Pregledi koda

Seminarski rad u okviru kursa
Metodologija stručnog i naučnog rada
Matematički fakultet

Nikola Janković, Anđela Milićević, Katarina Savičić, Dunja Spasić
nikola_jankovic@tuta.io, milicevica_32@gmail.com,
katarina_savicic@hotmail.com, spasicdunja3013@gmail.com

21. mart 2020.

Sažetak

Sadržaj

1 Uvod

Kada budete predavali seminarski rad, imenujete datoteke tako da sadrže redni broj teme, temu seminarskog rada, kao i prezimena članova grupe. Precizna uputstva na temu imenovnja će biti data na formi za predaju seminarskog rada. Predaja seminarskih radova biće isključivo preko veb forme, a NE slanjem mejla. Link na formu će biti dat u okviru obaveštenja na strani kursa. Vodite računa da prilikom predavanja seminarskog rada predate samo one fajlove koji su neophodni za ponovno generisanje pdf datoteke. To znači da pomoćne fajlove, kao što su .log, .out, .blg, .toc, .aux i slično, **ne treba predavati**.

2 Vrste pregleda

2.1 Formalni pregledi

2.2 Neformalni pregledi

Istraživanja pokazuju da druge vrste pregleda koda, osim formalnih metoda, mogu da daju priblizhno dobre rezultate, sa mnogo manje uloženog novca u obučavanje i planiranje. Formalne metode sastanaka su ipak pokazale značajnu prednost u sprečavanje lažno pozitivnih problema (sprečavanje pronalaženje grešaka koje ne predstavljaju prave greške). Osim toga, 30% pronađenih grešaka na formalnim sastancima je otkriveno zajednički, kroz razgovor.[10] Uprkos tome, neformalne metode pregledanja koda štede dosta resursa i vremena pa je je korisno da se detaljnije razmotre.

2.2.1 Pregled preko ramena

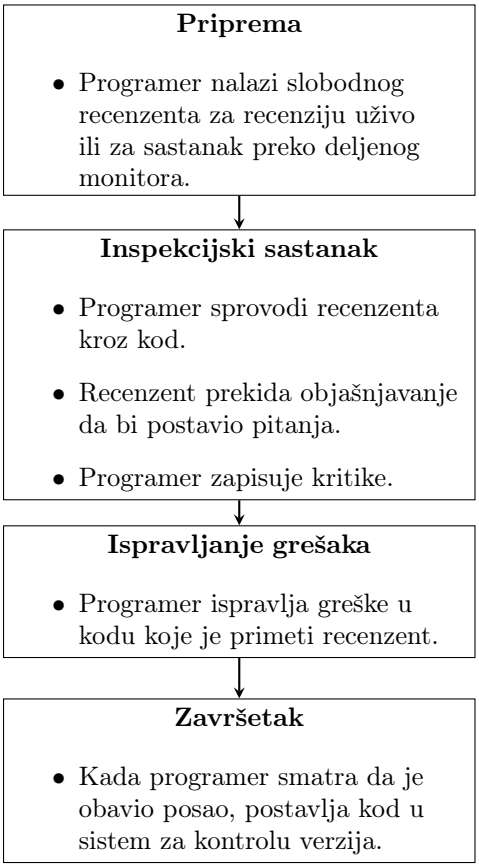
Pregled koda *preko ramena* podrazumeva da programer stoji nad radnim stolom autora koda, dok autor sprovodi recenzenta kroz najnovije izmene koda. Obično autor sedi za računarom, otvara razne fajlove i pokazuje nove linije koda koje su dodate ili one koje su izbrisane. Ako recenzent primeti sitne nepravilnosti, može odmah da ih istakne programeru i da one budu izmenjene na licu mesta.

Sa pojavom novih softvera za deljenje ekrana, pregledi koda korišćenjem metode *preko ramena* mogu da se obavljaju i na veće razdaljine. Ovakva komunikacija komplikuje proces pregleda koda jer je potrebno zakazati sanstanke ili telefonske razgovore.

Prednost ove metode je jednostavnost. Bilo ko može da učestvuje u pregledu preko ramena bez prethodnog obučavanja. Još jedna značajna prednost je što može da se primeni bilo kad, što je bitno kada potrebno pregledati jako važnu izmenu koda. Generalno, svi pregledi koda uživo su dosta korisni jer pružaju programerima mogućnost da razmene ideje koje ne bi delili preko imejla ili poruka.

Jednostavnost i neformalnost ovog pristupa takođe za sobom nose i neke nedostatke. Glavna mana je nemogućnost provere da li su pregledane sve izmena koda. Ne postoje nikakvi izveštaji, ili mere koje bi dokumentovale proces. Još jedan problem je što može veoma lako da se desi da autoru promakne da je napravio izmenu kada svoj kod prezentuje recenzentu. Recenzent jedino vidi ono što mu autor pokaže, nema mogućnost da sam pregleda ostale datoteke na koje bi ta izmena mogla da utiče. Ako autor napravi izmenu koja nije potpuno jasna svakom programeru bez pojašnjenja autora, sledećem programeru koji pregleda ili koristi kod

neće biti jasna izmena. Pregled koda *preko ramena* može efikasno da finkcioniše samo ako su autor i recenzent fizički blizu, u istoj prostoriji ili u rade u susednim zgradama. Svako prebacivanje recenzije na elektronski vid komunikacije gubi poentu same ideje pregleda *preko ramena*.



Slika 1: Dijagram pregleda koda *preko ramena*.

2.2.2 Preko mejla

2.2.3 Programiranje u paru

2.2.4 Asistenti alati

3 Alati

Moguće je naći veći broj softverskih rešenja koji mogu da posluže kao pomoć pri pregledu koda.
Podelićemo ih u dve bitne kategorije:

- Vlasnički
- Otvorenog koda

Odlučili smo se da više prostora odvojimo za opcije iz druge grupe jer verujemo da akademska misao i koncept softvera otvorenog koda imaju

veliku spregnutost.
Nabrojaćemo neke popularnije alate iz prve grupe, a čitaocu ćemo ostaviti referentne lokacije ukoliko je zainteresovan da više istraži:

- *Collaborator* [2]
- *CodeScene* [3]
- *Crucible* [6]
- *Veracode* [4]
- *Jarchitect* [5]

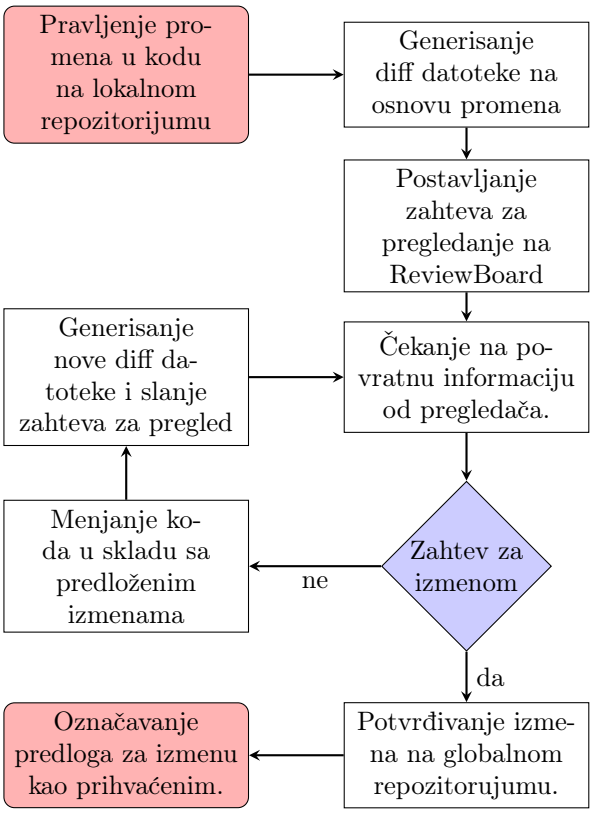
3.1 Alati otvorenog koda

- *Gerrit*
- *Codestriker*
- *ReviewBoard*
- *Gitlab*
- *Phabricator*

Gerrit : Gerrit Code Review je započet kao skup dodatnih mogućnosti na već ranije razvijen projekat pod nazivom *Rietveld* i prvobitna svrha mu je bila da služi projektu AOSP. [1] Kasnije je postao zaseban projekat sa novim, značajnijim mogućnostima koje je autor sistema *Rietveld*, Gvido van Rosum [2] odbijao da doda kako bi zadržao jednostavnost koda. U tom momentu je počela i značajna promena samog koda pa je bio potreban novi naziv. Odabrano je ime Gerrit, u čast holandskog arhitekte Gerita Ritvelda.
Verzija softvera Gerrit pod oznakom 2.X je bila značajna jer je izvorni kod napisan u Pajtonu reimplementiran uz pomoć programskog jezika Java. [1]
Ovaj softver je licenciran pod *Apache* licencom, ali postoje i vlasničke verzije softvera.

Codestriker : Codestriker je veb-aplikacija koja podržava on-lajn preglede koda. Moguće to učiniti na tradicionalan način, pregledom dokumentacije, ali takođe je podržan i pregled promena generisanih pomoću SCM [3] sistema.
Prva verzija ovog softvera je nastala u Decembru 2001 i objavljena je na *SourceForge* [4] platformi. Prvobitno, bio je implementiran kao ad hok rešenje u vidu manjeg skript programa napisanog u programskom jeziku Perl. Sve mogućnosti bi se mogle svesti u jednu rečenicu. *Prosledi svim potencijalnim pregledačima na mejl adresu izlaz komande iz programa koji služi kao posrednik u sistemu CVS i omogući pregledačima da mogu da ostavljaju komentare.*
Poslednje verzije ovog softvera omogućavaju i dalje ovakav manje formalan (*light-weight*) metod pregleda koda, ali podržava i potpuno formalan pristup.
Za razliku od 3.1, ovaj projekat je u svim svojim verzijama pod licencom koja spada u grupu onih koje zastupaju koncept softvera otvorenog koda,

[1] *Android Open Source Project*
[2] Guido van Rossum, poznatiji kao kreator programskog jezika Pajton
[3] Source Code Management
[4] Platforma na Internetu koja omogućava pristup softveru otvorenog koda.



Slika 2: Graf toka jednog procesa u radu sa alatom ReviewBoard

tačnije GPL [5].

ReviewBoard : Projekat ReviewBoard biva započet od strane dva programera, Kristejn Hemond (*Christain Hammond*) i David Troubridž (*David Trowbridge*), kompanije VMware. Oni su dobili zadatak da unaprede dotadašnji mehanizam pregledanja koda u timovima te kompanije. Koji se svodio na generisanje HTML koda koji je prikazivao staru i novu verziju koda i markirao delove koji se razlikuju. Članovi tima su imali mogućnost da dodaju i objašnjenja zašto su pravili te izmene i koje su sve testove sproveli nad novom verzijom koda. Nakon toga slao se zahtev potencijalnim pregledačima. Sve to je oduzimalo previše vremena, a dolazilo je i do gubitaka tih zahteva, pa je bilo neophodno unaprediti sve.
Izvorni kod ovog alata napisan je u programskom jeziku Pajton uz pomoć radnog okvira Django. Pod licencom je *Open Source MIT license*. [7]

[5]The GNU General Public License

4 Uticaji pregleda

Kako pregledi kôda utiču na programere koji zajedno rade u timu? Pokazalo se da dolazi do mnogih pozitivnih uticaja, ali i do nekih negativnih. [9] O ovim uticajima bi trebalo više govoriti – pozitivne pohvaliti, a na negativne posebno obratiti pažnju i predložiti načine da se oni reše. U ovom delu rada ćemo navesti i objasniti neke od njih.

4.1 Pozitivni uticaji

Programeri, znajući da će kolege iz tima pregledati njihov kôd, postaju pažljiviji. Svako želi da dobije što bolju kritiku i sigurno niko ne želi da bude onaj član tima koji uvek pravi neke početničke greške. Stoga će svako uložiti dodatni napor da sve proveri i ispoštuje pravila kodiranja. Ovaj uticaj se naziva „Ego efekat" (eng. *the "Ego Effect"*). [9]

Kod tehnike pregledanja, razmena znanja je obostrana. Iako se može pretpostaviti da u tom procesu uči samo programer čiji se kôd pregleda, nije tako. I pregledač može nešto novo naučiti gledajući kôd. Prvenstveno, iskusniji programeri pregledaju rad mlađih programera. Tada oni mogu uočiti neke stvari na koje, zbog navike, nisu obraćali pažnju, otkriti nove ideje i usvojiti nov način razmišljanja.

Pregledanje ne podstiče samo konverzaciju o kôdu, već i lični razvoj. Programer će saznati koje su to greške koje često ponavlja, a kojih možda nije ni bio svestan. Radiće na tome da ih ispravi. Vremenom će postati produktivniji i efikasniji, bez nekog pritiska, samo posmatrajući sebe. [9]

4.2 Negativni uticaji

Niko ne prihvata kritike najradije i svakome bude neprijatno kada mu se ukaže greška. Ipak, ljudi reaguju na kritike na različite načine. Neki ljudi to lakše podnesu, isprave grešku koju su napravili, okrenu je na šalu i tako prevaziđu situaciju. Drugi, posebno ako smatraju da su dali sve od sebe, kritike shvataju vrlo lično i povlače se u sebe. O tome treba voditi računa. Menadžeri moraju promovisati stav da su nedostaci pozitivni. Svaki od njih je prilika za poboljšanje kôda, a cilj postupka pregledanja je učiniti kôd što boljim. Cilj je eliminisanje što većeg broja oštećenja, bez obzira na to ko je prouzrokovao grešku. Pronalazak nedostatka ne znači „autor je napravio grešku i pregledač ju je pronašao", već znači da su autor i pregledač zajedno kao tim radili na poboljšanju proizvoda. [9] [8]

Pored lošeg prihvatanja kritike, negativni uticaj je i efekat „Velikog brata" (eng. *the "Big Brother" effect*). [9] Programer može steći utisak da ga neko stalno posmatra, pogotovo ako radi sa alatima za pregledanje. Zabeleženi i izmereni podaci su značajni za proces pregledanja, ali mogu izazvati loš efekat. Ako programer misli da će ti podaci biti iskorišćeni protiv njega, ne samo da će biti neprijateljski nastrojen prema procesu pregledanja, već će se fokusirati na poboljšanje svoje statistike umesto da zaista napiše bolji kôd. Menadžeri moraju biti svesni ovoga. Ako im izmereni podaci pomognu da otkriju neki problem, izdvajanje pojedinca će pre izazvati nove probleme nego što će rešiti tekući. Bolje je obratiti se grupi kao celini. Takođe, bolje je ne sazivati poseban sastanak u ovu svrhu, već samo taj problem uvrstiti u neki uobičajeni postupak. [8]

Važno je imati na umu i to da je teži kôd skloniji greškama. Vrši se detaljnije pregledanje i očekuje se da će biti dosta nedostataka, pa se često veliki broj istih više pripisuje složenosti kôda nego sposobnostima autora.

4.3 Agilni timovi i pregledi kôda

5 Saveti za dobro pregledanje

Literatura

[1] Internet lokacija Gerrit projekta. https://www.gerritcodereview.com/about.html.

[2] Zvanična internet adresa kompanije SmartBear. https://smartbear.com/product/collaborator/overview/.

[3] Zvanična internet adresa sa detaljnim informacijama. https://codescene.io/.

[4] Zvanična internet adresa sa detaljnim informacijama. https://www.veracode.com/.

[5] Zvanična internet adresa sa detaljnim informacijama. https://www.jarchitect.com/.

[6] Zvanična internet lokacija kompanije Atlassian. https://www.atlassian.com/software/crucible.

[7] Bosu, A. i Carver, J. Peer Code Review in Open Source Communities Using ReviewBoard. In *ACM 4th annual workshop on Evaluation and usability of programming languages and tools*, 2012.

[8] Jason Cohen. 11 proven practices for more effective, efficient peer code review. 2011. https://www.ibm.com/developerworks/rational/library/11-proven-practices-for-peer-review/.

[9] Teleki S. Cohen, J. and E. Brown. *Best Kept Secrets of Peer Code Review*. Smart Bear Inc., 2006.

[10] Philip M. Johnson. Does every inspection really need a meeting. *Empirical Software Engineering*, pages 9–35, 1998.

A Dodatak

Ovde pišem dodatne stvari, ukoliko za time ima potrebe. Ovde pišem dodatne stvari, ukoliko za time ima potrebe. Ovde pišem dodatne stvari, ukoliko za time ima potrebe. Ovde pišem dodatne stvari, ukoliko za time ima potrebe. Ovde pišem dodatne stvari, ukoliko za time ima potrebe.