvyššiu prioritu a treba ich implementovať, aby sa dosiahol požadovaný „work flow“ pre daný šprint. Táto diskusia je dôležitou časťou každého šprintu, čo vyplýva aj z princípov agilného vývoja softvéru.

Ďalším dôležitým prvkom vplývajúcim na kvalitu architektúry je určenie si istých štandardov. Z hľadiska architektúry je dôležité si určiť ako sa budú volať jednotlivé časti systému a ako majú medzi sebou fungovať. To pomôže k homogénnosti systému a uľahčí to začlenenie nových častí.

Pôvodne bol náš systém typu „restful“ avšak potom sa ukázalo, že výhodnejšie bude pre nás využiť „JSON „ na komunikáciu medzi jednotlivými časťami. Toto riešenie nám umožnilo variabilitu vstupov a výstupov bez zbytočných komplikácií a preťažovaní metód.

Overiť kvalitu návrhu nie je tak jednoduché ako overiť kvalitu kódu, pretože sa

nedajú spustiť testy, alebo algoritmy, ktoré prejdú návrh a zhodnotia ho na základe nejakých parametrov. Nám sa osvedčilo ústne a na tabuli simulovať používateľa a systém. Takáto simulácia niektorých scenárov nám pomohla pochopiť ako systém pracuje, prípadne nepracuje, pretože v návrhu sú medzery, ktoré treba opraviť, alebo doplniť.

Takáto metóda nám umožňuje otestovať návrh ešte pred jeho implementáciou. Avšak skutočná kvalita návrhu sa ukáže až neskorších fázach vývoja, kedy sa nové časti systému budú pridávať bez problémov a zdržaní, alebo naopak bude potrebné meniť mnoho už implementovaných častí a ich architektúra. Posledné miesto kde sa najvýraznejšie ukáže kvalita, je keď projekt prevezme niekto iný a bude môcť na ňom stavať ako na dobrom základe, ktorý netreba meniť.

1.2.7 Vedenie členov tímu

Zodpovedná osoba: Martin Šustek

Úlohou vedúceho tímu je manažment výkonnosti členov tímu a delegovanie interakcie medzi nimi za účelom dosiahnutia synergického efektu, čo má predpokladaný pozitívny dopad na mieru plnenia projektového plánu. Tímový vedúci je zároveň prostredníkom medzi product ownerom a samotným tímom. Jeho úlohou teda nie je len dohliadať na plnenie úloh, ale aj zabezpečovať oboznámenosť zadávateľa projektu so situáciou v tíme. Počas týchto činností si vedie tímový vedúci rôzne, preňho užitočné, záznamy, z ktorých je možné neskôr odvodiť závery užitočné pre ďalšie smerovanie tímu alebo pre výkonostnú analýzu tímu ako celku a jeho členov samostatne. Táto podkapitola ponúka prehľad záverov získaných aplikovaním manažmentu výkonnosti členov tímu.

- **Využívanie metodík** - vedúci tímu dohliada na uplatňovanie procesov opísaných metodikami vo vhodných situáciách. Postupy v nich opísané boli navrhnuté za účelom urýchlenia práce v tíme a ich nedodržiavanie by malo za následok stagnáciu výkonnosti tímu.
- **Rozdelenie zodpovedností** - táto oblasť sa primárne zameriava na rozdeľovanie úloh členom tímu podľa roly, ktorú zastávajú. Cieľom takéhoto postupu je rovnomerné rozdeľovanie úloh podľa početnosti výskytu aj náročnosti. Nakoľko roly boli vytýčené tak, aby sa úlohy spadajúce do ich kompetencií vyskytovali rovnomerne, tento proces je zatiaľ, až na okrajové situácie, úspešný.
- **Informovanosť účastníkov** - jednou z úloh vedúceho tímu je udržiavať stav, v ktorom sú všetci členovia tímu neustále a včas informovaní o všetkých skutočnostiach a zároveň majú ucelený obraz o aktuálnom aj celkovom smerovaní projektu. Vedúci tímu však musí uvážiť aké informácie posúva členom, aby nedošlo k ich zahlteniu. Každý je teda informovaný o udalostiach spadajúcich do jeho oblasti pôsobenia.

- **Kontrola smerovania projektu** - oblasť, v ktorej vedúci tímu úzko spolupracuje s product ownerom a manžérom plánovania. V každej iterácii vyhodnocujú, či sa tím neodklonil od stanovených cieľov implementáciou úloh, ktorých výsledkom nie je požadovaná funkcionalita.
- **Dohľad nad členmi tímu** - v neposlednom rade dozerá tímový vedúci na to, aby členovia tímu plnili im pridelené úlohy. Členov tímu v dostatočnej (nie však prehnanej) miere kontroluje, motivuje k svedomitému plneniu úloh a musí všas identifikovať neschopnosť riešiteľa dotiahnuť jemu zverenú úlohu do zdarného konca.

1.3 Sumarizácie šprintov

1.3.1 1. šprint

V tomto šprinte je dôležité oboznámiť sa s technológiami a citačnými indexami (SCOPUS a WOS) a ich API. Taktiež je dôležité analyzovať metadáta v SCOPUSe resp. WOSe. Ďalším cieľom šprintu je získanie prístupu na server a inštalácia potrebných technológií.

Úlohy pre šprint:

1. vytvorenie tímového webu
2. oboznámenie sa s technológiami
3. naštudovanie dokumentácií
4. manuálne vyhľadávanie dát v Scopus-e
5. analýza dátového modelu
6. inštalácia servera
7. vytvorenie JiRa a jej naplnenie taskami
8. inštalácia Tomcat-u na server
9. návrh dátového modelu schémy pre uchovávanie ohlasov
10. návrh spôsobu ukladania FULLTEXTov
11. návrh Workflow-u od vykonania dopytu, cez uloženie medzivýsledkov, po vrátenie finálneho výsledku

Vyhodnotenie šprintu:

Hlavnú rolu v tomto šprinte nakoniec zohrala inštalácie a konfigurácie potrebných technológií, čo zahŕňalo ich štúdium. Podarilo sa nám získať prístup k softvéru Jira, ktorá je nastavená pre potreby nášho projektu a sú v nej evidované úlohy. Na poskytnutý server sme nainštalovali Tomcat pre servlet ktorý bude vyvíjaný v nasledujúcich šprintoch. Podarilo sa nám z väčšej časti oboznámiť sa s citačnými indexmi WOS a SCOPUS. Pomocou manuálnych dopytov sme získali a analyzovali ukážkové dáta. Výstupom tohto šprintu je navrhnutý základný dátový model navrhovaného systému a taktiež základný workflow vykonávania dopytov na dielo alebo ohlasy. Výstupom šprintu je taktiež vytvorené webové sídlo nášho tímu s uloženými dokumentmi vytvorenými počas tohto šprintu. Počas návrhu sme sa rozhodli na ukladanie dát získaných z citačných indexov ukladať do PostrgreSQL databázy a ukladať fulltexty do mongo databázy. Tieto databázy však ešte nie sú nenainštalované.

1.3.2 2. šprint

Po prvom šprinte je tím je oboznámený s technológiami a vývojové prostredie je takmer kompletne pripravené. Pristupujeme teda k vytvoreniu prvej verzie aplikácie. Je potrebné stanoviť pravidlá pre prácu v tíme (komunikácia, zdieľané úložisko, spôsob dokumentovania, ...). Na konci šprintu počítame s funkčným prototypom, ktorý bude získavať a ukladať jednoduché dopyty.

Úlohy pre šprint:

1. vytvoriť jednoduché servlety pre získavanie záznamov ako aj ohlasov z WOS a SCOPUS APIs na základe základných parametrov
2. budovať vlastnú databázu záznamov a ich identifikátorov z pracovísk univerzít, WOS-u, SCOPUS-u spolu s väzbou na ohlasy na tieto dokumenty a prípadné fulltexty
3. manažér komunikácie stanoví princípy a pravidlá komunikácie
4. implementácia dopytov na Elsevier Scopus
5. príprava Git repozitára
6. dopyty na Thomson WOS pomocou API verzie 3.0
7. príprava PostgreSQL databázy spolu s navrhnutím a implementáciou dátového modelu
8. príprava databázy MongoDB spolu s vytvorením hierarchie kolekcií

9. vytvorenie plánu projektu a plánov pre aktuálny šprint
10. vytvorenie Wiki projektu
11. vytvorenie testov pre doteraz implementovanú funkcionalitu
12. zdokumentovanie doterajšieho štúdia knihovníckych štandardov

Vyhodnotenie šprintu:

Keďže nastali isté problémy s dopytovaním sa do citačných indexov, práca sa značne spomalila. Pre uskutočnenie dopytu do SCOPUSu bolo potrebné pripojiť sa cez školskú IP adresu a bolo potrebné prejsť na najnovšiu verziu Thomson API a to na verziu 3.0 pre dopytovanie sa z WOSu. Výstupom sú 4 Get metódy:

1. na získavanie záznamov dokumentov z WOSu na základe mena autora a názvu diela
2. na získavanie záznamov dokumentov zo SCOPUSu na základe mena autora a názvu diela
3. na získavanie ohlasov na dokument z WOSu na základe WOS ID
4. na získavanie ohlasov na dokument zo SCOPUSu na základe názvu diela

Výstupom týchto metód je zatiaľ iba xml dokument vrátený z citačných indexov. Implementovaný servlet je už prepojený s databázou PostrgeSQL pomocou objektovo relačného mapovača hibernate, pričom je navrhnutá, a vytvorená schéma databázy, avšak získané dáta zatiaľ neukladáme. Taktiež je nainštalovaná na servery Mongo databáza na ukladanie fulltextov. Na zdieľanie a spoluprácu pri implementácií sme si založili privátny repozitár na bitbucket.org. Využili sme taktiež možnosť vytvorenia Wiki v bitbuckete, kde zhromažďujeme všetky vytvorené dokumenty.

1.3.3 3. šprint

Je vytvorená prvá verzia servletov. V priebehu šprintu je potrebné ich funkcionalitu zjednotiť a navzájom poprepájať. Jadro funkcionality aplikácie je síce testovateľné, je však potrebné pokryť všetky funkcionality testami. Je potrebné prerobiť aplikáciu aby sa všetky správy vymieňali vo formáte JSON. Aktuálny stav zdrojových kódov sme identifikovali potrebu refaktoringu kódu.

Úlohy pre šprint:

1. vytvorenie testovacích JSON dát, ktoré budú slúžiť ako vstupné údaje pre unit testy
2. refaktoring kódu

3. uplatnenie softvérového návrhu pre rozbitie monolitických tried do hierarchie
4. pokrytie aplikácie testami
5. vytvorenie POST metódy pre vyhľadanie diela a ukladanie ohlasov na dielo
6. ukladanie získaných údajov do databázy
7. vytvorenie hierarchie výnimiek reflektujúcich chybové kódy vracané APIs citačných indexov, pomocou ktorých bude možné tieto rýchlo identifikovať a upozorniť na ne (prípadne ich ošetriť)
8. zaznamenanie konfiguračných hodnôt skritpov, programov, podporných prostriedkov a utilít do Wiki, aby každý z členov mal možnosť s nimi odborne manipulovať

Vyhodnotenie šprintu:

Výstupom tohto šprintu je refactorovaný zdrojový kód, v ktorom je implementovaná hlavná POST metóda servletu, ktorá na základe typu požiadavky, prijatej vo formáte JSON, volá metódy, ktoré boli vytvorené z pôvodných GET metód. Tieto metódy získané dáta uložia do databázy a poskytujú ich na výstup vo forme JSON súboru. Spôsob vyhľadávania ohlasov zo SCOPSu bol rozšírený, vyhľadáva sa na základe mena autora, názvu diela a názvu zdroja daného diela. Nie je však implementované ošetrenie chybného vstupu, teda ak chýba niektorá zo spomínaných položiek. Identifikovali sme taktiež potrebu zmeny workflow získavania ohlasov, na základe, ktorých budú musieť byť metódy na získavanie ohlasov upravené. Funkcionality aplikácie sú z väčšej časti pokryté testami, je však potrebné doplniť niektoré scenáre. Pre možnosť odbornej manipulácie každého člena tímu s podpornými prostriedkami a nástrojmi boli zdokumentované ich konfigurácie do tímovej Wiki.

1.4 Používané metodiky

- **Metodika spôsobu komunikácie a komunikačné kanály**
  Touto metodikou sa riadia členovia tímu pri vzájomnej komunikácií alebo pri komunikácií s inými osobami zainteresovanými v projekte. Pre každý komunikačný kanál metodika určuje pravidlá pre jeho využívanie a taktiež situácie, v ktorých má byť použitý. Na dodržiavanie týchto pravidiel dohliada manažér komunikácie.
- **Štábna kultúra písania kódu v jazyku Java**
  Táto metodika určuje pravidlá pre všetkých členov zasahujúcich do kódu. Týka sa ich to pri písaní zdrojového kódu v prípade programovania, testovania, refactoringu a generovania JavaDoc. Metodika určuje pravidlá pre písanie zdrojového kódu aby ho členovia tímu písali správne a aby ho písali rovnako.

- **Životný cyklus issue v Jire**
  Na základe tejto metodiky sa riadia všetci členovia tímu pri vytváraní a uzatváraní issue v Jire. Určuje pravidlá, musia byť použité pri vytváraní tasku a takisto aj ako uzavrieť task podľa pravidiel.

- **Metodika verziovania a správy verzií**
  Touto metodikou sa riadia všetci členovia tímu Nitrogen Family. Je veľmi dôležité aby bol kód písaný zrozumiteľne a práve preto je potrebné tieto pravidlá dodržiavať. Dodržiavajú sa pri každom commite programu. Riadime sa podľa nich ako správne commitovať, kde využívame štruktúru práve podľa tejto metodiky.

- **Písanie a identita tímových dokumentov**
  Táto metodika ja pre každého člena tímu Nitrogen Family, ktorí pracujú na dokumentoch a prezentáciách projektu. Je dôležité dodržiavať štruktúru pri písaní dokumentov ako retrospektíva alebo zápisnica a a pri iných formálnych dokumentov ohľadom projektu. Pravidlami tejto metodiky sa riadime pri formálnom prezentovaní tímového projektu.

- **Komentovanie zdrojového kódu a jeho dokumentovanie pomocou JavaDoc**
  Pomocou tejto metodiky postupujeme pri komentovaní zdrojového kódu. Určuje pravidlá, ktorými sa riadime pri komentovaní zdrojového kódu priamo v kóde alebo pri písaní komentárov pre nástroj JavaDoc. Riadia sa ňou členovia, ktorí sa podieľajú na písaní zdrojového kódu programu. Na dodržiavanie pravidiel dozerá manažér kvality a dokumentácie.

- **Priebeh šprintu**
  Táto metodika slúži na správny priebeh šprintu. Členovia tímového projektu sa ňou riadia počas celého šprintu. Pracujú podľa určitých pravidiel, ktoré sú v tejto metodike. Sú pravidlá pre začiatok šprintu jeho polčas a takisto aj jeho koniec, ktoré je potrebné dodržať pre hladký priebeh celého šprintu.

Kapitola 2

Zoznam kompetencií tímu

2.1 Prehľad členov tímu

Bc. Marek Černák

Skúsenosti:

- Java, C, MySQL
- XML, Data mining z webu

Absolvent bc. štúdia na fakulte FIIT v odbore INFO. V bakalárskej práci nadobudol vedomosti v oblasti zhlukovacích algoritmov a navrhol svoj vlastný, respektíve zdokonalil už používaný algoritmus. Jeho silnou stránkou je zodpovednosť nadobudnutá počas práce na projektoch ako v práci, tak aj v škole.

Bc. Martin Cibula

Skúsenosti:

- Java, C, C++
- MySQL
- vyše rok programovania dotazníkov a spracovávanie zozbieraných anketárskych dát

Najviac skúseností získal pri práci na bakalárskej práci kde si prehĺbil znalostiprogramovaní v Jave a získal nové vedomosti o získavaní dát z webu, spolupráci s webovými aplikáciami a práci s XML. Má taktiež nejaké skúsenosti s webovými technológiami. Jeho silnou stránkou je schopnosť dobre si naplánovať čas a zefektívniť tak svoju prácu, vďaka čomu doposiaľ nezmeškal žiaden deadline.

Bc. Vladimír Demčák

Skúsenosti:

- Java, Android, C
- JBoss AS, REST WS, MySQL

Absolvent bc. štúdia na FIIT odbor Informatika. V rámci bakalárskej práce sa venoval odporúčaniu článkov, ktoré sú používatelovi prezentované prostredníctvom mobilného zariadenia s OS Android. V rámci tejto práce nadobudol skúsenosti aj s XML a JSON formátmi.

Bc. Jan Krivý

Skúsenosti:

- C, Objective-C, Python(Framework Flask), MySQL
- FB, Google API
- Pracovné skúsenosti: vývoj mobilných aplikácií - iOS development

Absolvent bc. štúdia na FIIT odbor Informatika. V rámci bakalárskej práce sa venoval hudobným odporúčaniam a konkrétne problému studeného štartu, ktorý pri nich vzniká. Momentálne pracuje ako iOS developer.

Bc. Jozef Melko

Skúsenosti:

- Java, C
- MySQL

Absolvent bakalárskeho štúdia na FIIT odbor informatika. V rámci bakalárskej práce sa venoval neštruktúrovaným peer-to-peer sieťam kde vytváral algoritmy pre vyhľadávanie zdrojov v týchto sieťach. Najviac skúseností má s programovacími jazykmi C a Java, v ktorých programoval aj bakalársku prácu. Rád pracuje aj s databázami konkrétne ho zaujalo MySQL, ktoré obľubuje.

Bc. Juraj Michalička

Skúsenosti:

- HTML, CSS, PHP, MySQL v kombinácii s MVC(framework CodeIgniter)
- Jazyky z rodiny C na mierne pokročilej až pokročilej úrovni
- C#(.NET Framework) - semestrálny projekt zo sietí(fiit) a ročníkový projekt(matfyz)
- vývoj mobilných aplikácií pre Android

Bakalárska práca zameraná na zaznamenávanie jázd pomocou mobilného zariadenia GPS a synchronizáciu so serverom.

Bc. Martin Šustek

Skúsenosti:

- HTML5, Less, Grunt, PHP (CakePHP framework), Javascript, Twitter Bootstrap
- MongoDB, MySQL
- Erlang, Bash
- práca s údajmi vo formátoch XML a JSON, základy tvorby Unit testov
- Linux, Git, Jira

Absolvent bc. štúdia na FIIT odbor Informatika. V rámci bakalárskej práce sa venoval tvorbe rámca pre vytváranie škálovateľných aplikácií, ktorý bol testovaný v distribuovanom prostredí. V súčasnosti pracuje ako vývojár webových aplikácií.

2.2 Vybrané témy

1. **Dav proti vizuálnemu smogu**

   Prečo nás téma zaujala:

   - zhodli sme sa na tom, že vizuálny smog v naších mestách, najmä v našom hlavnom meste, je seriózny problém,
   - reklama podľa nás nie len dehonestuje naše mestá ale napríklad aj rozptyľuje vodičov,
   - v tomto projekte vidíme veľký potenciál, kedže tento problém vníma veľmi veľa ľudí, ktorých by sme mohli našou aplikáciou osloviť.

   Náš tím môže pre danú temú ponúknuť:

   - skúsenosti s vývojom mobilných aplikácií pre platformu Android v rámci bakalárskych prác,
   - skúsenosti s vývojom backendu pre android mobilné aplikácie na strane servera JBOSS AS, REST WS,
   - inovatívne nápady,
   - sme pripravení zapojiť do vývoja aplikácie jej cieľových používateľov a nové príspevky vytvárať s prihliadnutím na ich postrehy,
   - máme skúsenosti s vytváraním webových aplikácií s použitím niektorého z na to určených frameworkov,
   - viacerí z nás majú skúsenosti s funkcionálnymi jazykmi a vývoj pod platformou Ruby on Rails by bol pre nás zaujímavou výzvou.

   Naše nápady pre prínos k téme:

   - rôznymi spôsobmi ľudí motivovať (napr. spôsobom, aký používa aplikácia Waze),
   - inšpirovať sa pri použití gamifikácie projektami ako Foursquare či Duolingo,
   - pridať offline režim zberu dát zachytených mobilným zariadením,
   - vytvorenie hodnotení používateľov ako motivácia (podobne ako avatary vo Waze). Vyšší rank = vyššie právomoci a zviditeľnenie používateľa v dave,
   - identifikácia poškodených reklám (poškodená reklama pôsobí hrozivejšie ako nová),

- možnosť zaujať reklamné spoločnosti, ktoré majú straty z dôvodu obrovského množstva ilegálnych reklamných plôch.

2. **Virtuálna FIIT**

   Prečo nás téma zaujala:

   - téma je spojením viacerých informačných systémov do jednej aplikácie,
   - použitie GPS technológií považujeme za zaujímavý spôsob rozšírenia aplikáci okrem, navigácie vonku je potrebné sa zamerať aj na navigáciu a uľahčenie orientácie v komplexných budovách,
   - všetci sme používateľmi a zároveň vekými fanúšikmi aplikácie VirtualFIIT a potešil nás fakt, že má začať využívať Beacon vysielače.

   Náš tím môže pre danú temú ponúknuť:

   - záujem o nové technológie,
   - skúsenosti s vývojom mobilných aplikácií pre platformu Android v rámci bakalárskych prác,
   - skúsenosti s vývojom backendu pre android mobilné aplikácie na strane serveru JBOSS AS, REST WS,
   - inovatívne nápady,
   - schopnosť rýchlo naštudovať solídny základ, ktorý aplikácia už má, a promptne do nej integrovať novú funkcionalitu.

   Naše nápady pre prínos k téme:

   - vylepšenie používateľskeho prostredia,
   - vyššia miera personalizácie aplikácie pre konkrétneho používateľa,
   - testovanie a možné postupné nasadzovanie v nákupných centrách,
   - použiť aplikáciu nie len v rámci fakulty ale aj komerčne - napr. pri vstupe do,
   - nákupných centier (človek nosí informačnú tabuľu vo vrecku).

3. **Distribuovaný výpočtový systém - webové rozhranie**

   Prečo nás téma zaujala:

   - myslíme si, že tento dobrý nápad si zaslúži reprezentatívne webové rozhranie,
   - keďže pre projekty tohto typu ešte neexistuje žiadne oficiálne rozhranie, vidíme to ako príležitosť vytvoriť také rozhranie, ktoré zaujme a prevezmú ho aj iné univerzity.

Náš tím môže pre danú tému ponúknuť:

- skúsenosti s vývojom webových aplikácií,
- schopnosť rýchlo sa oboznámiť aj s komplexnejším backendom a začať preň vytvárať rozhranie,
- nadšenie pre myšlienku zdieľania procesorového času,
- pedantnosť pri bezpečnom spracovávaní používateľských údajov.

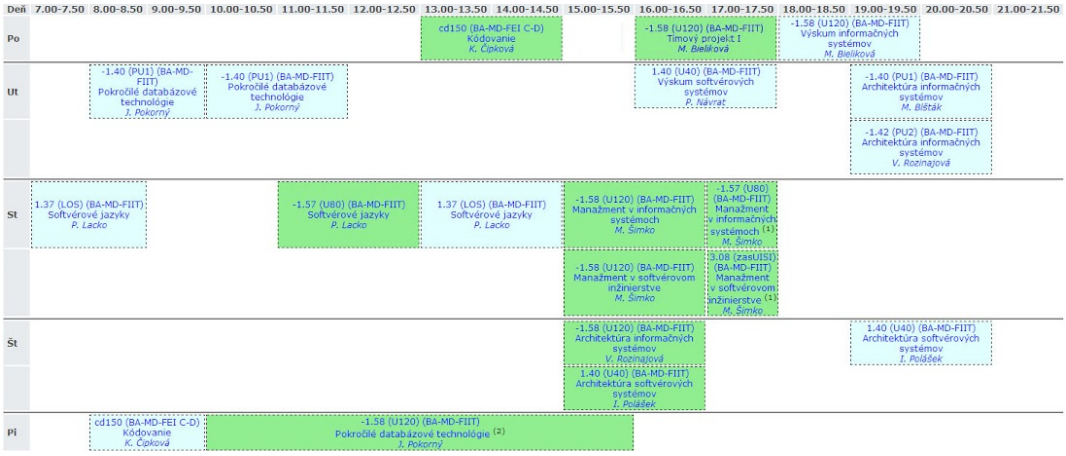
Naše nápady pre prínos k téme:

- rozhranie chceme navrhnúť tak, aby ani používateľ, ktorý nie je úplný "geek" nemal problém s jeho ovládaním a aby vyžadovalo čo najmenej úsilia pre zapojenie nového člena do gridu,
- spopularizovanie projektu vytvorením identity.

2.3 Témy zoradené podľa priority

1. Dav proti vizuálnemu smogu
2. Virtual FIIT
3. Distribuovaný výpočtový systém - webové rozhranie
4. Inteligentný mobilný asistent
5. Calendar people matcher
6. Platforma pre monitorovanie chýb a iných udalostí v aplikácií
7. Hra pre mobilné zariadenia šitá na mieru osobnosti hráča
8. Sofistikované spracovanie dát a ich prezentácia
9. Vizualizácia informácií v obohatenej realite
10. Eyeblink - vyhodnocovanie frekvencie žmurkaní používateľa
11. Big data kvality a ich využitie pri validácií a obohacovaní existujúcich dát
12. 3D UML improved version

Rozvrh všetkých členov tímu



Kapitola 3

Priebeh šprintu - Metodika

Autor: Martin Šustek

Vymedzenie obsahu a dedikácia metodiky

Metodika pojednáva o dianí počas šprintu v rámci metodiky Scrum. Riadia sa ňou študenti 1. ročníka inžinierskeho štúdia na FIIT STU počas predmetu Tímový projekt I. Opísané postupy sú platné pre členov tímu číslo 15 počas školského roka 2014/2015.

Zoznam nadväzujúcich metodík a dokumentov

- Správa verzií (Vladimír Demčák)
- Písanie a identita tímových dokumentov (Juraj Michalička)
- Kanály a spôsoby komunikácie (Jozef Melko)
- Životný cyklus tasku (Ján Krivý)

Vymedzenie pojmov

tímová WIKI - https://bitbucket.org/BigDataFiit/bigdata/wiki/Home
tímový web - http://labss2.fiit.stuba.sk/TeamProject/2014/team15is-si/
JiRa – systém na správu úloh
JiRa Agile – podporný prostriedok pre vedenie metodiky Scrum s využitím KanBan tabule
Scrum – iteratívny a inkrementálny rámec pre proces vývoja produktu
KanBan – plánovací systém deliaci úlohy do skupín podľa stavu v ktorom sa nachádzajú
BigPicture – výsledný výstupný dokument z pôsobenia v predmete Tímový projekt

Súhrn rôl a zodpovedností účastníkov

- **Product owner** - schvaľuje rozhodnutia o smerovaní implementácie výsledného produktu hodnotí výstupy po každom týždni šprintu poskytuje odborné rady týkajúce sa problémovej oblasti
- **Vedúci tímu** - zvoláva a, ak nie je určené inak, vedie tímové stretnutia rozdeľuje úlohy ich riešiteľom
- **Manažér plánovania** - určuje rozsah funkcionality implementovanej v rámci šprintu určuje prioritu riešenia úloh
- **Manažér rizík** - vykonáva analýzu rizík a upozorňuje členov tímu na možnosť ich nastania
- **Manažér dokumentácie** - dopĺňa Big Picture a pridáva záznamy do tímovej WIKI dbá na to, aby členovia zdokumentovali technológie a postupy, ktoré použili pri riešení konkrétnych úloh
- **Vedúci stretnutia** - zodpovedá za naplnenie obsahu stretnutia a koriguje smerovanie v rámci neho
- **Zapisovateľ stretnutia** - zapisuje dianie na stretnutí tímu odosiela zápisnicu zo stretutia tímovej vedúcej a uverejňuje ju na určených miestach

Opisy postupov

PONDELOK	UTOROK	STREDA	ŠTVRTOK	PIATOK	SOBOTA	NEDEĽA
			ZAČIATOK ŠPRINTU	1. SYNCHRO - NIZÁCIA		
1. KONTROLNÝ BOD		1. PRÍPRAVA	**POLČAS ŠPRINTU**	2. SYNCHRO - NIZÁCIA		
2. KONTROLNÝ BOD		2. PRÍPRAVA	**KONIEC ŠPRINTU**			

Obr. 3.1: Poradie procesov v rámci šprintu

Procesy

1. **Začiatok šprintu (Tímové stretnutie)**

   Vstupné dokumenty:

   - retrospektíva predchádzajúceho šprintu (ak sa nejaký konal),
   - export diania v predchádzajúcom šprinte získaný zo systému JiRa,
   - webové obrazy projektových denníkov členov tímu,
   - analýza rizík pre začínajúci šprint.

   Priebeh:

   - tímový vedúci otvorí v systéme JiRa Agile nový šprint,
   - manažér plánovania v spolupráci s product ownerom určia funkcionality, ktoré musia byť hotové a funkčné na konci šprintu,
   - tímový vedúci vytýči 2 až 3 globálne ciele pre šprint
   - členovia tímu zúčastnení na stretnutí identifikujú čiastkové úlohy, ktoré je nutné vykonať pre naplnenie vytýčených cieľov,
   - členovia tímu zúčastnení na stretnutí ohodnotia zložitosti identifikovaných úloh (Scrum poker) a poskytnú tak optimistický a pesimistický časový odhad ich trvania, na základe čoho tímový vedúci úlohám pridelí odhadovaný čas ich trvania pre systém JiRa,
   - manažér rizík na základe pripravenej analýzy rizík prediskutuje s členmi tímu riziká, ktoré treba mať na pamäti pri riešení úloh počas šprintu,
   - tímový vedúci vykoná riadené ohliadnutie sa za uplynulým týždňom (Čo bolo dobre? Čo treba urobiť inak?).

   Výstupné dokumenty:

   - zápisnica z tímového stretnutia,
   - fotodokumentácia zápiskov zo stretnutia

2. **Synchronizácie**

   Vstupné dokumenty:

   - zápisnica z predchádzajúceho tímového stretnutia

   Priebeh:

   - pri prvej synchronizácii tímový vedúci zanesie elementárne úlohy, identifikované počas úvodného stretnutia v šprinte, spolu s odhadovanými časmi ich vykonávania do systému JiRa,

- zapisovateľ stretnutia poskytne zápisnicu zo stretnutia tímovej vedúcej a uverejní ju na tímovom webe a vo WIKI,
- pri prvej synchronizácii manažér plánovania vytvorí plán aktuálneho šprintu podľa šablóny opísanej v metodike "Písanie a identita tímových dokumentov". Takto vypracovaný plán šprintu pošle tímovej vedúcej a uverejní ho na tímovom webe a vo WIKI.

Výstupné dokumenty:

- plán šprintu bez vyplnenej kapitoly o vyhodnotení.

3. **Kontrolné body**

   Priebeh:

   - tímový vedúci uplatní postupy opísané v metodike "Kanály a spôsoby komunikácieä zvolá stretnutie ku príležitosti vzniku kontrolného bodu,
   - tímový vedúci vedie organizovanú diskusiu, počas ktorej implementátori jednotlivých funkcionalít vo vyvíjanom produkte objasňujú ostatným členom tímu ich fungovanie a za nimi stojace algoritmy.

   Výstupné dokumenty:

   - vyplývajúca dokumentácia k novoimplementovaným fukcionalitám.

4. **Prípravy**

   Priebeh:

   - tímový vedúci uplatní postupy opísané v metodike "Kanály a spôsoby komunikácieä zvolá stretnutie z dôvodu potrebnej prípravy na prezentáciu product ownerovi,
   - tímový vedúci vedie organizovanú diskusiu, v ktorej mu všetci členovia tímu oznámia v akom štádiu sú im pridelené úlohy, na základe čoho v spolupráci s manažérom plánovania identifikujú úlohy, ktoré musia byť prioritne dokončené do nasledujúceho dňa. V prípade časového sklzu navrhuje manažér plánovania úpravy rozsahu (scope) šprintu,
   - všetci členovia tímu aktualizujú webové obrazy svojich projektových denníkov.

   Výstupné dokumenty:

   - webové obrazy projektových denníkov všetkých členov tímu

5. **Polčas šprintu (Tímové stretnutie)**

   Vstupné dokumenty:

   - webové obrazy projektových denníkov členov tímu,
   - plán šprintu.

   Priebeh:

   - tímový vedúci vyhodnotí doterajší postup a mieru splnenia úloh
   - manažér plánovania určí priority pre zostávajúce úlohy a dohodne termíny ich vyriešenia s ich riešiteľmi
   - manažér komunikácie, ktorý je zároveň hlavným testerom, odprezentuje product ownerovi mieru pokrytia doteraz napísaného kódu testami. Product owner navrhne ďalšie testovacie scenáre, ktoré hlavný tester spracuje, doplní do dokumentácie a ich implementáciu pridelí konkrétnym členom tímu,
   - manažér dokumentácie doplní do dokumentácie kvantitatívnu a kvalitatívnu analýzu aktuálneho stavu zdrojového kódu.

   Výstupné dokumenty:

   - zápisnica z tímového stretnutia,
   - analýza aktuálneho stavu zdrojového kódu v dokumentácii
   - testovacie scenáre v tímovej WIKI

6. **Ukončenie šprintu (Tímové stretnutie) Vstupné dokumenty:**

   - webové obrazy projektových denníkov členov tímu,
   - plán šprintu.

   Priebeh:

   - manažér plánovania vyhodnotí dianie v šprinte a upraví na základe výsledkov časť s vyhodnotením v pláne šprintu,
   - tímový vedúci skonzultuje prírastok v produkte s manažérom rizík, ktorý má zároveň na starosti verziovanie. Pokiaľ prírastok dosiahol požadovanú úroveň, zistenú na základe metodiky "Správa verzií", tímový vedúci vytvorí v systéme JiRa novú verziu produktu, ktorá bude od tej chvíle predvolenou pre nové úlohy. Manažér rizík, ktorý je taktiež hlavný vývojár, nanovo vygeneruje JavaDoc tak, aby odzrkadľoval aktuálny stav zdrojového kódu, nasadí aktuálnu verziu podľa spomínanej metodiky a označí súčasný stav novou verziou,
   - tímový vedúci uzatvorí v systéme JiRa Agile všetky vyriešené úlohy, pričom sa riadi metodikou "Životný cyklus tasku", a uzavrie aktuálny šprint,

- manažér rizík oboznámi členov tímu s vypracovanou analýzou rizík,
- vedúci stretnutia vykoná ohliadnutie sa za končiacim šprintom (retrospektívu), z ktorej vyplynuté závery spracuje manažér dokumentácie do dokumentu podľa šablóny opísanej v metodike "Písanie a identita tímových dokumentov",
- manažér dokumentácie odošle retrospektívu tímovej vedúcej a uverejní ju na tímovom webe a vo WIKI,
- manažér dokumentácie doplní dokument Big Picture o export úloh zo systému JiRa spolu s burndown diagramami a inými výstupnými dokumentami špecifickými pre konkrétny šprint.

Výstupné dokumenty:

- retrospektíva šprintu,
- export diania v šprinte získaný zo systému JiRa,
- analýza rizík pre začínajúci šprint.

Kapitola 4

Životný cyklus issue - Metodika

Autor: Ján Krivý

Vymedzenie obsahu a dedikácia metodiky

Tento dokument opisuje postupy pri vytváraní a uzatváraní issue v systéme pre správu úloh v rámci projektu BIGDATA. Ako systém pre správu úloh bola zvolená Jira.

Týmto dokumentom sa riadia členovia tímu č. 15 – Nitrogen family. Dané postupy sú závezné pre všetkých členov tímu počas trvania celého projektu v rámci predmetu Tímový projekt I,II v akademickom roku 2014/2015.

Pre jednoznačnosť bude tento dokument obsahovať nepreložené pojmy, ktoré sa v rovnakom znení objavujú aj v systéme správy úloh Jira.

Zoznam nadväzujúcich metodík a dokumentov

- Správa verzií (Vladimír Demčák)
- Priebeh šprintu (Martin Šustek)
- Komentovanie zdrojového kódu a jeho dokumentovanie pomocou JavaDoc (Martin Cibula)
- Zoznam kompetencií a pôsobnosti členov tímu (Martin Cibula)

Predpoklady a požiadavky

- Prístup do projektu BIGDATA v Jire
- Prístup do repozitára BigDataFiit v bitbuckete, kde sa nachádza tímová stránka Wiki
- Znalosť workflowu issue v systéme pre správu úloh Jira

Vymedzenie pojmov a skratiek

- Issue – úloha v Jire
- Tímová Wiki – systém pre uchovávanie dokumentácie k projektu
  - https://bitbucket.org/BigDataFiit/bigdata/wiki/browse/
- Jira – systém správy úloh
  - o https://jira.fiit.stuba.sk/browse/BIGDATA
- JavaDoc – dokumentácia vo formáte HTML zo zdrojového kódu

Súhrn rol a zodpovednosti účastníkov

- **Vedúci tímu**
  - Je zodpovedný za zadávanie hlavných issue (new feature alebo tasku) šprintu ľuďom podľa oblasti pôsobnosti
  - Uzatvára issue new feature na konci šprintu
- **Manažér dokumentácie**
  - Zodpovedný za kontrolu zdokumentovania každého issue
  - Zodpovedný za rovnomerné rozdelenie issue pre členov tímu
- **Asignee**
  - Momentálny vykonávateľ issue
  - Zodpovedný za vykonanie issue podľa požiadaviek vyplývajúcich zo zadania
- **Reporter**
  - Zadávateľ issue
  - Uzatvára issue, ktoré zadával po splnení všetkých náležitostí
- **Manažér plánovania**
  - Zabezpečuje správne priradenie issue k členovi tímu ak je to nejednoznačné

Opisy postupov

1. **Typy issue**

   New feature

   - Pridanie nejakého celku funkcionality v danom šprinte
   - Task vytvára vedúci tímu po stretnutí na začiatku šprintu
   - Každá new feature obsahuje subtasky na základe ktorých sa dospeje k požadovanému výsledku

   Task

   - Stanovená úloha pre daný šprint
   - Task vytvára vedúci tímu na začiatku šprintu
   - Task môže obsahovať subtasky, na základe ktorých sa dospeje k požadovanému výsledku

   Subtask

   - Elementárna úloha
   - Subtasky vytvára člen tímu, ktorý je ako asignee hlavného issue – teda Tasku alebo New feature

   Bug

   - Issue, ktorý vytvára slúži na vyriešenie nájdeného defektu
   - Bug vytvára hociktorý člen tímu počas šprintu

2. **Vytvorenie issue**

   Vybranie typu issue

   - Každý issue musí mať presne stanovený typ podľa predošlého rozdelenia

   Komu issue priradíme - asignee

   - Jednotlivé issue priraďujeme podľa oblastí posobnosti jednotlivých členov tímu (viac v zozname kompetencií tímu), pokiaľ to nevyplýva inak zo stretnutia tímu alebo neurčí manažér plánovania inak
   - Ak nieje jednoznačne jasné komu issue prideliť, pridelí sa manažérovi plánovania, ktorý rozhodne komu issue patrí

   Pomenovanie issue - summary

   - Pomenovanie issue musí byť stručné, jasné a zrozumiteľné pre každého člena tímu

- Názov issue je vo formáte sloveso + podstatné meno
  - sloveso je v tvare slovesného podstatného mena (napr. vytvorenie alebo aktualizovanie)
  - podstatné meno vystihuje podstatu problému, pre ktorý sa daný issue vytvára

Stanovenie priority

- Blocker
  - priorita blocker sa nastaví ak výsledok tohto issue je vstupom nejakého ďalšieho issue a je ho teda nutne spraviť čo najskôr, keďže blokuje ďalšie issue
- Critical
  - priorita critical sa nastaví, ak je nutné tento issue dokončiť tento šprint a v čo najkratšom čase
- Minor
  - priorita minor sa nastaví, ak je potrebné tento issue dokončiť v danom šprinte, nie je však podstatné kedy

Stanovenie dátumu – due date

- Ak nevyplýva zo stretnutia inak, due date sa stanoví na koniec šprintu

Popis issue – description

- Každý issue musí obsahovať popis
- Z popisu pre issue musí byť jasné:
  - čo je úlohou člena tímu, ktorý to rieši
  - aký ma byť výstup

Časový odhad – estimate time

- Časový odhad pre issue vyplýva z úvodného stretnutia tímu a plánovania šprintu

3. **Ukončenie issue**

- Ukončenie issue sa skladá z dvoch častí:
  - vyriešenie – resolve
  - ukončenie - close

Vyriešenie issue – resolve

- Musia byť splnené tieto podmienky

  - Splnenie požiadaviek vyplývajúcich zo zadania
  - Otestovanie novej implementácie
  - Zdokumentovanie riešenia (JavaDoc, Wiki)
- Ak sú splnené všetky podmienky je potrebné spraviť resolve issue a ten musí obsahovať:
  - reálny strávený čas pri riešení
  - odkazy na dokumentáciu vo Wiki
  - krátky a stručný popis riešenia a jeho výstupu
  - Asignee je zadávateľ issue

Kontrola zadávateľom – reporter

- Je nutné skontrolovať, či je naozaj issue vyriešený podľa stanovených požiadaviek
- Priradí issue na kontrolu manažérovi dokumentácie aby skontroloval zdokumentovanie daného problému
- Ak sú splnené požiadavky a problém bol dostatočne zdokumentovaný môže sa uzavrieť issue jeho zadávateľom
- Výnimku tvoria len issue typu New feature, ktoré uzatvára vedúci tímu na tímovom stretnutí
- Ak nebola splnená nejaká z požiadaviek issue sa znova otvorí:
  - Priradí sa členovi tímu, ktorý nesplnil požiadavku
  - Musí obsahovať jasný popis:
    * Aká požiadavka nebola splnená
    * Prečo nebola splnená
    * Čo je ešte nutné spraviť aby bola splnená

Kapitola 5

Písanie a identita tímových dokumentov

Autor: Juraj Michalička

O metodike

Rozsah tejto metodiky

Tento dokument obsahuje postupy pri tvorbe textových a prezentačných tímových dokumentov v rámci Tímového projektu 1 a 2.

Kto sa riadi touto metodikou

Riadi sa ňou každý člen 15. tímu, Nitrogen family, Tímového projektu 1 a 2 v akademickom roku 2014/2015 počas celého jeho trvania.

Nadväzujúce metodiky

- Metodika - písanie dokumentácie
- Šablóna zápisnica
- Šablóna retrospektíva

Požiadavky

- Prístup do tímového priečinku „Team15“ na Dropbox
- Prístup do administrátorskej časti webstránky Tímového projektu „http://team15-14.ucebne.fiit.stuba.sk/“

Roly a ich zodpovednosť za tvorbu dokumentov

- **Zapisovateľ**
  Je zvolený v rámci tímového stretnutia vedúcim tímu, alebo jeho zástupcom v prípade neprítomnosti. Je zodpovedný za vytvorenie a zdieľanie zápisnice zo stretnutia
- **Dokumentarista**
  Je to ktorýkoľvek člen tímu, ktorý vytvorí, alebo naštuduje novú časť v rámci tímového projektu. Je zodpovedný za zdokumentovanie tejto časti
- **Manažér kvality a dokumentácie**
  Je zodpovedný za kontrolu dokumentácie

Pojmy a skratky

- **Dropbox** – služba na zdieľanie súborov medzi viacerými počítačmi
- **Verejné dokumenty** – dokumenty, ktoré majú byť dostupné komukoľvek, nie len členom tímu
- **Textový dokument** – dokumenty takého typu, na ktorých písanie sa bežne používa MS word, alebo LaTex

Metodika

Titulná strana textových dokumentov
Titulná strana všetkých tímových textových dokumentov musí obsahovať:

- Názov dokumentu
- Názov a adresa školy
- Logo tímu
- Meno vedúceho tímového projektu
- Zoznam členov tímu
- Zoznam autorov dokumentu
- Dátum vytvorenia dokumentu

Vzor titulnej strany je v prílohe č. 1.

Písanie dokumentácie
Písaniu dokumentácie sa venuje Metodika – písanie dokumentácie.

Písanie zápisnice Na písanie zápisnice sa používa šablóna, ktorá je umiestnená na dropbox v priečinku „Team15\zapisnice\“.
Vyplnia sa údaje:

- Téma stretnutia
- Dátum stretnutia
- Miesto stretnutia
- Meno vedúceho stretnutia
- Meno zapisovateľa
- Zoznam prítomných a neprítomných
- Obsah stretnutia
- Tabuľka novovzniknutých úloh
- Tabuľka dokončených úloh

Následne sa dokument zdieľa s tímom a aj ako verejný dokument.

Písanie retrospektívy

Na písanie retrospektívy sa používa šablóna, ktorá je umiestnená na dropbox v priečinku Team15\retrospektiva\.
Vyplnia sa údaje:

- Číslo šprintu v názve v dokumente
- Autori dokumentu
- Dátum vytvorenia dokumentu
- Pozitíva, nedostatky a prevencia nedostatkov produktu
- Pozitíva, nedostatky a prevencia nedostatkov procesu
- Tabuľka hodnotenia úloh

Následne sa dokument zdieľa s tímom.

Písanie ostatných textových dokumentov

Ostatné textové dokumenty sa píšu po konzultácii s ďalšími členmi tímu, alebo vedúcim tímového projektu.

Titulná strana prezentácií

Na titulnej strane(slajde) prezentácie musí byť:

- Názov prezentácie
- Logo tímu
- Názov tímu

Tímové video

Na začiatku tímového videa musí byť:

- Názov tímu
- Tímové logo

Na konci tímového videa musí byť:

- Meno vedúceho tímového projektu
- Zoznam členov tímu
- Zoznam úloh vo videu

Zdieľanie tímových dokumentov

Všetky tímové dokumenty treba umiestniť do tímového priečinka „Team15\“ na dropbox, respektíve do vhodného podpriečinku. Pokiaľ nie je explicitne jasné, do ktorého podpriečinka patrí daný dokument, ani to nevyplíva z tímovej konzultácie, treba vhodný podpriečinok vytvoriť. Textové dokumenty treba zdieľať vo formáte „.doc“. Tabuľky treba zdieľať vo formáte „.xls“. Prezentácie sa zdieľajú v pôvodno formáte, kvôli zachovaniu efektov a dizajnu. Taktiež treba uložiť ich kópi vo formáte „.pdf“. Pokiaľ ide o verejný dokument treba sa prihlásiť do administrátorskej časti webstránky tímového projektu na adrese „http://team15-14.ucebne.fiit.stuba.sk/wp-admin/“. Následne v časti „Media „ treba nahrať daný dokument. Pokiaľ ide o textový dokument, musí byť vo formáte „.pdf“. Nakoniec treba hyperlink nahraného dokumentu pridať na stránku „Dokumenty“ vo formáte „dátum pridania – hyperlink s krátkym popisom“.

Vzor titulnej strany textového dokumentu

Názov dokumentu

Fakulta informatiky a informačných technológií STU v Bratislave, Ilkovičova 2, 842 16 Bratislava 4, Slovenská republika

Tímový projekt 2014/2015: Big Data

Vedúca tímového projektu: Ing. Nadežda Andrejčíková, PhD.
Členovia tímu: Bc. Marek Černák, Bc. Marin Cibula, Bc. Vladimír Demčák, Bc. Ján Krivý, Bc. Jozef Melko, Bc. Juraj Michalička, Bc. Marin Šustek

Autori dokumentu:
Dátum vytvorenia dokumentu:

Kapitola 6

Komentovanie zdrojového kódu a jeho dokumentovanie pomocou JavaDoc

Autor: Martin Cibula

Úvod

Cieľom tejto metodiky je stanoviť postupy komentovania zdrojového kódu v programovacom jazyku Java. Zaoberá sa dvoma spôsobmi komentovania zdrojového kódu a to komentovaním častí kódu klasickými komentármi priamo v kóde alebo komentovanie za účelom následného generovania dokumentácie nástrojom JavaDoc. Metodika má slúžiť pre programátorov, ktorý sú povinný sa ňou riadiť. Či je kód komentovaný podľa postupov daných v tomto dokumente má za úlohu dohliadať manažér kvality a dokumentácie.

Pojmy a skratky

JavaDoc - je generátor dokumentácie vo formáte HTML zo zdrojového kódu v jazyku JAVA

Súvisiace metodiky

- metodika inštalácie a konfigurácie JavaDoc vo vývojovom prostredí Eclipse
- metodika písania zdrojového kódu

Metodika komentovania zdrojových kódov

Správne a dostatočne okomentovaný kód je nevyhnutný pre pochopenie zdrojového kódu a pre jeho prípadné úpravy a to v čo najkratšom čase. V nasledujúcich častiach

sú popísané postupy, ktoré musíte dodržať pri komentovaní častí kódu komentármi priamo v kóde.

Zásady komentovania zdrojového kódu

Potreba komentovanie kódu závisí od výstižnosti názvov premenných a metód pri písaní kódu, čím sa zaoberá metodika písania kódu. Ak je z kódu jasné čo, ktorá časť kódu vykonáva, je potreba komentárov úplne minimálna, a čím je menej komentárov v kóde, tým je prehľadnejší a lepšie sa s ním pracuje.
Pri písaní komentárov musia byť dodržané nasledovné zásady:

- Píšte komentáre v slovenskom jazyku.
- Komentujte hneď počas písania kódu.
- Komentár píšte vždy nad časť kódu ku ktorej sa viaže.
- Komentár píšte jednoducho a jednoznačne.
- Komentár píšte vždy na prázdny riadok, nikdy na riadok s častou kódu.
- Komentár nepopisuje celú funkcionalitu časti kódu, iba doplňuje informácie, ktoré nie sú jasné priamo z kódu.

V jave sú dve možnosti písania komentárov vzhľadom na rozsah komentára:

- Jednoriadkové - označené na začiatku riadka znakmi //
- Blokové - začínajú znakmi /* a končia */

Všade, kde sa vyskytuje viacej riadkový komentár použite blokové komentovanie. Začiatok blokového komentára je vždy riadok nad prvým a koniec riadok pod posledným riadkom komentára. Príklad blokového komentára:

```
/*
Toto je komentár na
viacej riadkov.
*/
```

Dokumentácia zdrojových kódov v JavaDoc

JavaDoc bol vytvorený na generovanie dokumentácie na základe komentátor elementov, ako balíky, triedy alebo metódy, v jazyku JAVA. Pri dodržaní pravidiel komentovania kódu definovaných v JavaDoc je zaručený kvalita výstupnej vygenerovanej dokumentácie. Výstupom JavaDocu je dokumentácia vo formáte HTML.

Vytváranie komentárov

Komentáre pre JavaDoc majú stanovenú formu. Píšu sa vždy pred komentovaným elementom a skladajú sa z nasledujúcich troch častí, pričom každá z týchto častí je umiestnená na nový riadok.

- Začiatočné znaky komentára - prvý riadok komentára označený znakmi /**, v tomto riadku nič ďalšie nepíšte
- Telo komentára - každý riadok tela sa začína znakom *, za týmto znakom je obsah komentára
- Koniec komentára - posledný, ukončovací riadok komentára označený znakmi */, tak isto ako pri začiatku komentára tu nič nepíšte

Ukážka komentára pre JavaDoc:

```
/*
* Obsah komentára
*
* @značka - popis
* @značka - popis
*
*/
```

Poradie zančiek v komentáry je dané a treba ho dodržiavať aby boli komentáre konzistentné. Poradie hlavných značiek aj s potrebnými atribútmi je nasledovné:

- @author [celé meno autora]
- @version [číslo verzie v ktorej bol element vytvorený]
- @param [názov parametra] [popis parametra]
- @return [typ] [popis návratovej hodnoty]
- @throws [typ výnimky] [popis výnimky]

Medzi značkou a atribútmi sa pomlčky nepíšu, jedine pri značke @return medzi typom a popisom. Ak je počet rovnakých značiek väčší ako jedna, tieto značky musia byť zoradené bezprostredne za sebou.

Komentovanie balíkov

Pri balíkoch zdrojových súborov popíšte iba jednoducho aké zdrojové kódy sú združené v danom balíku. V prípade, že z názvu balíka to je jednoznačné, tak netreba daný balík komentovať. Pri komentovaní balíka nepoužívajte žiadne značky.

Komentovanie tried

Každú trieda riadne okomentujte, pričom komentár musí obsahovať popis, autorov danej triedy a verziu v ktorej bola táto trieda vytvorená. Popis triedy píšte formou oznamovacích viet. Výsledný komentár pre triedu s dvoma autormi bude vyzerať nasledovne:

```
/**
* Výstižný popis triedy aby z neho bolo jasné načo komentovaná trieda slúži
*
* @author [celé meno 1. autora]
* @author [celé meno 2. autora]
* @version [číslo verzie softvéru]
*/
```

Komentovanie metód

Podobne ako každá trieda tak aj každú metódu riadne okomentujte. Komentár metódy pre JavaDoc musí obsahovať nasledovné:

- popis - výstižný popis metódy formulovaný do oznamovacích viet
- @author - celé mená všetkých autorov danej metódy
- @version - číslo verzie v ktorej bola metóda vytvorená
- @param - ak metóda má vstupné parametre, musia byť všetky uvedené a popísané
- @return - návratová hodnota metódy
- @throws - všetky výnimky, ktoré môžu daná metóda vyhodiť, pričom z popisu výnimky musí byť jasné kedy môžu nastať

Okrem týchto povinných značiek môžete použité aj ostatné značky podporované Javadoc-om ak je to dôležité. Komentár metódy s jedným autorom, dvoma vstupnými parametrami a jednou výnimkou bude vyzerať nasledovne:

```
/**
* Popis metódy.
*
* @author [celé meno jediného autora]
* @version [číslo verzie softvéru]
* @param [názov 1.parameter][popis parametra]
* @param [názov 2.parameter][popis parametra]
* @return [typ] - [popis návratovej hodnoty]
* @throws [názov výnimky][popis výnimky]
*/
```

Generovanie JavaDoc v Eclipse

V prostredí Ecslipse, kde je už rozbehaný JavaDoc, vygeneruje manažér dokumentácie dokumentáciu, vždy po zlúčení zdrojových kódov na hlavnú, master, vetvu, podľa nasledujúceho postupu:

1. Eclipse→Project→Generate JavaDoc
2. V políčku "JavaDoc command:"vyberte cestu k javadoc.exe, zvyčajne to je cesta: $C : /ProgramFiles/Java/jdk1.7.0_15/bin/javadoc.exe$ kde sa môže meniť verzia JDK
3. Vyberte projekt z ktorého chcete generovať dokumentáciu
4. Nastavte cestu kde bude dokumentácia uložená
5. Stačte tlačidlo finish

Výsledná dokumentácia je uložená vo vybranom priečinku vo forme HTML súborov, pričom index.html je hlavný, koreňový súbor. Následne vloží dokumentáciu do úložiska s dokumentmi.

Kapitola 7

Štábna kultúra písania kódu v jazyku Java - Metodika

Autor: Marek Černák

Obsah dokumentu

Tento dokument obsahuje spísané pravidlá pre správne písanie zdrojového kódu stanovené pre projekt BigData, ktorý vyvíja tím č. 15 (Nitrogen Family) v rámci predmetu Tímový projekt.

Platnosť

Metodika je platná a záväzná pre každého člena tímu, ktorý zasahuje do kódu. Úlohy členov tímu, ktorých sa táto metodika týka:

- Programovanie: Každý člen tímu, ktorý prispieva do kódu novou funkcionalitou sa riadi touto metodikou.
- Testovanie: Takisto pri vytváraní testov pomocou pomocných knižníc je nutné túto metodiku dodržiavať.
- Refactoring: Pri prestavbe štruktúry tried si musí člen tímu dávať pozor, aby ostali pravidlá písania kódu dodržané.
- Generovanie JavaDoc: Pred vygenerovaním dokumentácie komentárov je požadovaná kontrola kódu, aby dokumentácia obsahovala okomentované iba správne zapísané časti kódu.

Členovia tímu

- Martin Šustek
- Martin Cibula
- Marek Černák
- Vladimír Demčák
- Jozef Melko
- Juraj Michalička
- Ján krivý

Nadväzujúce metodiky

S metodikou štábnej kultúry písania kódu v jazyku Java úzko súvisí metodika „Komentovanie zdrojového kódu a jeho dokumentovanie pomocou JavaDoc“ od Martina Cibulu, ktorá sa zaoberá písaním kódov a a generovaním JavaDoc dokumentácie.

Slovník pojmov

- refactoring - optimalizačný proces
- JavaDoc - nástroj na dokumentovanie komentárov v jazyku Java
- kód - zdrojový kód vyvíjaného produktu
- kučeravá zátvorka - znaky „“ (otvárajúca) a „“ (uzatvárajúca)

Úvod

Existencia konvencií písania kódu je veľmi dôležitá. Dôvody pre dodržiavanie týchto konvencií sú rôzne:

- softvér zvyčajne nie je udržiavaný až do konca jeho existencie pôvodným autorom
- konvencie písania kódu zvyšujú čitateľnosť kódu, čo umožňuje programátorom pochopiť nové časti kódu oveľa lepšie a rýchlejšie
- väčšia časť života softvéru je strávená jeho údržbou

Nižšie rozpísané pravidlá prispievajú k efektívnejšej práci a môžu viesť k lepšej funkcionalite výsledného produktu.

Písanie názvov

Pri tvorbe názvov je dôležité dodržiavať rozdiely, ktoré programátorom zabezpečia lepšiu čitateľnosť. Aby sa predišlo zbytočnému chaosu, pri tvorbe názvov sa musí používať výhradne anglický jazyk.

Názvy premenných a metód

Názvy premenných a metód sa píšu vždy so začiatočným malým písmenom.

Listing 7.1: Názvy premenných a metód 1

```
public String name;
public void write() {...};
```

Ak je premenná tvorená z viacerých názvov, každé ďalšie slovo napájame na predošlé so začiatočným veľkým písmenom.

Listing 7.2: Názvy premenných a metód 2

```
public String sureName;
public void writeMore() {...};
```

Názvy premenných nemôžu byť viacvýznamové. Metódy nemôžu obsahovať názvy tried, pretože by malo byť z kódu jasné, ktorému objektu patrí volaná metóda.

Listing 7.3: Názvy premenných a metód 3

```
object.getLength(); //spravne: object.getObjectLength
```

Metódy musia byť pomenované podľa toho, čo vykonávajú, prípadne čo vracajú. Pre dvojice metód opisujúce opačnú funkcionalitu musí programátor používať opozitá (napr. get/set, add/remove).

Názvy tried

Názvy tried sa začínajú vždy s veľkým začiatočným písmenom.

Listing 7.4: Názvy tried 1

```
public class Mountain {...}
```

Ak sa názov triedy skladá z viacerých slov, každé ďalšie slovo napájame na predošlé so začiatočným veľkým písmenom.

Listing 7.5: Názvy tried 2

```
public class MountainBike extends Bicycle {...}
```

Výnimky musia v názve obsahovať príponu „Exception“.

Názvy objektov

Pri pomenovaní nového objektu sa používa konvencia ako pri deklarácii premennej.

Pri písaní skratiek nepoužívame veľké písmená, postupujeme podľa predchádzajúcich konvencií, a teda veľké zostáva aj v tomto prípade iba začiatočné písmeno.

Listing 7.6: Názvy objektov 1

```
public String shortHtmlName; //spravne: public String shortHTMLName;
```

Názvy balíkov

Názov každého balíka začína prefixom „sk.fiit.team15.*“. Nasledovný reťazec v názve (namiesto znaku *) vyplýva z vlastností tried umiestnených v balíku.

Názvy konštánt

Konštanty musia byť písané s veľkým písmom. Ak sa skladajú z viacerých slov, jednotlivé slová oddeľujeme podčiarkovník.

Listing 7.7: Názvy konštánt 1

```
public static final int MAX_SECONDS = 25;
```

Metódy a premenné typu boolean (logické)

Pre boolean metódy a premenné požívame prefix „is“.

Kolekcie a polia

Pre pomenovanie kolekcií a polí používame množné čísla. Premenné používané pre značenie iterácií v slučkách označujeme od písmena i vyššie podľa abecedného poradia.

Súbory

Názov súboru sa musí zhodovať s názvom triedy, rozhrania alebo výnimky, ktorá je v ňom implementovaná a musí sa začínať veľkým písmenom.

Všeobecné pravidlá

Názvy nesmú obsahovať skrátené tvary. Skratky bývajú často mätúce a programátori, ktorí kód prevezmú im nemusia rozumieť.

Listing 7.8: Všeobecné pravidlá 1

```
computeAverage(); //spravne: compAvg();
```

Formát kódu

Pri písaní kódu si programátor musí dávať pozor na formátovanie textu v textovom editore. Pre tímový projekt používame všetci rovnaké formátovanie v editoroch (veľkosť tabulátora, dĺžka riadku) aby nenastali problémy vo formáte pri importovaní kódu.

Riadok

Dĺžka riadku nesmie presiahnuť 80 znakov. Práca s kódom je nepraktická, keď sa programátor musí presúvať pomocou horizontálneho scrollbaru. Takéto kroky sú nadbytočné a spomaľujú prácu. Riadok musí byť ukončený znakom kučeravej zátvorky, bodkočiarky, čiarky alebo jedným z operátorov.

Listing 7.9: Riadok 1

```
private void Example (int example1, int example2 , int example3, //
                                    int example4) { // Koniec deklaracie

int sumOfExamples = example1 + example2 + example3 + //
                                           example4; // Koniec inicializacie

...

} // Koniec riadku
```

Medzery

Pre ďalšie zvýšenie prehľadnosti píšeme medzery vždy, keď je to možné.

Listing 7.10: Medzery 1

```
a+=b+(c*d)*e;
```

Na príklade Medzery 1 zápisu bez použitia medzier vidno, ako sa pomerne jednoduchá časť kódu môže stať neprehľadnou. V skutočnosti sa môže jednať o oveľa komplikovanejšie zápisy.

Listing 7.11: Medzery 2

```
a += b + (c * d) * e;
```

Importy

Importy musia byť udržiavané podľa aktuálnej potreby. Nepotrebné balíky odstraňujeme. Je zakázané importovať všetky triedy balíka, ak z nich používame iba niektoré.

Listing 7.12: Importy 1

```
import java.util.List; // zle pouzitie: import java.util.*;
```

```
import java.util.ArrayList;
import java.util.HashSet;
```

Poradie definície triedy alebo rozhrania

1. deklarácia triedy alebo rozhrania
2. deklarácia/definícia premenných (v prípade rozhrania len konštanty)
3. deklarácia/definícia metód
4. definícia konštruktorov (v prípade triedy)

Písanie kučeravej zátvorky

Pri písaní otváracej kučeravej zátvorky využívať koniec prvého riadku bloku kódu (pozri Správne použitie kučeravej zátvorky 3) namiesto začiatku nasledujúceho riadku.

Listing 7.13: Písanie kučeravej zátvorky ("nesprávne" použitie)

```
for (i = 0; i < 5; i++)
{
        System.out.println(i);
}
```

V príklade Písanie kučeravej zátvorky ("nesprávne" použitie) dje v zátvorkách schválne použitý výraz „nesprávne". Tento zápis vo všeobecnosti nesprávny nie je, no v rámci nášho projektu ho za nesprávny budeme považovať. Dôvodom je dosiahnutie písania kódu jednotným spôsobom, v dôsledku čoho sa zvýši jeho čitateľnosť.

Predchádzanie chybám

Správne použitie kučeravej zátvorky

Kučeravú zátvorku používame v každom prípade, aj tam, kde nie je nevyhnutná, aby sme predišli prípadným chybám z nepozornosti.

Listing 7.14: Správne použitie kučeravej zátvorky (nesprávne použitie)

```
for (i = 0; i < 5; i++)
        System.out.println(i);
```

V príklade Správne použitie kučeravej zátvorky (nesprávne použitie) vidíme použitie slučky bez kučeravých zátvoriek. Pri doplňovaní kódu by sa mohlo ľahko stať, že ich programátor zabudne doplniť, čo môže vyústiť do neočakávaného správania programu.

Listing 7.15: Správne použitie kučeravej zátvorky (nesprávne použitie – neočakávaná funkcionalita)

```
for (i = 0; i < 5; i++)
        System.out.println(i);
        a += i; // nastane neocakavana funkcionalita
```

Listing 7.16: Správne použitie kučeravej zátvorky (správne použitie)

```
for (i = 0; i < 5; i++){
        System.out.println(i);
}
```

Porovnávanie

Pri porovnávaní premennej s hodnotou sa riadime pravidlom ľavej ruky.

Listing 7.17: Porovnávanie 1

```
if (a == 10) {...} // Pravidlo pravej ruky
if (10 == a) {...} // Pravidlo lavej ruky
```

Pravidlo ľavej ruky musíme používať, aby sa predchádzalo prípadným chybám spočívajúcim v neúmyselných priradeniach namiesto porovnaní. Takéto chyby sa ťažšie hľadajú, keďže prekladač v takomto prípade chybu nenahlási.

Listing 7.18: Porovnávanie 2

```
if (a = 10) {...} // Prekladac nehlasi chybu
if (10 = a) {...} // Prekladac hlasi chybu
```

Typová konverzia

Konverzia typov musí byť explicitne vyznačená. Dodržaním tohto pravidla programátor indikuje, že je konverzia úmyselná.

Listing 7.19: Typová konverzia

```
floatValue = (int) intValue; // zle pouzitie: floatValue = intValue;
```

Komentáre

Po každej časti kódu musí programátor napísať komentár, aby bolo úplne jasné načo daný celok kódu slúži. Pre bližší popis ku komentárom, prosím, pozrite metodiku s názvom „Komentovanie zdrojového kódu a jeho dokumentovanie pomocou JavaDoc“ od autora Martina Cibulu.

Kapitola 8

Správa verzií - Metodika

Autor: Vladimír Demčák

Dokument opisuje postupy pri správe verzií v projekte BigData, ktorý vyvíja tím č. 15 – Nitrogen Family (ďalej NGF).

Na verziovanie projektu BigData používa tím NGF distribuovaný systém na správu verzií - GIT. Každý člen tímu NGF je povinný vytvoriť si lokálny repozitár a naklonovať remote repozitár.

Pred prácou a vykonávaním zmien a commitovaním, by si mal každý člen uvedomiť, že zmeny, ktoré vykonáva sú zverejnené pre ostatných členov tímu, a preto je potrebné písať kód zrozumiteľne a kultivovane (podrobnejšie v metodikách Štábna kultúra písania kódu v jazykoch Java, Komentovanie zdrojového kódu a jeho dokumentovanie pomocou JavaDoc). Pre každú novú verziu je potrebné aktualizovať dokumentáciu.

Verziovanie je úzko späté s nástrojom na riadenie požiadaviek JIRA2,ktorý NGF pri práci na projekte aktívne používa.
Slovník pojmov a skratiek:

- remote repozitár – vzdialený repozitár
- commit – odovzdanie zmien
- task a subtask – úloha zadaná pomocou nástroja JIRA
- refacotring – optimalizačný proces
- review zmien – prehodnotenie a preskúmanie vykonaných zmien
- issue – úloha v projekte (vylepšenia, oprava chýb)
- NGF – Nitrogen Family tím
- GIT – distribuovaný systém na správu verzií

Verziovanie

Roly

Pri používaní GIT sú definované dve základné roly: Repository owner a contributor.
Repository owner(Správca remote repozitára) je zodpovedný za:

- remote repozitár
- vytvorenie remote repozitára
- pridávanie vývojárov
- pravidelná údržba
- riešenie problémových commitov
- riešenie problémových spájaní vetiev
- produkčnú vetvu master

Contributor (Vývojár) je zodpovedný za:

- vlastný lokálny repozitár
- naklonovanie remote repozitára
- vytváranie vetiev „feature“
- push a spájanie vetiev
- push a spájanie vetiev do remote repozitára
- formát a obsah správ
- zrozumiteľný zdrojový kód a komentáre
- riešenie jednoduchých konfliktov

Všetci členovia tímu majú pridelené práva role vývojár. Vlastník remote repozitára vystupuje v role „správca“.

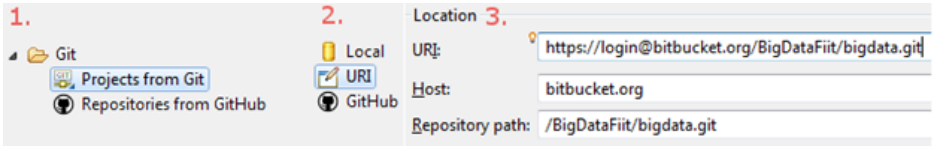
Požiadavky a predpoklady používania GIT

- Vytvorený účet na BitBucket a prístup do repozitára BigDataFiit
- Lokálny repozitár GIT a nainštalovaný server Tomcat v7 pre lokálne testovanie a vývoj verzií (najmä vývojová verzia)
- IDE pre vývoj aplikácií v jazyku JAVA a GUI pre GIT– odporúčaný Eclipse

Pre prístup k repozitáru BigDataFiit je potrebné kontaktovať správcu repozitára. Je potrebné naklonovať remote repozitár do lokálneho.

Najjednoduchší a odporúčaný postup importu verziovaného projektu z GIT do Eclipse je nasledovný:

1. File – Import
2. Vyber Projects from Git v časti „Git“
3. Vyber a vyplň URI



Obr. 8.1: Import projektu z remote git repozitára do IDE Eclipse[1]

Aktualizácia repozitára

Člen tímu je povinný pracovať s aktuálnou verziou. Preto je potrebné vždy pred prácou na projekte aktualizovať remote repozitár.

Pri aktualizácií lokálneho repozitára z remote, je možné pristupovať dvoma spôsobmi:

Fetch:

- nezlučovacia aktualizácia repozitára
- použitie v prípade ak je potrebné stiahnuť zmeny z remote repozitára a zároveň zachovať zmeny v lokálnom repozitári.
- niektorí vývojári používajú automatizovaný fetch pomocou „cron“ alebo iných plánovacích systémov. Tento spôsob sa neodporúča.
- fetch je možné vykonať pomocou Eclipse pod „Team – remote – Fetch From“

Pull:

- vykonáva zlučovaciu aktualizáciu repozitára
- pri vykonávaní pull je potrebné vybrať vetvu z remote repozitára a aktuálnu vetvu v lokálnom repozitári.

Bežný pracovný postup

1. aktualizácia kópie z remote repozitára
2. vykonanie zmien v súboroch a obsahu projektu
3. review zmien, ktoré boli vykonané
4. commit zmien, prípadne push (commit –m „COMMIT správa“)

Commitovanie

Pred commitovaním zmien:

- musí byť projekt otestovaný na funkcionalitu, ktorá mohla byť ovplyvnená vykonanou zmenou. Veľmi dôležité je skontrolovať, či je projekt skompilovateľný.
- ktoré ovplyvňujú build projektu(zmena classpath/pridanie knižnice), je potrebné po commite informovať ostatných členov tímu NGF.
- je potrené ubezpečiť sa, že sa commitujú len súbory, ktoré boli modifikované.
- dostatočne okomentovať zdrojový kód (javadoc, komentáre v zdrojovom kóde)
- vykonať revíziu zdrojového kódu a dodržať štandardy písania kódu v JAVE
- vytvoriť vhodnú commit správu a dodržať štandardnú formu commit správy.

Forma commit správ

Všetky commit správy musia jednoznačne identifikovť o akú úlohu, respektíve issue, vytvorenú v nástroji JIRA ide.
Štandardný tvar commit správy je nasledovný:

$$IDISSUE–NazovISSUE–krtkyopiszmien, vylepen, fix, update$$

Príklad:

$$BIGDATA–6–VytvaraniePOSTmetody–upravenyJsonPost, spracovaniesearchrequest, update$$

Základným jazykom pri vytváraní commit správ je slovenčina. Dĺžka správy by nemala byť väčšia ako 150 znakov (ID a názov issue sa do dĺžky správy nezapočítava).

Odstraňovanie súborov

Súbor so zdrojovým kódom je možné odstrániť až po overení, že je skutočne nepotrebný.

Konfiguračné súbory je možné odstrániť až po dohode s team leadrom vývoja.

Riešenie konfliktov

V prípade riešenia konfliktov je vhodné použiť GUI nástroj na porovnávanie obsahov súborov, ktoré sú v konflikte. Je potrebné použiť minimálne nástroj „GIT GUI“. Neodporúča sa porovnávanie v konzole. V prípade použitia vývojárskeho prostredia Eclipse je možné použiť možnosť – „Compare With“.

Ak nie je možné jednoducho vyriešiť konflikt, je potrebné dohodnúť konzultáciu s členom tímu, ktorý vykonal zmeny, pred aktuálnymi zmenami, ktoré sú v konflikte. Prípadne konzultovať so správcom remote repozitára.

Vetvenie a verzie

Vývoj projektu BigData prebieha iteratívnym a inkrementálnym spôsobom. Vývoj prebieha na viacerých vetvách a projekt v určitých časových intervaloch vychádza ako nová stabilná verzia (produkčná).

Vetvy vývoja

Projekt je vytváraný v dvoch hlavných vetvách. Cieľom vytvárania vetiev je oddeliť vývojovú verziu systému od stabilnej verzie, pričom je potrebné zohľadniť aj možný vznik chýb v stabilnej verzii.

Hlavnými vetvami vývoja projektu sú:

Master:

Produkčná vetva vývoja. Táto vetva obsahuje verziu, ktorá je stabilná, otestovaná a aktuálne nasadená na serveri.

Devel:

Vetva devel je vývojová vetva. Všetky zmeny sú pri vývoji commitované na túto vetvu. Tím NGF si určí termín meetingu, kde sa analyzuje vývojová verzia. Vykonajú sa nové, ale aj reverzné testy a ak je testovanie úspešné vetva devel sa spojí na master vetvu.

Ďalšími podpornými vetvami sú:

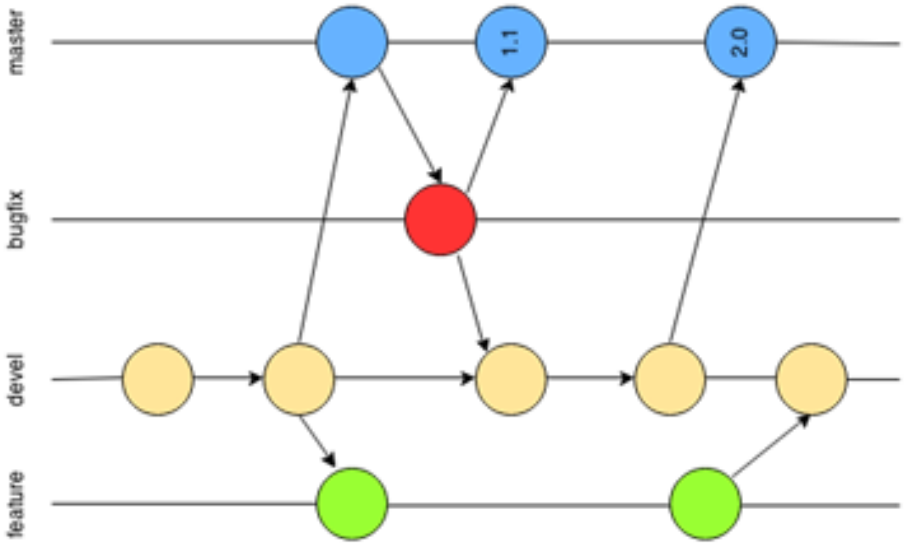
Fixbug:

Vetva, na ktorej sa opravujú chyby, ktoré boli zistené v nasadenej produkčnej verzii a je potrebné takéto chyby v čo najkratšom čase opraviť.

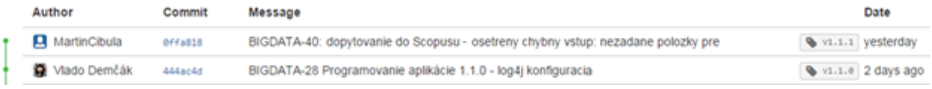
Feture-xyz:

Vetva na ktorej prebiehajú úpravy alebo iné pridávané funkcionality. Pre každú novú funkcionalitu je potrebná nová vetva, ktorá je vhodne pomenovaná v tvare „feature-uloha“ (napr. feature-refactoring).

Príkaz GIT tag sa v rámci projektu BigData používa pri vydaní novej stabilnej verzie alebo pri oprave chýb. Vývojové verzie sa tagovať môžu, ale názov tagu sa nesmie začínať písmenom „v“. Formát tagov verzií je: vX.X.X, pričom je potrebné stručne uviesť pre každý tag stručný opis.



Obr. 8.2: Obrázok ilustruje model pre používanie vetiev pri vývoji



Obr. 8.3: Obrázok zobrazuje tagy pre verziu 1.1.0 a jej fix vo verzii v1.1.1

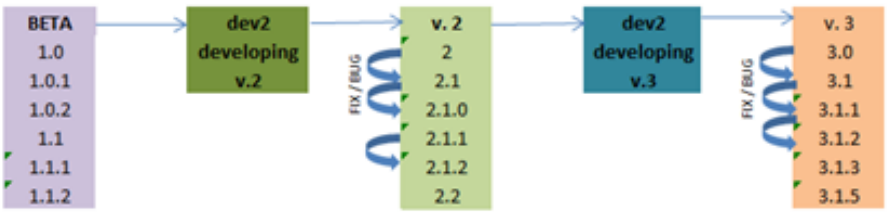
Verzie projektu

Každý merge na vrstvu master predstavuje vydanie novej stabilnej verzie systému. Verzie systému sú štandardne označované celým číslom typu: 1,2,3.

V prípade ak sa jedná o opravu chyby produkčnej verzie (FIX alebo BUG) na vetve master, takto opravená verzia má označenie typu A.B.C, pričom sa inkrementuje posledná hodnota.

Číslo vo vyššej úrovni v označení verzie (viac vpravo), predstavuje opravy chýb alebo vylepšení (viď. tagy v časti: 4.1).

Vývoj novej verzie má označenie prefixom „dev“ – napr dev2. Príklad vývoja verzií je zobrazený na obrázku.



Obr. 8.4: Obrázok zobrazuje spôsob značenia verzií

Verzia BETA je označovaná pod číslom 1. Keďže projekt je prevažne webová

aplikácia, vytváraná ako war balíček (REST WS), produkčná a vývojová verzia, ktoré sú nasadené na serveri, majú nasledovné označenia:

Produkčná verzia: TP_team15
cesta: http://server.xy/TP_team15
Vývojová verzia: TP_team15_devXX (XX predstavuje číslo aktuálnej vývojovej verzie)
cesta: http://server.xy/TP_team15_devXX

Produkčná verzia

Po ukončení vývoja verzie devXX, je potrebné pred vydaním a nasadením na produkčnú (stabilnú) verziu:

- vykonať automatizované reverzné testovanie
- na tejto verzii vykonať nové automatizované testy
- vykonať manuálne testovacie scénare
- aktualizovať javadoc
- deploy produkčnej verzie projektu - TP_team15
- archivácia TP_team15_devXX.war na dropboxe a undeploy devXX verzie

Kapitola 9

Spôsob komunikácie a komunikačné kanály - Metodika

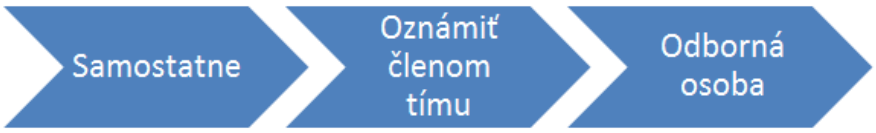
Autor: Jozef Melko

Úvod

Komunikácia v tíme je veľmi dôležitá. Má rôzne podoby od písaného textu, cez obrázky až po hovorené slovo. Je to spôsob, ako sú členovia tímu vždy v kontakte. V prípade, že člen tímu má problém s nejakou úlohou alebo nejakú otázku, komunikuje práve s ostatnými členmi aby daný problém alebo nejasnosť vyriešil rýchlo a správne. Od problému alebo otázky závisí, či je potrebná komunikácia s každým členom týmu, alebo iba s niektorými. Je dôležité aby sa ostatní členovia tímu venovali svojím úlohám a preto, ak to nie je potrebné, nemusí sa zvolávať celý tím. Princípom je aby sa všetky problémy vyriešili s pokojom. To znamená žiadne hanlivé slová, urážky a hlúpe poznámky.

Spôsob komunikácie pri riešení zadanej úlohy

Člen tímu sa snaží najskôr úlohu riešiť sám. Ak nastal problém, tak to skúša riešiť bez pomoci. Nerobí to však veľmi dlho. V prípade, že mu to trvá dlhšie, tak s problémom oboznámi tím, ktorého členovia sa mu snažia pomôcť. Jasne a zreteľne napíše názov problému maximálne troma slovami. Potom problém opíše. V prípade, že program funguje, napíše akú chybu program robí a potom napíše čo sa očakáva že má program robiť. Ak sa problém nepodarilo vyriešiť v rámci tímu, nasleduje oslovenie vedúcej projektu, a oznámi jej daný problém. Ona nás, na základe problémovej oblasti, ohlási na odbornú pomoc. Odbornou pomocou sa myslí, že sa dohodne stretnutie s osobou, ktorá sa danej oblasti rozumie a s problémom nám pomôže. Ak člen tímu rieši nejakú úlohu, ktorá ovplyvňuje build projektu (zmena classpath/ pridanie knižnice), tak je potrebné po commite informovať ostatných členov tímu prostredníctvom skype.



Spôsob komunikácie pri riešení bugu

Pokiaľ sme našli v programe nejakú chybu, kontaktujeme konkrétneho člena, ktorý implementoval časť, ktorá vytvára bug. Za cieľom lepšieho riešenia problému, člen tímu, ktorý chybu našiel vytvorí report chyby.
Report chyby:

- Poradové číslo chyby
- Názov chyby
- Opis chyby (čo program vykonáva)
- Ako by to malo byť správne

Keď sa chyba vyriešila, riešiteľ to oznámi osobám, ktorí boli s daným problémom oboznámení o tom, že sa problém podarilo vyriešiť.

Komunikačné kanály

Skype

Skype používame v prípade, že máme s niečím problém a nevieme s ním pohnúť. Skype môžeme použiť aj v prípade, že s niekým na nejakom riešení spolupracujeme. Nepíšu sa tam správy typu kto má čo urobiť lebo na to je určená JIRA. Ak s niekým spolupracujeme alebo potrebujeme niečo od konkrétneho člena, píšeme mu správu osobne a nepíšeme to do skupinovej konverzácie. Ak máme niečo na srdci, čo by sme nechceli aby sa stratilo, tak to nepíšeme do skupinovej konverzácie ale použijeme iný komunikačný kanál. Skype takisto používame na hovor alebo videohovor. Praktizujeme v prípade, že sa vedúca tímového projektu osobne zúčastniť nemôže. Skupinový hovor alebo videohovor uskutočňujeme ak je stretnutie neosobné ale má mať náplň osobného stretnutia akoby sme boli spolu na rovnakom mieste. Vzhľadom na kvalitnú komunikáciu v tíme je skype veľmi dobrým a využívaným kanálom. V komunikácií zastáva funkciu ako normálna konverzácia. Funguje ako chat, kde sa

členovia tímu rozprávajú, radia si ohľadom projektu. V tomto komunikačnom kanály neexistuje žiadna šablóna, podľa ktorej by mali členovia formulovať svoje správy.

Nástenka na facebook-ovej skupine

Na nástenku sa píšu príspevky, ktoré nechceme aby sa rýchlo stratili v skupinovom chate na skype alebo aby sa na ne nezabudlo na stretnutí. Sú to veci ako prihlasovacie údaje k veciam potrebným na bezproblémový beh projektu. Na nástenku FB stránky sa píšu aj príspevky spojené s informáciami od vedúcej TP alebo iné dôležité informácie. V prípade, že vieme dôležitú informáciu hoci z e-mailu tak je potrebné ju hodiť na nástenku, lebo nie každý môže byť tak pohotový aby si tú informáciu na spoločnom maily prečítal včas. Ďalšie príspevky ktoré nechceme aby sa stratili sú informácie, ktoré sme sa dozvedeli skôr ako budú potrebné. Keďže sa použijú v projekte neskôr tak je dôležité ich dať na miesto kde budú jednoducho prístupné a budú dobre viditeľné. Skupina na facebooku je dobrá aj v prípade, keď sa informácia viaže k nejakému dátumu, ktorý treba striktne dodržať. Ak je potrebné o niečom hlasovať, v tom prípade sa vytvára v skupine príspevok na hlasovanie. Podobne ako na skype, tak ani na facebooku nie je štruktúra, podľa ktorej je nutné písať príspevky.

Ak máme informáciu k projektu od osoby, je dôležité napísať od koho to je. Takisto je dôležité stručne a jasne formulovať čoho sa to týka, čo sa od nás vyžaduje. Medzi dôležité informácie patria aj tie o konaní stretnutí. Je dôležité sa dohodnúť na vhodnom termíne pre všetkých. Keďže existuje demokracia tak to znamená, že sa tím nebude podriaďovať menšine ohľadom času stretnutia. Keďže chceme mať kvalitnú komunikáciu tak sa vyberá najvhodnejší termín. Následne sa napíše na nástenku skupiny informácie o stretnutí:

- Dátum a čas
- Čo sa má riešiť na stretnutí
  - meno člena
  - čo by chcel riešiť – stručný popis, prípadne s kým by chcel daný problém riešiť

Stretnutia

Stretnutia sú minimálne raz do týždňa. To povinné stretnutie je vo štvrtok. Na tomto stretnutí zhodnocujeme celý týždeň. Čo sme urobili a čo nie. Čo na koľko percent splnené. Na týchto stretnutiach sme všetci členovia tímu a aj vedúca projektu. Čiže v rámci dobrej komunikácie je dôležité pýtať sa ak niečomu nerozumieme, lebo je to hneď vysvetlené. Ak sa počas týždňa vyskytne viac problémov ktoré sa dajú riešiť len osobným stretnutím, členovia tímu sa dohodnú na termíne kedy sa stretnutie uskutoční. Keďže dosť komunikujeme na skype aj v škole keď sa stretneme, tak to je postačujúci aj ten jeden termín stretnutia.

E-mail

Dôležitým komunikačným kanálom v našom tíme je e-mailová adresa. Do tejto adresy nám chodia dôležité správy od osôb, ktoré v danom odbore pracujú a spolupracujú s nami na projekte. Sú to požiadavky alebo pomocné materiály, ktoré nám výrazne uľahčujú prácu. Takisto sú to správy, ktoré obsahujú informácie, ku ktorým by sme sa nedostali alebo je to literatúram ktorú odporúčajú. Na vhodnú reprezentáciu tímu a komunikovanie s externými osobami je oficiálny mail veľmi dôležitý a môžu ho používať všetci členovia.

Aby bola práca kvalitná je komunikácia veľmi dôležitá a preto je dôležité včas a vhodne komunikovať.

Motto tímu

Pýtať sa, pýtať sa, pýtať sa.

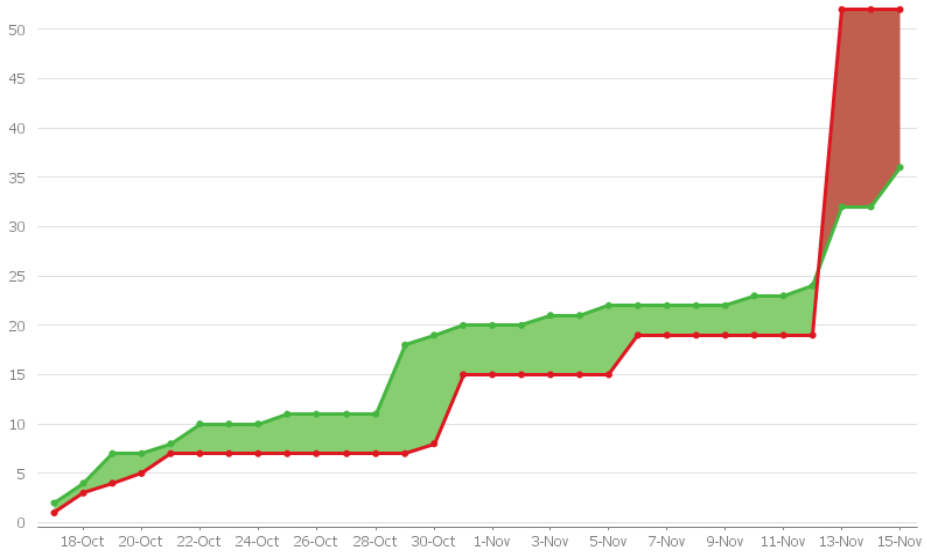
Kapitola 10

Export evidencie úloh

Táto podkapitola poskytuje prehľad úloh vytvorených počas riešnia projektu. Exporty týchto úloh sú rozdelené do dvoch častí. Prvá časť prezentuje celkový doterajši stav, zatiaľ čo druhá sa sústreďuje na prezentáciu čiastkových stavov.

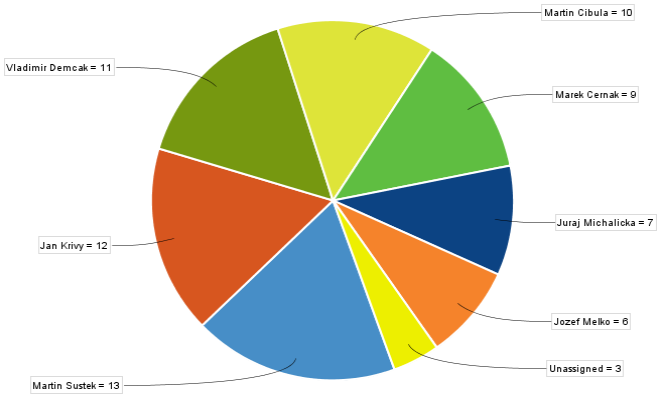
10.1 Celkový stav

Nasledujúci graf prezentuje vývoj pomeru vytvorených úloh ku vyriešeným úlohám v čase. Z vývoja krivky môže pozorovateľ usúdiť, že tím priebežne rieši stanovené úlohy. Výnimkou v tomto stave je obdobie vytvárania projektovej dokumentácie, v ktorom vzniklo veľké množstvo úloh na vysokej úrovni granularity, čím došlo k dočasnému narušeniu nastoleného štandardu. Pozn. graf bol generovaný v rannom štádiu vytvárania projektovej dokumentácie.



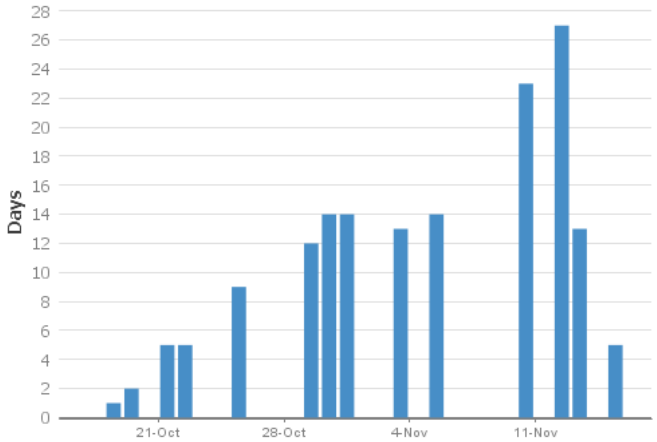
Obr. 10.1: Pomer vytvorených úloh ku vyriešeným úlohám

Ďalší graf zobrazuje rozloženie úloh na jednotlivých členov tímu. Úlohou sa v kontexte tohto grafu rozumie úloha na hocakej úrovni granularity, vrátane podúloh. Z grafu môžeme vidieť, že úlohy boli adekvátne distribuované medzi jednotlivých členov tímu. Graf však nevypovedá o zložitosti úloh a preto môže byť toto rozloženie mierne zavádzajúce. V kompetencii tímového vedúceho však bolo zabezpečiť, ab boli úlohy rozdelené rovnomerne nie len z hľadiska počtu, ale aj náročnosti.



Obr. 10.2: Počet úloh na člena tímu

Z posledného grafu v tejto časti prehľadu úloh môžeme odpozorovať trend zvyšujúcej sa náročnosti úloh. Na grafe vieme identifikovať približné umiestnenie šprintov, na základe čoho vieme tento trend obhájiť tým, že prvý šprint slúžil na oboznámenie sa s doménou a prácou v tíme, zatiaľ čo ďalšie šprinty už obsahovali aj úlohy implementačného charakteru. V treťom šprinte taktiež prebehol refaktoring, ktorý spomalil celkový vývoj. Dovoľujeme si teda tvrdiť, že ďalší vývoj časovej zložitosti riešenia úloh sa ustáli na úrovni šprintu číslo 2.



Obr. 10.3: Čas potrebný na vyriešenie úlohy v čase

10.2 Stav po šprintoch

Táto podkapitola prezentuje vývoj úloh v priebehu riešenia projektu, rozdelený po šprintoch. Šprint číslo 1 nie je do kapitoly zahrnutý, nakoľko slúžil iba na oboznámenie sa s doménou a s kompetenciami účastníkov a nebol počas neho vytvorený projekt v systéme JiRa. Nebolo tak možné získať export o stave úloh.

2. šprint - hlavnou náplňou tohto šprintu boli podporné úlohy. Bolo potrebné počas neho vytvoriť repozitár, nastaviť vývojové prostredie, vykonať prvé dopyty na APIs citačných indexov. Stanovené úlohy sa podarilo vyriešiť, základný prototyp bol vytvorený a nebolo tak nutné žiadne úlohy prenášať do ďalšieho šprintu.

T	Key	Components	Summary	Assignee	Reporter	P	Status	Resolution	Created	Updated	Due
	BIGDATA-24		Cookie problem	Vladimir Demcak	Vladimir Demcak		Resolved	Fixed	20/Oct/14	29/Oct/14	
	BIGDATA-23		Vytvorenie wiki	Jan Krivy	Jan Krivy		Closed	Fixed	19/Oct/14	23/Oct/14	22/Oct/14
	BIGDATA-21		BIGDATA-6 / Vytvorit "metodu" na servlete pre ZISKANIE referencie pre clanok na SCOPUS /elsevier.com/	Martin Cibula	Vladimir Demcak		Closed	Fixed	18/Oct/14	13/Nov/14	
	BIGDATA-20		BIGDATA-6 / Vyhladavat v scopuse diela cez meno autora a nazvu diela	Jozef Melko	Jan Krivy		Closed	Done	17/Oct/14	15/Nov/14	22/Oct/14
	BIGDATA-19		Vytvorenie tried entit + connection manager	Marek Cernak	Marek Cernak		Closed	Fixed	16/Oct/14	29/Oct/14	22/Oct/14
	BIGDATA-17		BIGDATA-6 / Ziskavat pomocou SCOPUSu cez servlet zakladne data	Vladimir Demcak	Vladimir Demcak		Closed	Done	16/Oct/14	31/Oct/14	19/Oct/14
	BIGDATA-12		Plány pre prvé 2 šprinty	Martin Sustek	Martin Sustek		Resolved	Fixed	16/Oct/14	29/Oct/14	20/Oct/14
	BIGDATA-6		Vývoj základného servletu	Vladimir Demcak	Martin Sustek		Resolved	Fixed	16/Oct/14	31/Oct/14	29/Oct/14
	BIGDATA-1		Vytvorenie a setup bitbucket repozitara	Martin Sustek	Martin Sustek		Closed	Fixed	16/Oct/14	29/Oct/14	18/Oct/14

Obr. 10.4: Export úloh zo systému JiRa za obdobie 16.10.2014 - 30.10.2014

3. šprint - šprint bol zameraný na rozšírenie existujúcej funkcionality prototypu a vytvorenie prvej verzie aplikácie. Táto mala byť schopná poskytnúť výsledky pre dopyty týkajúce sa vyhľadania diel autora a ohlasov na konkrétne dielo. Taktiež bola doplnená funkcionalita ukladania získaných údajov do navrhnutej databázovej schémy. Dôležitou časťou šprintu bolo vykonanie refaktoringu kódu. S použitím princípov softvérového návrhu boli rozbité monolitické triedy a vytvorená hierarchia tried, z ktorých aplikácia pozostáva a ktoré budú ďalej rozširované. Na záver šprintu boli vytvorené prvé testovacie scenáre a adekvátne ku nim boli doplnené testy.

T	Key	Components	Summary	Assignee	Reporter	P	Status	Resolution	Created	Updated	Due
	BIGDATA-34		Konfigurácia do WIKI	Marek Cernak	Martin Sustek		Resolved	Fixed	31/Oct/14	15/Nov/14	06/Nov/14
	BIGDATA-32		BIGDATA-28 / Vytvorenie POST metódy pre ukladanie ohlasov na dielo	Martin Cibula	Martin Sustek		Closed	Incomplete	31/Oct/14	15/Nov/14	10/Nov/14
	BIGDATA-31		BIGDATA-28 / Ukladanie získaných údajov do databázy	Marek Cernak	Martin Sustek		Closed	Fixed	31/Oct/14	15/Nov/14	09/Nov/14
	BIGDATA-30		BIGDATA-28 / Vytvorenie POST metódy pre vyhľadanie diela	Vladimir Demcak	Martin Sustek		Closed	Done	31/Oct/14	15/Nov/14	05/Nov/14
	BIGDATA-29		Refaktoring kódu	Martin Sustek	Martin Sustek		Closed	Fixed	31/Oct/14	15/Nov/14	02/Nov/14
	BIGDATA-28		Programovanie aplikácie 1.1.0	Vladimir Demcak	Martin Sustek		Closed	Fixed	31/Oct/14	15/Nov/14	13/Nov/14
	BIGDATA-27		Vytvaranie Testov	Jozef Melko	Vladimir Demcak		Closed	Fixed	30/Oct/14	15/Nov/14	12/Nov/14
	BIGDATA-26		Vytvorit konvenciu pomenovavania dopytov	Jan Krivy	Vladimir Demcak		Closed	Won't Fix	21/Oct/14	05/Nov/14	
	BIGDATA-25		BIGDATA-28 / Vyhladanie dokumentu podla autora a nazvu diela	Juraj Michalicka	Jan Krivy		Closed	Done	21/Oct/14	03/Nov/14	30/Oct/14
	BIGDATA-22		BIGDATA-28 / Vytvaranie SOAPMessage	Jan Krivy	Vladimir Demcak		Closed	Fixed	18/Oct/14	10/Nov/14	
	BIGDATA-16		BIGDATA-27 / Testy pre základný servlet	Juraj Michalicka	Martin Sustek		Closed	Won't Fix	16/Oct/14	13/Nov/14	21/Oct/14
	BIGDATA-5		BIGDATA-27 / Príprava testovacích dát	Jan Krivy	Martin Sustek		Closed	Fixed	16/Oct/14	13/Nov/14	20/Oct/14

Obr. 10.5: Export úloh zo systému JiRa za obdobie 30.10.2014 - 13.11.2014

4. šprint - aktuálne prebiehajúci šprint. Najväčšou úlohou šprintu je zdokumentovanie doteraz vytvorenej funkcionality do projektovej dokumentácie, ktorá bude jedným z výstupov riešenia projektu. Po jej doplnení pristúpia členovia tímu k doplneniu funkcionality pre upravené workflowy obslúženia dopytu. Časť tímu sa bude taktiež zaoberať vytvorením rozhrania pre zadávanie testovacích dopytov. V tomto šprinte boli úlohy rozpísané na vyššiu úroveň granularity, aby bolo možné lepšie sledovať mieru ich naplnenia. K tomuto kroku bolo pristúpené po skúsenosti z predošlého šprintu. Taktiež bolo nutnosťou prenášať implementáciu viacerých úloh z predchádzajúceho šprintu. K tejto udalosti došlo následkom nedostatočnej komunikácie pri počiatočných problémoch s riešením daných úloh.

T	Key	Components	Summary	Assignee	Reporter	P	Status	Resolution	Created	Updated	Due
	BIGDATA-72		BIGDATA-43 / Aktualizacia finalneho dokumentu	Marek Cernak	Jan Krivy		Open	*Unresolved*	13/Nov/14	17/Nov/14	
	BIGDATA-71		BIGDATA-43 / Zoznam kompetencii timu	Martin Cibula	Jan Krivy		Closed	Done	13/Nov/14	17/Nov/14	
	BIGDATA-70		BIGDATA-43 / Exporty z jiry	Martin Sustek	Jan Krivy		Closed	Fixed	13/Nov/14	18/Nov/14	
	BIGDATA-69		BIGDATA-43 / Oblasti manazmentu - Kanaly komunikacii a zhodnotenie ich dodrziavania	Jozef Melko	Jan Krivy		Closed	Done	13/Nov/14	19/Nov/14	
	BIGDATA-68		BIGDATA-43 / Oblasti manazmentu - Narocnost setupu a odovodenie pouzitia technologii	Marek Cernak	Jan Krivy		Closed	Fixed	13/Nov/14	18/Nov/14	
	BIGDATA-67		BIGDATA-43 / Oblasti manazmentu - Kvalita architektury produktu	Juraj Michalicka	Jan Krivy		Closed	Done	13/Nov/14	18/Nov/14	
	BIGDATA-64		BIGDATA-43 / Oblasti manazmentu - opis vykonnosti ucastnikov	Martin Sustek	Jan Krivy		Open	*Unresolved*	13/Nov/14	13/Nov/14	
	BIGDATA-63		BIGDATA-43 / Oblasti manazmentu - analyza rizik	Vladimir Demcak	Jan Krivy		Closed	Done	13/Nov/14	17/Nov/14	
	BIGDATA-62		BIGDATA-43 / Vytvorenie kapitoly metodiky	Jan Krivy	Jan Krivy		Closed	Fixed	13/Nov/14	18/Nov/14	
	BIGDATA-61		BIGDATA-43 / Retrospektivy	Juraj Michalicka	Jan Krivy		In Progress	*Unresolved*	13/Nov/14	19/Nov/14	
	BIGDATA-60		BIGDATA-43 / Vytvorenie kapitoly Celkový pohľad na systém	Martin Sustek	Jan Krivy		Open	*Unresolved*	13/Nov/14	13/Nov/14	
	BIGDATA-58		BIGDATA-43 / Uvod dokumentu	Marek Cernak	Jan Krivy		Closed	Fixed	13/Nov/14	19/Nov/14	
	BIGDATA-57		BIGDATA-43 / Vytvorenie sablon dokumentu	Martin Sustek	Jan Krivy		Closed	Fixed	13/Nov/14	17/Nov/14	
	BIGDATA-55		BIGDATA-42 / Vytvorenie testov podla testovacich scenarov	Juraj Michalicka	Jan Krivy		Open	*Unresolved*	13/Nov/14	13/Nov/14	
	BIGDATA-54		BIGDATA-42 / Uprava metody pre vyhladanie dokumentu	Vladimir Demcak	Jan Krivy		Open	*Unresolved*	13/Nov/14	13/Nov/14	
	BIGDATA-53		BIGDATA-42 / Uprava metody na vyhladavanie ohlasov	Martin Cibula	Jan Krivy		Open	*Unresolved*	13/Nov/14	13/Nov/14	
	BIGDATA-52		BIGDATA-42 / Upravenie requestov a response	Jan Krivy	Jan Krivy		Resolved	Fixed	13/Nov/14	15/Nov/14	
	BIGDATA-51		BIGDATA-42 / Doplnenie testovacich scenarov	Jozef Melko	Jan Krivy		Open	*Unresolved*	13/Nov/14	13/Nov/14	
	BIGDATA-50		BIGDATA-42 / Prerobit workflow diagramy	Marek Cernak	Jan Krivy		Resolved	Fixed	13/Nov/14	15/Nov/14	
	BIGDATA-49		BIGDATA-37 / Mauál k testovaciemu rozhraniu	Martin Sustek	Martin Sustek		Open	*Unresolved*	13/Nov/14	13/Nov/14	27/Nov/14
	BIGDATA-48		BIGDATA-37 / Testy pre testovacie rozhranie	Martin Cibula	Martin Sustek		Open	*Unresolved*	13/Nov/14	13/Nov/14	27/Nov/14
	BIGDATA-47		BIGDATA-37 / Implementácia testovacieho rozhrania	Martin Sustek	Martin Sustek		Open	*Unresolved*	13/Nov/14	13/Nov/14	25/Nov/14
	BIGDATA-46		BIGDATA-37 / Schválenie návrhu rozhrania	Martin Sustek	Martin Sustek		Open	*Unresolved*	13/Nov/14	13/Nov/14	23/Nov/14
	BIGDATA-45		BIGDATA-37 / Návrh rozhrania	Martin Cibula	Martin Sustek		Open	*Unresolved*	13/Nov/14	13/Nov/14	22/Nov/14
	BIGDATA-43		Big picture	Jan Krivy	Martin Sustek		Open	*Unresolved*	13/Nov/14	13/Nov/14	21/Nov/14
	BIGDATA-42		Rozšírenie prípadov použitia	Jan Krivy	Martin Sustek		Open	*Unresolved*	13/Nov/14	13/Nov/14	20/Nov/14
	BIGDATA-41		Načítavanie konfigurácie pri štarte servletu	Vladimir Demcak	Martin Sustek		Resolved	Done	13/Nov/14	13/Nov/14	27/Nov/14
	BIGDATA-40		Ošetrenie chybného vstupu POST metód	Martin Cibula	Martin Sustek		Resolved	Fixed	13/Nov/14	16/Nov/14	20/Nov/14
	BIGDATA-39		Vytváranie testov pre existujúce testovacie scenáre	Jozef Melko	Martin Sustek		Open	*Unresolved*	13/Nov/14	13/Nov/14	27/Nov/14
	BIGDATA-37		Rozhranie pre zadávanie testovacích dopytov	Martin Sustek	Martin Sustek		Open	*Unresolved*	06/Nov/14	13/Nov/14	
	BIGDATA-33		Hierarchia výnimiek	Juraj Michalicka	Martin Sustek		Open	*Unresolved*	31/Oct/14	15/Nov/14	06/Nov/14

Obr. 10.6: Export úloh zo systému JiRa za obdobie 13.11.2014 - 16.11.2014

Dodatok A

Preberací protokol

PREBERACÍ PROTOKOL

Tímový projekt 2014/2015

Tím 15 - Nitrogen family

Predmet odovzdávania:

- Dokumentácia k inžinierskemu dielu (priebežná verzia za zimný semester)
- Dokumentácia riadenia (priebežná verzia za zimný semester)

Vedúca projektu: Ing. Nadežda Andrejčíková, PhD.
Podpisom potvrdzuje prevzatie vyššie uvedených častí projektu a/alebo dokumentácie

V Bratislave

.....	.....
Dátum	*Vlastnoručný podpis*