Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

# reCommers

## Metodika k prehliadkam kódu

**Tím:** 03 - reCommers
**Vedúci tímu:** Ing. Ivan Srba, PhD.
**Vyhotovil:** Matúš Cuper
**Dátum poslednej zmeny:** 10.05. 2018
**Verzia:** 1.4

# 1. Úvod

Táto metodika je venovaná prehliadkam kódu. Popisuje ako vzniká potreba prezerať kód, ako vyzerá, ako prebieha samotná prehliadka a ako sú zapracované pripomienky do zdrojového kódu. Uvedené postupy je možné v obmedzenej miere uplatniť aj na vznikajúce dokumenty.

# 2. Prehliadka kódu

Nasledujúce podkapitoly opisujú všetky fázy prehliadky kódu.

## 2.1. Potreba prehliadky kódu

Potreba môže vyplynúť z plánovania šprintu, môže o ňu požiadať *product owner*, *scrum master* alebo aj samotný člen tímu, ktorý danú funkcionalitu implementuje. Člen tímu zodpovedný za implementáciu je povinný vytvoriť v systéme na správu úloh novú úlohu s názvom *Code review* a prideliť jej predpokladané množstvo hodín. Nie je nutné, aby úloha bola ihneď niekomu pridelená. Jeden člen z tímu si ju môže prideliť z vlastnej iniciatívy.

## 2.2. Pridelenie úlohy prehliadky kódu

Ak je úloha nepridelená aj po dokončení implementácie, člen tímu, ktorý za ňu zodpovedá napíše do *Slacku* správu, v ktorej vyzve ostatných členov tímu k prideleniu úlohy. Správa je poslaná do kanálu *code-reviews* a obsahuje označenie @*channel*, čím sú notifikovaní všetci členovia kanálu. Po diskusii je úloha pridelená jednému z členov tímu. Zo začiatku tímového projektu robia prehliadky najmä členovia tímu, ktorí sú skúsenejší v danej oblasti. Prehliadku vykonáva spravidla jeden člen tímu, v prípade že sa autor a *reviewer* nezhodnú, môžu komunikovať aj s ostatnými členmi tímu. Ak prehliadku vykonáva viacero členov, musia mať vytvorené osobitné úlohy v systéme na správu úloh.

## 2.3. Začatie prehliadky kódu

Po implementovaní funkcionality v lokálnej vetve je potrebné zapracovať zmeny z vetvy *dev*, ktoré mohli medzičasom pribudnúť. Každý *commit* musí byť plne funkčný, a preto môže vzniknúť potreba ich agregácie. Tento krok nie je potrebný, nakoľko ho je možné vykonať až po schválení požiadavky na zlúčenie vetiev. Počet *commits* môže byť vo výsledku rôzny, no zároveň by mal zlepšovať čitateľnosť autorových zmien.

| | |
|---|---|
| git checkout 6461_add_feature | prepnutie na *feature branch* |
| git rebase dev | zapracovanie zmien z *dev* vetvy |
| | |
| git rebase -i HEAD~3 | posledné 3 *commits* sú agregované do jedného |

Otvorí sa okno, kde sú zobrazené posledné 3 *commit messages*

```
pick 01d1124  Add something
pick 6340aaa  Rename something
pick ebfd367  Move something
```

Agregáciu posledných 2 *commits* do jedného, spravíme prepísaním súboru na:

```
pick    01d1124    Add something
squash  6340aaa    Rename something
squash  ebfd367    Move something

git push origin 6461_add_feature    pridanie vetvy na server
```

Prehliadka kódu začína kliknutím na *New pull request* v githube, podľa nasledujúcich obrázkov

Obr. 1: Vytvorenie požiadavky na prehliadku kódu - *Pull Request*

Následne je nutné vybrať vetvy, ktoré sa budú zlučovať, vybrať kolegu, ktorý bude vykonávať prehliadku kódu a vložiť sumarizačný popisu implementovaných zmien. Ak názov vetvy jednoznačne určuje úlohu, ktorej sa týka pomocou *Task ID*, nie je nutné pridávať sumarizačný popis.

Obr. 2: Potvrdenie vytvorenia prehliadky kódu

Autor môže vytvoriť *pull request* iba v prípade ak spĺňa všetky nasledujúce body:

- vytvorený zdrojový kód formálne zodpovedá definovaným štandardom
- nachádza sa v zodpovedajúcej vetve repozitára
- bola vytvorená žiadosť o zlúčenie s vetvou *dev*
- úloha *Code review* má prideleného člena tímu, ktorý bol oboznámený s dokončením implementácie

## 2.4. Priebeh prehliadky kódu

Účelom je najmä nájsť logické chyby v zdrojovom kóde, *reviewer* preto overuje a dohliada na:

- dodržanie požiadavok úlohy
- zvážiť potenciálne vedľajšie efekty - pridávanie novej funkcionality nesmie zmeniť existujúce funkcie
- čitateľnosť a jednoduchosť - z použitých názvov musí byť zrejmý ich význam, zdrojový kód nesmie byť komplikovaný ani pre niekoho nezainteresovaného
- spracovanie výnimiek - overenie správania kódu pri neplatných vstupoch
- dodržanie DRY (Don’t Repeat Yourself) - logika programu by musí byť zachytená vo funkciách, aby sa predišlo kopírovaniu rovnakého kódu na viaceré miesta
- spoľahlivosť - úniky pamäte, pretečenie pamäte, bezpečnosť, atď.

Komentáre musia byť vecné a musia v krátkosti popisovať navrhované riešenie. *Reviewer* ich píše k jednotlivým častiam kódu, výhradne v github. Ak má pripomienky ku kódu ako celku, kontaktuje autora osobne, prostredníctvom súkromnej správy v *Slacku* alebo v kanály *code-reviews*. V prípade, že sa členovia tímu nevedia zhodnúť na riešení, musia požiadať

ostatných členov o vyjadrenie svojho postoju v kanály *code-reviews* alebo priamo v komentároch v zdrojovom kóde, tak ako znázorňuje nasledujúci obrázok

Obr. 3: Ukážka konverzácie členov tímu ku konkrétnym riadkom kódu

Nový komentár automaticky spustí prehliadku kódu. Je možné pridávať komentáre ku konkrétnym riadkom kódu, alebo aj k celej prehliadke.

### 2.5. Zapracovanie pripomienok

Výsledný návrh musí autor zapracovať do zdrojového kódu a opäť prejsť celým procesom prehliadania kódu. Nie je potrebné vytvárať nový *pull request*, nakoľko sa nové *commits* automaticky zobrazia.

### 2.6. Ukončenie prehliadky kódu

Vetvu je možné zlúčiť s vetvou *dev* v prípade ak sú splnené nasledujúce podmienky:

- *reviewer* akceptoval implementovanú funkcionalitu v jej celom rozsahu
- všetky predchádzajúce a nové testy prebehli úspešne
- všetky pripomienky v podobe komentárov sú vyriešené

Nakoniec *reviewer* schváli prehliadku zvolením *Approve* a kliknutím na *Submit review*

Obr. 4: Schválenie prehliadky kódu

Zlúčenie vetvy robí *reviewer* kliknutím na *Squash and merge*

Obr. 5: Schválená prehliadka

Predtým však ešte zvolí správny typ zlúčenia vetiev

Obr. 6: Výber typu zlúčenia vetiev

Nakoniec je potrebné zlúčenú vetvu odstrániť, kliknutím na tlačidlo *Delete branch*

Obr. 7: Odstránenie zlúčenej vetvy