Computer Science and Information Systems 0(0):1–6 DOI: N/A

Ekstremno programiranje - programiranje u paru

Ivan Stanković

Matematički fakultet,Studentski trg 16
11000 Belgrade, Serbia
istankvic986@gmail.com

Sadržaj

1. Uvod

Programiranje u paru spada u tehnike agilnog razvoja softvera u kojoj dva programera rade zajedno na jednom računaru. Jedan, koji se naziva “pilot”(vazač), piše kod, dok drugi, koji se naziva “kopilot” ili posmatrač, navigotor itd., koji procenjuje svaku liniju koda koju je pilot otkucao. Ova dva programera često menjaju uloge. Prilokom procene (reviewing), kopilot takođe uzima u obzir strategiju po kojoj će se dalje odvijati zadatak. On takođe razmišlja o tome kako da poboljša napisani kod i o problemima koji mogu da se jave prilikom daljeg razvoja. Na ovaj način izbegvamo opterećivanje pilota tim problemima tako da je pažnja pilota maksimalno fokusirana na tehničke aspekte završavanja trenutnog zadatka.

2. Načini uparivanja

Postoje više načina kako je moguće raditi u paru. Tri glavna načina su:

1.Expert – Expert uparivanje: Može se činiti kao očigledan način uparivanja koji će davati najbolje rezultate i imati najveću produktivnost, ali često daje mali uvid u nove načine za rešavanje problema, jer obe strane verovatno neće da preispituju ustanovljene tehnike koje dobro funkcionišu u praksi.

2.Expert – Početnik uparivanje: postoji mnogo mogućnosti za podučavanje početnika od strane experta. Ovo može dovesti do novih načina za rešavanje problema kao i novih ideja, pošto će početnik najverovatnije da dovede u pitanje ustanovljenje tehnike za rešavanje problema. Međutim,ovo ponekad dovodi do situacije “posmatraj mentora”(watch the master) ili do situacije da se početnik oseća suvišnim i da nema šta da doprinese rešenju.

3.Početnik – Početnik uparivanje: čini se kao najmanje optimalan način uparivanja za produktivnost i kvalitet, međutim rezultat često bude značajno bolji nego kad dva početnika rade samostalno. Ovakav način uparivanja se često može videti u akademskom okruženju.

2.1. Uparivanje ljudi na udaljenim lokacijama

Uparivanje ljudi na daljinu takođe poznato kao programiranje virtuelnog para ili distribuiranog para je programiranje u paru u kojoj su dva programera na različitim lokacijama koji rade pomoću editora za saradnju u realnom vremenu, deljenog desktopa(shared desktop). Kod uparivanja na daljinu su uoceni problemi koji se ne javljaju kada se radi licem u lice, kao što je dodatno kašnjenja zbog koordinacije, veća zavisnost od alata za praćenje zadataka i nedostatak verbalne komunikacije koja dovodi do zabune i konflikta oko toga ko upravlja tastaturom.

3. Prednosti

Ekonomičnost: Parovi utroše oko 15% više vremena na programe nego pojedinac. Međutim, dobijeni kod ima oko 15% manje nedostataka. Zajedno sa vremenom za razvijanje koda ostali faktori kao što su operativni troškovi i osiguranje kvaliteta takođe utiču na ukupne troškove. IBM je prijavio da je potrošio oko 250 miliona dolara na popravku i implementaciju ispravki za 30,000 problema koje su prijavili klijenti. Programiranje u paru značajno umanjuju ove izdatke tako što smanjuje broj nedostataka u programu.

Kvalitet dizajna softvera: Sistem uparenih programera poseduje veći potencijal za generisanje više različitih rešenja za probleme iz tri razloga: (1) programeri donese različita prethodna iskustva za rešavanje zadatka, (2) mogu da pristupe informacijama koje su bitne za rešavanje zadatka na različite načine, (3) imaju različite odnose prema problemu na osnovu njihovih funkcionalnih uloga. Programeri moraju otvoreno da pregovaraju o koracimo što se tiče zajedničkog plana kada nastane konflikt između njih. Radeći to, oni uzimaju u obzir više načina za rešavanje problema nego što bi to uradio jedan čovek sam. Ovo značajno poboljšava kvalitet dizajna programa jer smanjuje šansu za odabir loših metoda.

Zadovoljstvo: U anketi koja je sprovedena onjaln među programerima koji programiraju u paru, 96% njih se izjasnilo da više uživaju u njihovom poslu kada programiraju u paru nego kada programiraju sami. Dodatno,95% programera koji su anketirani su se izjasnili da su sigurniji u njihova rešenja kada programiraju u paru. Postoji korelacija između zadovoljstva među programerima i sigurnosti u kod koji su napisali, na primer, parovi programera više uživaju u njihovom poslu zato što su više sigurni u ono što su uradili.

Učenje: Znanje se konstantno razmenjuje među programerima koji su u paru, od raznih saveta kao što su pravila za određeni programski jezik do generalnog znanja o dizajniranju softvera. U "promiskuitetnom uparivanju", svaki programer komunicira i radi sa svim ostalim programerima koji su u timu pre nego da bude u paru samo sa jednim partnerom, što dovodi do toga da se znanje o sistemu koji se razvija širi na ceo tim. Programiranje u paru omogućuje programerima da ispitaju kod njihovih partnera i obezbeđuje povratne podatke koji su potrebni da bi se povećala njihova sopstvena sposobnost za njihovo samostalno učenje i razvijanje.

Izgradnja tima i međusobna komunikacija unutar tima: Programiranje u paru omogućuje članovima tima da podele probleme i rešenje brzo čineći manje verovatnim da članovi tima imaju skirvene namere jedni od drugih. Ovo pomaže programerima da lakše komuniciraju međusobno. Ovo povećava protok komunikacije kao i učestalost komunikacija u projektu, iz čega proizilazi veća prosečna komunikacija u timu.

4. Istraživanja

Empirijska istraživanja teže da ispitaju nivo produktivnosti i kvalitet koda. U daljem tekstu se navode neka empirijska istraživanja.

4.1. Williams: smanjenje grešaka povećanjem truda

Laurie Williams sa Univerziteta Juta u Solt lejk Sitiju je pokazala da su upareni programeri 15% sporiji od dva nezavisna pojedina programera, dok su "kodovi bez greške" povećani sa 70% na 85%. Sa obzirom da je testiranje i debagovanje često mnogo puta skuplje nego inicijalno programiranje, ovo je neverovatan rezultat. Parovi obično razmatraju više alternativa u dizajnu nego programeri koji rade sami, i prave dizajnove koji se lakše održavaju. Takođe, parovi programera ranije uočavaju nedostatke u dizajnu nego pojedinac.

Williams studija je pokazala napretke što se tiče korektnosti od oko 15% i od 20% - 40% smanjenje što se tiče vremena potrebno za razvoj. Međutim, u istraživanju je primećeno povećanje truda od 15%- 60%, koje je izmereno u utrošenim programerskim satima.

4.2. Lui: poređenje parova programera sa samostalnim programerom

Kim Man Lui je sproveo istraživanje koje upoređuje par programera tipa Početnik – Početnik sa programerom početnikom koji radi sam, i par programera tipa Expert – Expert sa programerom expertom koji radi sam. Istraživanje je pokazalo da par programera tipa Početnik – Početnik iskusio veće dobitke produktivnosti nego programer koji radi samostalno u odnosu na par programera tipa Expert – Expert i samostalni expert programer.

4.3. Arisholm: složeniji sistemi su više korektni,jednostavniji sistemi brži

Ovo istraživanje koje je sproveo Arisholm je pokazalo povećanje korektnosti od 48% za složenije sisteme, ali nije pokazalo značanje razlike u vremenu. Jednostavniji sistemi su pokazali 20% smanjenje u vremenu, ali nije bilo značajne promene u korektnosti. Sve u svemu, nije bilo smanjenja u vremenu ili porast u korektnosti, već povećanje od 84% u uloženom trudu.

4.4. Lui,Chan I Nosek: dizajn zadatka

Lui,Čan i Nosek su izvršili dva eksperimeta: (1) proceduralni zadaci (Procedural Solution Tasks) i (2) deduktivno rešavanje problem (Deductive Problem Solving). Od 2002 – 2003, 15 diplomaca treće godine su radili posao sa pola radnog vremena na politehničkom

univerzitetu u Honk Kong-u, i zaposleni programeri koji rade puno radno vreme su pozvani da popune upitnik sa više različitih odgovora na 15 pitanja vezanih za 5 algoritama za proceduralne zadatke. ”Analiza grafikona je osnovo znanje potrebno za dizajn programa”. Ključni procesi extremnog programiranja i programiranja u paru su rečeni studentima pre eksperimenta. U kontrolisanom eksperimentu, ispitanici su podeljeni u 5 programerskih parova i 5 pojedinačnih programera. Zbog “malog broja test uzoraka”, korišćen je test vilkonsonovih rang suma da “statistički proceni rezultate”. Parovi programera su nadmašili samostalne programere i u testu na kome se meri vreme i u testu broja povratnih odgovora sa verovatnoćom 0.03. Deset od petnaest ispitanika su se složili da učestvuju u narednom eksperimentu. U ovom eksperimentu, studenti su upitani da reše deduktivne probleme koji pomažu da se ispita programerova mogućnost da se snađe sa kompleksnim problemom. Ispitanici su prvo radili u parovima na jednom problemu, a zatim su radili pojedinačno na drugom problemu. Test vilkonsonovih rang suma je dokazao da su programerski parovi nadmašili samostalne programere imajuci u vidu i vreme koje je proteklo i broj povratnih odgovora sa verovatnoćom od 0.14 i 0.01.

4.5. John T.Nosek: kvalitet koda

Nosek je izveo eksperiment sa 15 sistemskih programera koji rade za firmu koja se bavi prodajom softvera i koji su zaposleni puno radno vreme. Deset programera je radilo upareno, a pet programera je radilo samostalno. ”Ispitanici su upitani da napišu skript koja proverava konzistentnost baze podataka, gde će izlazni podaci biti smešteni u fajl” i dao im je rok od 45 minuta za izvršavanje zadatka. Grupe su bodovane na osnovu čitljivosti i funkionalnosti koda koji su napisali. Parovi programera su osovjili 30% bolji rezultat što se tiče čitljivosti koda i 23% bolji rezultat za funkiconalnost napisanog koda. Ovi rezultati pružili dokaz da programiranje u paru poboljšava proces rešavanja problema.

4.6. Jerzy R.Nawrocki: programerski trud

Navrocki je izvršio eksperiment sa 15 studenata koji su na četvrtoj godini studija da proceni relativni trud kod programera koji rade u paru i programera koji rade samostalno. On je koristio koncept “Relativni trud koji su uložili parovi” za svoj eksperiment. Grupa je podeljenja na 5 parova programera i 5 samostalnih programera. Svi programeri su imali dve godine formalnog obrazovanja u programskim jezicima C i C++. Ispitanici su upitani da napišu četiri programa. To su bili programi za pronalaženje standardne devijacije uzoraka numeričkih podataka, pronalaženje parametara linearne regresije, prebrojavanje linija koda u programu i ukupan broj linija koda za ceo program. Njegov eksperiment je otkrio da samostalni programeri i programeri koji rade u paru utrošili istu količinu vremena da završe prva tri programa. Stoga,on je zaključio da programiranje u paru nije toliko efikasno kao što je navedeno u nekim studijama.

5. Zaključak

Programiranje u paru ima mnoge prednosti kao što su brže učenje i razmena znanja medju programerima koji rade u parovima, programeri imaju veće sampouzdanje i posao im je zanimljiviji,pišu kod sa manje grešaka koji je optimalniji i pouzdaniji nego kad ga razvija jedan programer samostalno.Neke studije su pokazale da je mnogo veća produktivnost programera u paru. Medjutim, u praksi se ne korsiti toliko često,jer je skuplje da se primenjuje ova tehnika.

Reference

1. "Pair Programming: Issues and Challenges"
2. "Wikipedia"
3. Laurie, W. (ed.): "Integrating Pair Programming into a Software Development Process"
4. Man, L.K. (ed.): "Pair programming productivity: Novice–novice vs. expert–expert"