1

Jasan kod

Ovu knjigu čitate iz dva razloga. Prvi razlog je što ste programer. Drugi, zato što želite da budete bolji programer. Odlično. Potrebni su nam bolji programeri.

Ovo je knjiga o dobrom programiranju. Prepuna je koda. Kod ćemo gledati iz svakog ugla. Gledaćemo na njega odozgo, odozdo na gore i iznutra ka spolja. Kad završimo, znaćemo mnogo o kodu. Štaviše, moći ćemo da utvrdimo razliku između dobrog i lošeg koda. Znaćemo kako da napišemo dobar kod. I znaćemo kako da loš kod pretvorimo u dobar kod.

Neka bude kod

Moglo bi da se tvrdi da je knjiga o kodu nekako zastarela – kod više nije problem, i da bi trebalo umesto toga da brinemo o modelima i zahtevima. Zaista, postoje mišljenja da smo blizu kraja rada sa kodom, jer će se, navodno, uskoro svi kodovi generisati a ne pisati. U tom slučaju programeri neće biti potrebni jer će poslovni ljudi generisati programe iz specifikacija.

Besmislica! Nikada se nećemo osloboditi od koda, jer kod predstavlja detalje zahteva. Na nekom nivou ti se detalji ne mogu zanemariti ili apstrahovati, moraju da budu specifikovani. Tako detaljno preciziranje da mašina može da ih izvrši naziva se *programiranje*. Ta specifikacija je *kod*.

Očekujem da će nivo apstrakcije naših jezika i dalje rasti. Takođe očekujem da će i broj namenskih jezika i dalje rasti. To će biti dobra stvar. Ali to neće eliminisati kod. Zaista, sve specifikacije napisane na tom višem nivou i namenskom jeziku biće kod! I dalje će morati da bude strog, tačan i toliko formalan i detaljan da ga mašina može razumeti i izvršiti.

Ljudi koji misle da će kod jednog dana nestati su poput matematičara koji se nadaju da će jednog dana otkriti matematiku koja ne mora biti formalna. Oni se nadaju da ćemo jednog dana otkriti način stvaranja mašina koje mogu da rade ono šta želimo, a ne ono šta kažemo. Ove mašine će morati tako dobro da nas razumeju tako da mogu da prevedu naše nejasne zahteve u savršeno izvršavanje programa koji tačno zadovoljavaju željene potrebe.

To se nikada neće dogoditi. Ni ljudi, sa svom intuicijom i kreativnošću, nisu u stanju da stvore uspešne sisteme na osnovu nejasnih osećanja korisnika. Zaista, ako nas je disciplina specifikacije zahteva išta naučila, onda je to da su tačno određeni zahtevi formalni kao i sam kod i mogu delovati kao izvršni testovi tog koda!

Zapamtite da je kod zaista jezik na kojem na kraju izražavamo zahteve. Možemo stvoriti jezike koji su bliži zahtevima ili možemo stvoriti alate koji će nam pomoći da analiziramo i sastavimo te zahteve u formalne strukture. Ali nikada nećemo eliminisati potrebnu preciznost – zato će uvek postojati kod.

Loš kod

Nedavno sam pročitao predgovor Kent Beckove knjige *Implementation Patterns*[1]. U njemu piše: „...ova knjiga je zasnovana na prilično krhkoj premisi: da je dobar kod važan... “ *Krhka* premisa? Ne slažem se! Mislim da je ta premisa jedna od najatraktivnijih,

1. [Beck07]

podržanih i preopterećenih od svih u našem poslu (i mislim da Kent to zna). Znamo da je dobar kod važan jer smo toliko dugo morali da se bavimo njegovim nedostacima.

Znam jednu kompaniju u kojoj je krajem 80-ih godina prošlog veka razvijena *strašna* aplikacija. Bila je veoma popularna, a mnogi profesionalci su je kupili i koristili. Ali ciklusi objavljivanja novih verzija aplikacije su počeli da se produžavaju. Greške nisu popravljene od jedne do druge verzije. Vreme učitavanja je raslo, a rušenja su postala učestala. Sećam se dana kad sam frustrirano zatvorio aplikaciju i nikad je više nisam koristio. Kompanija je ubrzo nakon toga prestala da postoji.

Dve decenije kasnije sreo sam jednog od nekadašnjih zaposlenih u toj kompaniji i pitao ga šta se dogodilo. Odgovor je potvrdio moje bojazni. Požurili su da izbace proizvod na tržište a imali su veliki nered u kodu. Dodavanjem sve više i više funkcija, kod je postajao sve lošiji i lošiji sve dok jednostavno nisu više mogli da imaju kontrolu nad njim. Loš kod je srušio kompaniju.

Da li *vam* je ikad jako zaglavio loš kod? Ako ste programer sa bilo kakvim iskustvom, onda ste to mnogo puta osetili. Postoji i ime za to. Mi to nazivamo *probijanje*. Probili smo se kroz loš kod. Probijamo se kroz mora zapetljanog trnja i skrivenih zamki. Borimo se da pronađemo svoj put, nadajući se nekom nagoveštaju, nekom tragu onome šta se dešava, ali sve što vidimo je sve besmisleniji kod.

Naravno da ste bili zaglavljeni u lošem kodu. Pa, zašto ste ga napisali?

Da li ste žurili? Verovatno je tako. Možda ste mislili da nemate vremena da dobro uradite posao, da će se vaš šef ljutiti na vas ako potrošite vreme da „ispeglate“ svoj kod. Možda ste se samo umorili od rada na tom programu i želeli ste da ga završite. Ili ste možda pogledali koliko zaostaju druge stvari koje ste obećali da ćete uraditi i shvatili da morate da „zbrzite“ ovaj modul da biste mogli da pređete na sledeći. Svi smo to radili.

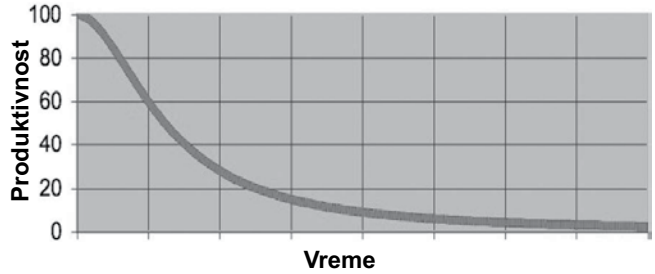
Svi smo videli nered koji smo napravili, a zatim smo odlučili da ga ostavimo za drugi dan. Svi smo osetili olakšanje kada smo videli kako naš neuredni program radi i odlučili da je nered koji radi bolji od ničega. Svi smo rekli da ćemo se vratiti i očistiti kasnije. Naravno, u one dane nismo poznavali LeBlancov zakon: Kasnije znači nikad.

Ukupan trošak nereda

Ako ste programer duže od dve ili tri godine, verovatno vas je usporio neuredan kod. Stepen usporavanja može biti značajan. Posle godinu ili dve, timovi koji su na početku projekta radili vrlo brzo, mogu se naći u situaciji da se kreću puževim korakom. Svaka promena koda utiče na dva ili tri druga dela koda. Nijedna promena nije beznačajna. Svako

dodavanje ili modifikacija sistema zahteva da se slojevi, zavoji i čvorovi „razumeju“ tako da se može dodati još više zapleta, zavoja i čvorova. S vremenom nered postaje toliko veliki i tako dubok i tako visok da ga ne mogu očistiti. Ne postoji način za to.

Kako nered raste, produktivnost tima nastavlja da pada, asimptotski se približavajući nuli. Kako se produktivnost smanjuje, menadžment čini jedino što može. Povećava broj saradnika u nadi da će povećati produktivnost. Ali ti novi saradnici nisu upućeni u dizajn sistema. Oni ne znaju razliku između promene koja odgovara dizajnerskoj nameri i promene koja sprečava dizajnersku nameru. Štaviše, oni i svi ostali u timu su pod strašnim pritiskom da povećaju produktivnost. Dakle – prave sve više i više nereda, usmeravajući produktivnost prema nuli. (Vidi sliku 1-1.)



Slika 1-1 Produktivnost u odnosu na vreme

Veliko redizajnirnje na pomolu

Na kraju, tim se pobuni i obaveštava menadžment da ne može da nastavi da razvija tu neprijatnu kolekciju kodova. Zahtevaju redizajn; menadžment ne želi da troši resurse na potpuno novi redizajn projekta, ali ne može da porekne da je produktivnost na užasno niskom nivou. Na kraju, rukovodioci se povinuju zahtevima programera i dozvoljavaju veliko redizajniranje.

Izabran je novi „opaki“ tim. Svi žele da budu u njemu jer se radi o projektu koji počinje od nule (engl. *Green-filed project*). Saradnici u timu počinju ispočetka i stvaraju nešto zaista lepo. Ali samo najbolji i najsjajniji su izbarani za opaki tim. Svi ostali moraju nastaviti da održavaju trenutni sistem.

Sada se radi o trci dve ekipe. Opaki tim mora da izgradi novi sistem koji radi sve što radi i stari sistem. I ne samo to, tim mora da bude u toku sa promenama koje se neprestano dešavaju u starom sistemu. Menadžment neće zameniti stari sistem sve dok novi sistem ne može da uradi sve ono što i stari sistem može.

Ova trka može da traje veoma dugo. Dešavalo se da traje i 10 godina. I do trenutka kada je to urađeno, prvobitni članovi opake ekipe su odavno otišli, a sadašnji članovi traže da se novi sistem redizajnira, jer je veliki nered.

Ako ste iskusili čak i jedan mali deo situacije koju sam upravo opisao, tada već znate da trošenje vremena na čišćenje koda nije samo isplativo, to je stvar profesionalnog opstanka.

Stav

Da li ste se ikada probijali kroz nered tako strašno da su vam bile potrebne nedelje da prođete ono što je trebalo da traje satima? Da li ste videli da ono što je trebalo da bude promena jedne linije koda, umesto toga stvori stotine različitih modula? Ove pojave su previše loše.

Zašto se to dešava kodu? Zašto se dobar kod tako brzo pretvori u loš kod? Postoji mnogo objašnjenja za to. Žalimo se da su se zahtevi promenili na takav način da remete originalni dizajn. Pozivamo se na rokove koji su bili tesni da bi se stvari odvijale kako treba. Blebetamo o glupim menadžerima, netolerantnim kupcima i beskorisnim marketinškim akcijama. Ali krivica, dragi Perice, nije u višoj sili, već u nama samima. Mi smo neprofesionalni.

Ovo je možda gorka pilula koju bi trebalo progutati. Kako bi ovaj nered mogao da bude naša greška? Šta je sa zahtevima? Šta je sa rasporedom? Šta je sa glupim menadžerima i beskorisnim marketingom? Zar oni ne snose neku krivicu?

Ne. Menadžeri i prodavci traže od *nas* informacije potrebne za davanje obećanja i obaveza; čak i kad nas ne pogledaju, ne bi trebalo da se stidimo da im kažemo šta mislimo. Korisnici traže potvrdu na koji će način zahtevi ući u sistem. Rukovodioci projekata zahtevaju od nas da pomognu u kreiranju rasporeda rada. U velikoj meri učestvujemo u planiranju projekta i delimo veliku odgovornost za eventualne propuste, pogotovo ako ti propusti imaju veze sa lošim kodom!

„Ali čekajte!“ reći ćete. „Ako ne učinim ono što moj šef kaže, biću otpušten.“ Verovatno nećete. Većina menadžera želi istinu, čak i kada se ne ponašaju tako. Većina menadžera želi dobar kod, čak i kada su pritisnuti rokovima. Oni mogu sa strašću braniti rokove i zahteve, ali to je njihov posao. *Vaš* posao je da sa istom strašću branite kod.

Da se vratim na poentu, da ste hirurg i da imate pacijenta koji zahteva da prestanete sa blesavim pranjem ruku u okviru priprema za operaciju, zato što to troši previše vremena?[2] Tada je pacijent šef, ali ipak doktor treba apsolutno da odbije taj zahtev. Zašto? Zato što lekar zna više od pacijenta o rizicima od bolesti i infekcija. Bilo bi neprofesionalno (a i kriminalno) da se lekar podređuje pacijentu.

Isto tako je neprofesionalno i da se programeri povinuju volji menadžera koji ne razumeju rizike stvaranja problema.

Početni problem

Programeri se suočavaju sa zagonetkom od suštinskog značaja. Svi programeri koji imaju više od nekoliko godina iskustva znaju da ih prethodni neredi u pisanju koda usporavaju. Pa ipak, svi programeri osećaju pritisak da prave nered da bi ispunili rokove. Ukratko, ne troše vreme da bi bili brzi!

Pravi profesionalci znaju da je drugi deo zagonetke pogrešan. *Nećete* ispuniti rok tako što ćete napraviti nered. Zaista, nered će vas odmah usporiti i naterati da propustite rok. *Jedini* način da se rok postigne – jedini način da se radi brzo – je da se kod u svakom trenutku održava što jasnijim.

2. Kada je 1847. godine Ignaz Semmelveis, prvi put preporučio lekarima pranje ruku, to je odbijeno na osnovu toga što su lekari bili previše zauzeti i nisu imali vremena da operu ruke.