Prevod drugog izdanja povodom 20-godišnjice prvog izdanja

Pragmatični programer

Vaš put do stručnosti

David Thomas
Andrew Hunt

kompjuter biblioteka

Pragmatični programer

Vaš put do stručnosti

Dave Thomas
Andy Hunt

Pearson
Addison-Wesley

kompjuter
biblioteka

Izdavač:

kompjuter biblioteka

Obalskih radnika 4a, Beograd

Tel: 011/2520272

e-mail: kombib@gmail.com

internet: www.kombib.rs

Urednik: Mihailo J. Šolajić

Za izdavača, direktor:
Mihailo J. Šolajić

Autori: Dave Thomas
Andy Hunt

Prevod: Slavica Prudkov

Lektura: Miloš Jevtović

Slog: Zvonko Aleksić

Znak Kompjuter biblioteke:
Miloš Milosavljević

Štampa: „Pekograf“, Zemun

Tiraž: 500

Godina izdanja: 2019.

Broj knjige: 523

Izdanje: Prvo

ISBN: 978-86-7310-546-8

The Pragmatic Programmer

Dave Thomas
Andy Hunt

ISBN 978-0-13-595705-9

Pohvale za drugo izdanje knjige „Pragmatični programer“

Neki kažu da su u knjizi „Pragmatični programer“ Andy i Dave ostvarili skoro neostvarivo; da je malo verovatno da će neko uskoro napisati knjigu koja može da pokrene celu industriju, kao što je ova. Međutim, ponekad, grom udara dva puta - ova knjiga je dokaz. Ažurirani sadržaj obezbeđuje da će knjiga ostati na vrhu liste „najboljih knjiga u softverskom razvoju“ u sledećih 20 godina, gde i pripada.

VM (Vicky) Brasseur,
direktor Open Source Strategyja, Juniper Networks

Ako želite da vaš softver može lako da se modernizuje i održava, neka vam je uvek u blizini kopija knjige „Pragmatični programer“. Knjiga je popunjena praktičnim savetima, i tehničkim i profesionalnim, koji će pomoći vama i vašim projektima u narednim godinama.

Andrea Goulet,
CEO, Corgibytes; Founder, LegacyCode.Rocks

„Pragmatični programer“ je knjiga za koju mogu da kažem da je potpuno promenila putanju moje karijere u softveru i usmerila me u pravcu uspeha. Otvorila mi je um ka mogućnostima da postanem zanatlija, a ne samo jedan od zubaca velike mašine. Ovo je jedna od najznačajnijih knjiga u mom životu.

Obie Fernandez,
autor knjige „The Rails Way“

Čitaoci koji prvi put čitaju ovu knjigu mogu očekivati uzbudljiv uvod u moderan svet softverske prakse, svet u čijem je oblikovanju prvo izdanje ove knjige igralo veliku ulogu. Čitaoci prvog izdanja će ovde ponovo otkriti uvide i praktične savete koji su ovu knjigu i učinile značajnom - ona sadrži stručna objašnjenja, sa mnogo novina.

David A. Black,
autor knjige „The Well-Grounded Rubyist“

Ja na svojoj polici imam staru papirnu kopiju originalne knjige „Pragmatični programer“. Ovu knjigu sam pročitao više puta i davno je promenila sve što se tiče načina na koji pristupam svom poslu kao programer. U novom izdanju promenjeno je sve i ništa: sada je čitam na svom iPadu i u primerima koda koriste se moderni programski jezici – međutim, osnovni koncept, ideje i stavovi su svevremenski i univerzalno primenljivi. Ova knjiga je relevantna kao i uvek u proteklih 20 godina. Srećna sam što znam da će sadašnji i budući programeri imati istu mogućnost da uče iz Andyjevih i Daveovih temeljnih uvida, kao što sam i ja nekada imala.

Sandy Mamoli, agilni trener,
autor knjige „How Self-Selection Lets People Excel“

Pre 20 godina, prvo izdanje knjige „Pragmatični programer“ u potpunosti je promenilo put moje karijere. Ovo novo izdanje može učiniti isto za vašu karijeru.

Mike Cohn,
autor knjiga „Succeeding with Agile“, „Agile Estimating and Planning“
i „User Stories Applied“

PREDGOVOR

Kada su Dave i Andy prvi put objavili tvit o novom izdanju ove knjige, to je bila velika novost. Gledao sam kako zajednica programera reaguje sa oduševljenjem, jer knjiga „Pragmatični programer" je relevantna i danas kao što je bila u prethodnih 20 godina.

Imao sam privilegiju da pročitam neobjavljenu kopiju da bih napisao ovaj predgovor i razumem zašto je pomenuti tvit izazvao veliko uzbuđenje. Iako je ovo tehnička knjiga, ne treba je takvom nazivati. Tehničke knjige su često zastrašujuće. Popunjene su „krupnim" rečima i nejasnim terminima i pretrpane su primerima koji, nenamerno, čine da se čitaoci osećaju pomalo glupi. Što je autor iskusniji, lakše zaboravlja kako je to kada se uče novi koncepti, kako je biti početnik.

Uprkos višedecenijskom iskustvu u programiranju, Dave i Andy su savladali težak izazov pisanja, sa istim uzbuđenjem kao i ljudi koji su naučili ove lekcije. Oni vas neće omalovažavati. Neće pretpostaviti da ste ekspert. Čak ne pretpostavljaju ni da ste pročitali prvo izdanje. Oni vas smatraju onakvim kakvi jeste - programeri koji samo žele da budu bolji. Mnoge stranice ove knjige su posvetili tome da vam pomognu, korak po korak.

Da budem iskren, oni su to već jednom uradili. Originalno izdanje je bilo prepuno korisnih primera, novih ideja i praktičnih saveta za izgradnju „mišića" za kodiranje i razvijanje „mozga" za kodiranje, koji važe i u današnje vreme. Međutim, ovo ažurirano izdanje uključuje dva poboljšanja.

Prvo je očigledno: uklonjeni su neke starije reference i zastareli primeri i zamenjeni su novim, modernijim sadržajem. Nećete pronaći primere invarijanti petlje ili izgradnje mašina. Dave i Andy su upotrebili svoj moćan sadržaj i uverili se da su lekcije i dalje aktuelne, bez ometajućih starih primera. Isključene su stare ideje, kao što je DRY (don't repeat yourself), i dodato im je malo svežine, što ih čini zaista sjajnim.

Drugo poboljšanje čini ovo izdanje zaista uzbudljivim. Naime, nakon pisanja prvog izdanja, autori su imali mogućnost da razmisle o onome što su pokušali da kažu, o onome što su želeli da čitaoci zapamte i kako je to bilo primljeno. Dobili su povratne informacije za te lekcije. Videli su gde su čitaoci „zapeli" i šta je pogrešno shvaćeno. U toku proteklih 20 godina, koliko je ova knjiga putovala kroz ruke i srca programera širom sveta, Dave i Andy su istražili povratne informacije i formulisali nove ideje, nove koncepte.

Otkrili su da programeri imaju nesumnjivo više agencija od bilo koje druge profesije. Započeli su ovu knjigu jednostavnom, ali dubokom porukom: „To je vaš život" - ona nas podseća na našu snagu u osnovi koda, u našim poslovima i u našim karijerama, što predstavlja osnov za sve ostalo u knjizi, a to je mnogo više od još jedne tehničke knjige popunjene primerima koda.

Ono što ovu knjigu zaista ističe na polici tehničkih knjiga je što se u njoj razume šta znači biti programer. Programiranje govori o pokušaju da se budućnost učini jednostavnijom. Govori o pojednostavljivanju nečega za članove vašeg tima, o greškama i mogućnosti njihovog ispravljanja, o formiranju dobrih navika i o razumevanju skupa alata. Kodiranje je samo deo sveta programera, a ova knjiga istražuje taj svet.

Mnogo vremena provodim razmišljajući o kodiranju. Ja nisam odrastao u svetu kodiranja; nisam to učio na koledžu. Nisam proveo tinejdžerske godine baveći se tehnikom. Ja sam kročio u svet kodiranja sredinom svojih dvadesetih godina i trebalo je da naučim šta znači biti programer. Zajednica programera se umnogome razlikuje od ostalih čiji sam deo bio. Postoji jedinstvena posvećenost učenju i vežbanju, što je i osvežavajuće i zastrašujuće.

Meni je to stvarno kao ulazak u novi svet, u novi grad. Trebalo je da upoznam susede, izaberem svoju prodavnicu i pronađem najbolje kafiće. Bilo je potrebno vreme da upoznam teritoriju i pronađem najefikasnije rute da bih izbegao ulice sa najgušćim saobraćajem i da znam kada je saobraćaj gust. Vreme je drugačije, bila mi je potrebna nova garderoba.

Prvih nekoliko nedelja, čak i meseci, u novom gradu je bilo strašno. Zar ne bi bilo lepo imati prijateljske, poznate susede koji već neko vreme žive ovde? Ko može da vam najbolje pokaže grad, da vam pokaže kafiće? To može samo neko ko je tamo već dovoljno dugo da poznaje kulturu i razume puls grada, tako da ne samo da se osećate kao kod kuće, već da postanete i član koji daje svoj doprinos tome gradu. Dave i Andy su takvi susedi.

Relativni novajlija veoma lako biva preplavljen ne samim činom programiranja, već i procesom postajanja programera. Potrebno je da dođe do potpune promene razmišljanja - promene navika, ponašanja i očekivanja. Proces postajanja boljeg programera ne dešava se samo zato što znate kako da kodirate; moraju postojati odlučna namera i stalne vežbe.

Ova knjiga je vodič kako da postanete bolji programer. Međutim, ne zavaravajte se – ona ne govori kako bi programiranje trebalo da bude. Nije filozofska na taj način. Ona govori, jasno i jednostavno, šta su pragmatični programeri - kako rade i kako pristupaju kodu. Vi treba da odlučite da li ćete da budete jedan od njih. Ako mislite da to nije za vas, niko vam neće zameriti, ali ako se opredelite da postanete pragmatični programer, autori će biti vaši prijateljski susedi koji će vam pokazati put.

Saron Yitbarek, osnivač i CEO of CodeNewbie

Host of Command Line Heroes

UVOD ZA 2. IZDANJE

Radili smo 1990-ih godina sa kompanijama čiji su projekti bili problematični. Svakoj smo sugerisali isto: možda bi trebalo da izvršite testiranje pre isporuke, zašto se kod gradi samo na jednoj istoj mašini, zašto niste pitali korisnike...

Da bismo uštedeli vreme sa novim klijentima, počeli smo da vodimo beleške, od kojih je nastala knjiga „*Pragmatični programer*". Na naše iznenađenje, knjiga je izgleda bila veoma korisna i bila je popularna svih narednih 20 godina.

Dve decenije je mnogo životnih vekova u pogledu softvera. Ako bismo programera iz 1999. godine ubacili u tim današnjih programera, sigurno bi se pomučio u ovom čudnom novom svetu. Svet iz 1990-ih godina je isto tako čudan za današnje programere. Reference u knjizi, kao što su CORBA, CASE alatke i indeksirane petlje, u najboljem slučaju su čudne, a verovatnije i zbunjujuće.

Proteklih 20 godina nije imalo nikakvog uticaja na zdrav razum. Tehnologija se možda promenila, ali ljudi nisu. Praksa i pristupi koji su tada predstavljali dobru ideju ostaju dobra ideja i danas. Ti aspekti knjige su i dalje vredni.

Kada je došlo vreme da se kreira ovo izdanje za *20. godišnjicu 1. izdanja*, trebalo je da donesemo odluku: mogli smo samo da ažuriramo tehnologije koje smo referencirali ili da preispitamo pretpostavke za praksu koju preporučujemo u svetlu dodatne dve decenije iskustva.

Na kraju, uradili smo i jedno i drugo.

Kao rezultat, ova knjiga je nešto kao *Theseusov brod*[1]. Otprilike jedna trećina tema u knjizi je potpuno nova. Ostatak je uglavnom prepisan, delimično ili u potpunosti. Naša namera je bila da teme učinimo jasnijim, relevantnijim i nadamo se, nekako svevremenskim.

1 Ako je tokom godina svaka komponenta broja zamenjena prilikom kvara, da li je rezultirajuće plovilo isti brod?

Doneli smo neke teške odluke. Izostavili smo Resources dodatak, jer bi bilo nemoguće održavati ga ažurnim i zato što je lakše pretraživati ono što želite. Preorganizovali smo i prepisali teme koje se odnose na konkurentnost, imajući u vidu aktuelno obilje paralelnog hardvera i nedostatak dobrih načina da se „nosimo“ sa njim. Dodali smo sadržaj da bismo reflektovali promenljive stavove i okruženja, od agilnog pokreta koji smo pomogli da se pokrene, do povećanog prihvatanja funkcionalnih programskih idioma i rastuće potrebe za razmatranje privatnosti i bezbednosti.

Interesantno je da je među nama bilo znatno manje rasprava o sadržaju ovog izdanja, nego u vreme kada smo pisali prvo izdanje. Obojica smo osećali da je lakše bilo identifikovati ono što je važno.

U svakom slučaju, uživajte u ovoj knjizi. Možda ćete prihvatiti neke nove načine rada. Možda ćete zaključiti da je nešto što mi preporučujemo pogrešno. Uključite se u svoj zanat. Pošaljite nam povratne informacije.

Najvažnije je da ne zaboravite da rad učinite zabavnim.

Kako je knjiga organizovana

Knjiga je napisana kao kolekcija kratkih tema - svaka je samostalna. Pronaći ćete mnoge unakrsne reference, koje će vam pomoći da svaku temu postavite u kontekst. Slobodno čitajte teme u bilo kom redosledu - ovo nije knjiga koju treba da čitate „bez preskoka“, od početka do kraja.

Povremeno ćete naići na okvir označen Savet nn (kao što je Savet 1, Brinite o svom zanatu, na stranici xxi). Kao što ističemo tačke u tekstu, smatramo da saveti imaju sopstveni život (živimo svakodnevno u skladu sa njima). Pronaći ćete rezime svih saveta na kartici unutar zadnje korice.

Uključili smo vežbe i izazove gde je to pogodno. Vežbe obično imaju relativno jasne odgovore, dok su izazovi otvoreniji. Da bismo vam pojasnili naše razmišljanje, uključili smo odgovore za vežbe u dodatku, ali veoma malo njih ima jedno tačno rešenje. Izazovi mogu formirati osnovu za grupne diskusije ili eseje na naprednim kursevima programiranja.

Takođe postoji kratka lista knjiga i članaka koje jasno referenciramo.

Šta to znači?

> *„Kada koristim reč", ističe prilično podrugljivim tonom Humpty Dumpty, „to znači samo ono što sam izabrao da znači - ni više, ni manje."*
>
> *Lewis Carroll, Through the Looking-Glass*

U knjizi ćete pronaći različite žargone - ili savršeno dobre reči koje su malo „oštećene" da bi značile nešto tehnički ili jezive izmišljene reči kojima su značenja dodelili računarski naučnici koji imaju otpor prema svom jeziku. Kada prvi put upotrebimo ove žargonske reči, pokušaćemo da ih definišemo, ili makar da vam damo neki nagoveštaj njihovog značenja. Međutim, sigurni smo da su neke od ovih reči odbačene, a druge, kao što su *objekat* i *relaciona baza podataka*, dovoljno su uobičajene, pa bi dodavanje definicije bilo dosadno. Ako naiđete na termin koji ranije niste videli, nemojte ga preskakati. Utrošite malo vremena da biste ga istražili, možda na Vebu ili u nekoj knjizi o računarskoj nauci. I, ako imate priliku, pošaljite nam e-mail i požalite se da bismo mogli da dodamo definiciju u sledećem izdanju.

Postoji, ponekad, savršeno dobra žargonska reč za koncepte, reči koje smo odlučili da ignorišemo. Zašto? Zato što je postojanje žargona normalno ograničeno na domen određenog problema ili za određenu fazu razvoja. Međutim, jedna od glavnih ideja ove knjige je da je većina tehnika, koje preporučujemo, univerzalna: na primer, modularnost se odnosi na kod, dizajn, dokumentaciju i organizaciju tima. Kada smo želeli da upotrebimo konvencionalnu žargonsku reč u širem kontekstu, ona je postala zbunjujuća - nismo mogli da prevaziđemo težinu originalnog termina. Kada se to desilo, odlučili smo da osmislimo sopstvene termine.

Izvorni kod i drugi resursi

Većina koda prikazana u ovoj knjizi je izdvojena iz kompilatorskih izvornih fajlova, dostupnih za preuzimanje sa veb sajta.[2]

Na ovom sajtu ćete takođe pronaći linkove ka resursima koje smatramo korisnim, zajedno sa ažuriranjima za knjigu, i novosti drugih pragmatičnih programera.

2 https://pragprog.com/titles/tpp20

Pošaljite nam povratne informacije

Bilo bi nam drago da čujemo vaše mišljenje. Pošaljite nam e-mail na adresu ppbook@pragprog.com.

Zahvalnice za 2. izdanje

Uživali smo bukvalno u hiljadama interesantnih razgovora o programiranju u proteklih 20 godina i upoznavali ljude na konferencijama, na kursevima i ponekad čak i u avionima. Svako od njih je dodao nešto našem razumevanju procesa razvoja i doprineo ažuriranju ovog izdanja. Hvala svima (i nastavite da nam ukazujete kada pogrešimo)!

Hvala učesnicima u beta procesu ove knjige! Vaša pitanja i komentari pomogli su nam da bolje objašnjavamo ono o čemu govorimo.

Pre nego što smo pokrenuli beta verziju, podelili smo knjigu nekolicini ljudi da bismo dobili komentare. Hvala VM (Vicky) Brasseur, Jeff Langr i Kim Shrier za detaljne komentare i José Valimui Nick Cuthbertu za tehnički pregled!

Hvala Ron Jeffriesu što nam je omogućio da upotrebimo Sudoku primer!

Mnogo se zahvaljujemo odgovornim ljudima u izdavačkoj kući „Pearson“ koji su nam odobrili da kreiramo ovu knjigu na naš način.

Posebno se zahvaljujemo nezamenljivoj Janet Furlow, koja savlada sve što preuzme i koja nas je držala usklađene.

I, na kraju, hvala svim pragmatičnim programerima koji su kreirali bolje programe u proteklih 20 godina!

UVOD ZA 1. IZDANJE

Iz uvoda u 1. izdanju

Ova knjiga će vam pomoći da postanete bolji programer.

Možete biti samostalni programer, član tima velikog projekta ili konsultant koji radi sa više klijenata odjednom - nije važno; ova knjiga će vam pomoći, kao pojedincu da bolje radite. Ova knjiga nije teoretska - koncentrišemo se na praktične teme i koristimo svoje iskustvo da biste bili informisaniji i lakše donosili odluke. Reč *pragmatičan* potiče od latinske reči *pragmaticus* - „vešt u poslu", što potiče od grčke reči πραγματικός, a znači „pogodan za upotrebu".

Ovo je knjiga o poslu.

Programiranje je zanat. Najjednostavnije rečeno, svodi se na to da učinite da računar uradi ono što vi želite da uradi (ili šta vaš korisnik želi da računar uradi). Kao programer, vi ste delom slušalac, delom savetnik, delom izvođač i delom diktator. Pokušavate da dokumentujete vaš rad tako da drugi mogu da ga razumeju i pokušavate da izgradite svoj rad tako da drugi mogu da ga dograde. Štaviše, nastojite da uradite sve to nasuprot sata projekta koji nemilosrdno otkucava. Vi pravite mala čuda svakog dana.

To je težak posao.

Postoji mnogo ljudi koji vam nude pomoć. Prodavci alata pričaju vam o čudima koja njihovi alati obavljaju. Gurui metodologije obećavaju da njihove tehnike garantuju odlične rezultate. Svi tvrde da je njihov programski jezik najbolji i svaki operativni sistem je odgovor na sve moguće „bolesti".

Naravno, ništa od toga nije istina. Ne postoje jednostavni odgovori. Ne postoji najbolje rešenje, bez obzira da li je reč o alatki, jeziku ili operativnom sistemu. Samo mogu postojati sistemi koji su pogodniji u određenom skupu okolnosti.

Tada „na scenu“ stupa pragmatizam. Ne bi trebalo da budete vezani ni za jednu određenu tehnologiju, već je potrebno da imate dovoljno široku pozadinu i iskustvo da biste mogli da izaberete dobra rešenja za određene situacije. Vaša pozadina potiče od razumevanja osnovnih principa kompjuterske nauke, a iskustvo potiče iz širokog raspona praktičnih projekata. Teorija i praksa kombinuju se i čine vas jakim.

Prilagodite svoj pristup da biste se uklopili u aktuelne okolnosti i okruženje. Procenite relativnu važnost svih faktora koji utiču na projekat i upotrebite iskustvo da biste kreirali odgovarajuća rešenja. I to radite kontinualno dok posao napreduje. Pragmatični programeri izvršavaju svoj posao, i to dobro.

Ko treba da pročita ovu knjigu?

Ova knjiga namenjena je ljudima koji žele da postanu efikasniji i produktivniji programeri. Možda se osećate frustrirano, jer vam se čini da ne dolazi do izražaja vaš potencijal. Možda gledate kolege koji koriste alatke da bi bili produktivniji od vas. Možda vaš aktuelni posao koristi starije tehnologije i želite da znate kako novije ideje mogu da se primene na ono što radite.

Mi se nećemo pretvarati da imamo sve odgovore, niti su sve naše ideje primenljive u svim situacijama. Ako pratite naš pristup, brzo ćete steći iskustvo, produktivnost će se povećati i bolje ćete razumeti ceo proces razvoja. I pisaćete bolji softver.

Šta čini pragmatičnog programera?

Svaki programer je jedinstven, sa individualnim snagama i slabostima, sklonostima i neslaganjima. Vremenom, svako će izgraditi sopstveno okruženje. To okruženje će odražavati individualnost programera podjednako snažno kao i njegovi hobiji, oblačenje i frizura. Međutim, ako ste pragmatičan programer, imaćete mnoge od sledećih karakteristika:

Rano usvajanje/brzo prilagođavanje

> Imate instinkt za tehnologije i tehniku i volite da isprobavate sve. Kada vam se pruži nešto novo, možete brzo to da razumete i integrišete u ostatak znanja. Vaše samopouzdanje proističe iz iskustva.

Radoznalost

> Volite da postavljate pitanja: kako ste to uradili, da li ste imali problema sa tom bibliotekom, šta je kvantno računarstvo, kako su implementirani simbolički linkovi... Vi sakupljate male činjenice koje će uticati na neke odluke u budućnosti.

Kritički mislilac

Retko prihvatate nešto, a da prvo ne saznate činjenice. Kada kolege kažu: „Zato što je to tako urađeno“, ili vam prodavac obeća rešenje za sve vaše probleme, vi ćete „namirisati“ izazov.

Realistični

Pokušavate da razumete osnovnu prirodu svakog problema sa kojim se suočavate. Ovaj realizam daje vam dobar osećaj koliko je nešto teško i koliko dugo će nešto trajati. Detaljno razumevanje da bi proces trebalo da bude težak ili da će potrajati duže dok se ne izvrši daje vam snage da nastavite.

Sveznalica

Trudite se da poznajete širok raspon tehnologija i okruženja i da budete „u toku“ sa novim dešavanjima. Iako vaš aktuelni posao može zahtevati da budete specijalista, uvek ćete moći da pređete na nove oblasti i nove izazove.

Najosnovnije karakteristike smo ostavili za kraj. Svi pragmatični programeri ih imaju. Ove karakteristike se mogu izraziti kao saveti:

Savet 1 Brinite o zanatu

Mislimo da nema smisla da razvijate softver ako vam nije stalo do toga da ga uradite dobro.

Savet 1 Razmišljajte o svom poslu

Da biste bili pragmatični programer, izazivamo vas da razmišljate o onome što radite. Nije reč o potrebi jednokratne revizije aktuelnih pristupa, već o neophodnosti stalne kritičke procene odluka koje donosite, svakog dana i u svakom projektu. Nikada nemojte raditi na „autopilotu“. Uvek razmišljajte i kritikujte svoj rad u realnom vremenu. Stari moto IBM korporacije *THINK*! je mantra pragmatičnog programera.

Ako vam ovo izgleda kao težak posao, pokazujete karakteristiku *realističnosti*. To će vam oduzeti dragoceno vreme. Nagrada je aktivnije bavljenje poslom koji volite, osećaj majstorstva nad sve većim rasponom tema i zadovoljstvo zbog stalnog napretka. Dugoročno će vam se isplatiti uloženo vreme, jer ćete vi i vaš tim postati efikasniji, pisaćete kod koji je lakši za održavanje i provodićete manje vremena na sastancima.

Individualni pragmatist, veliki timovi

Neki ljudi smatraju da u velikim timovima ili složenim projektima nema mesta za individualnost. Kažu da je softver inženjerska disciplina, koja se prekida ako pojedinačni članovi tima samostalno donose odluke.

Mi se sa tim ne slažemo.

Trebalo bi da postoji inženjerstvo u konstrukciji softvera. Međutim, to ne isključuje individualni rad. Pomislite na velike katedrale građene u Evropi tokom srednjeg veka. Za svaku je bilo potreban rad hiljada ljudi tokom mnogih decenija. Naučene lekcije su prosleđene sledećoj grupi građevinskih radnika, koji su svojim dostignućima unapredili građevinsko inženjerstvo. Međutim, stolari, kamenoresci, rezbari i staklari su bili zanatlije, koji su interpretirali zahteve inženjera da bi proizveli celine koje nadilaze čisto mehaničku stranu konstrukcije. Njihovo verovanje u sopstveni pojedinačni doprinos podržalo je projekte („*Mi koji samo sečemo kamenje uvek moramo da zamislimo katedrale*“).

U okviru celokupne strukture projekta uvek ima prostora za individualnost i zanatstvo. To je posebno tačno ako se ima u vidu aktuelno stanje softverskog inženjerstva. Sto godina od sada naše inženjerstvo možda će se činiti arhaično, kao što i današnjim građevinskim inženjerima izgledaju arhaične tehnike koje su koristili srednjovekovni graditelji katedrala, dok će naš zanat i dalje biti poštovan.

To je kontinualni proces

Turista koji je posetio Eton College u Engleskoj upitao je baštovana kako su travnjaci koje održava tako savršeni. „To je jednostavno", odgovorio je baštovan. „Očistite rosu svakog jutra, pokosite travu svakog drugog dana i valjate travnjak jednom nedeljno."

„To je sve?", upitao je turista. „Apsolutno", odgovorio je baštovan. „Radite to 500 godina, pa ćete i vi imati lep travnjak."

Dobrim travnjacima je potrebno malo nege svakog dana, kao i dobrim programerima. Savetnici menadžmenta vole da u razgovorima koriste reč kaizen – to je japanski termin koji obuhvata koncept kontinualnog postizanja mnogo sitnih poboljšanja. Smatra se jednim od glavnih razloga za dramatična poboljšanja produktivnosti i kvaliteta u japanskoj proizvodnji. Kaizen se primenjuje i na pojedince. Svakog dana radite da biste poboljšali veštine koje imate i da biste dodali nove alatke na vaš repertoar. Za razliku od Eton travnjaka, počećete da vidite rezultate za nekoliko dana. Za nekoliko godina bićete zadivljeni kako je vaše iskustvo obogaćeno i kako su unapređene vaše veštine.

Postanite član Kompjuter biblioteke

Kupovinom jedne naše knjige stekli ste pravo da postanete član Kompjuter biblioteke. Kao član možete da kupujete knjige u pretplati sa 40% popusta i učestvujete u akcijama kada ostvarujete popuste na sva naša izdanja. Potrebno je samo da se prijavite preko formulara na našem sajtu.
Link za prijavu: http://bit.ly/2TxekSa

Skenirajte QR kod
registrujte knjigu
i osvojite nagradu

POGLAVLJE 1

Pragmatična filozofija

Ova knjiga govori o vama.

Možete da postanete bolji programer i da pomognete drugima da postanu bolji. Možete da postanete pragmatični programer.

Po čemu se razlikuju pragmatični programeri? Mi smatramo da je suština u stavu, stilu, filozofiji pristupa problemima i njihovim rešenjima. Pragmatični programeri razmišljaju van okvira neposrednog problema, postavljajući ga u veći kontekst i tražeći širu sliku. Na kraju krajeva, kako možete da budete pragmatični bez ovog većeg konteksta? Kako možete da postižete inteligentne kompromise i donosite odluke na osnovu informacija?

Još jedan ključ uspeha pragmatičnih programera je što preuzimaju odgovornost za sve što rade, što ćemo opisati u odeljku „Mačka je pojela moj izvorni kod“. S obzirom da su odgovorni, pragmatični programeri neće sedeti mirno i gledati kako se njihovi projekti raspadaju zbog zanemarivanja. U odeljku „Entropija softvera“ opisaćemo kako možete da projekte zadržite u originalnom stanju.

Mnogim ljudima promene su teške, ponekad zbog dobrih razloga, ponekad zbog obične inercije. U odeljku „Supa od kamenja i kuvane žabe“ opisaćemo strategiju za podsticanje promena i (radi ravnoteže) predstavićemo vam priču o vodozemcu koji je ignorisao opasnost postepene promene.

Jedna od prednosti razumevanja konteksta u kojem radite je da postaje lakše da znate samo koliko dobar treba da bude softver. Ponekad je jedina opcija blizu savršenstva, ali često su uključeni i neki kompromisi. To ćemo istražiti u odeljku „Dovoljno dobar softver“.

Naravno, treba da imate dobru osnovu znanja i iskustvo da biste sve to iskoristili. Učenje je kontinualan i trajan proces. U odeljku „Portfolio vašeg znanja“ opisaćemo neke strategije za održavanje zaleta.

Na kraju, niko od nas ne radi u vakuumu. Svi provodimo mnogo vremena u interakciji sa drugima. U odeljku „Komunicirajte!“ izlistaćemo načine na koje može da se postigne ta interakcija.

Pragmatično programiranje proizilazi iz filozofije pragmatičnog razmišljanja. U ovom poglavlju postavićemo osnove za tu filozofiju.

1 To je vaš život

> *„Ja nisam na ovom svetu da bih ispunio vaša očekivanja i vi niste na ovom svetu da biste ispunili moja očekivanja“.*
>
> *Bruce Lee*

To je vaš život. Vi ga posedujete. Vi ga vodite. Vi ga kreirate.

Mnogi programeri sa kojima razgovaramo su frustrirani. Razlozi za njihovu zabrinutost su različiti. Neki smatraju da stagniraju u svom poslu, drugi da ih je tehnologija mimoišla, treći da su potcenjeni, ili nedovoljno plaćeni, ili da su njihovi timovi „toksični“. Možda žele da se presele u Aziju, ili Evropu ili da rade od kuće.

A naš odgovor je uvek isti.

Zašto ne možete to da promenite?

Razvoj softvera mora da se nalazi blizu vrha bilo koje liste karijera gde imate kontrolu. Naše veštine su tražene, naše znanje prevazilazi geografske granice, možemo da radimo sa udaljenosti. Dobro smo plaćeni. Zaista možemo da radimo sve što želimo.

Međutim, čini se da se programeri zbog nečega opiru promenama. Oni se samo pognu i nadaju se da će sve biti bolje. Pasivni su, jer njihove veštine postaju zastarele, a žale se da ih njihove kompanije ne obučavaju. Pregledaju oglase i traže egzotične lokacije za putovanje, zatim zakorače na hladnu kišu i krenu na posao.

Dakle, ovo je najvažniji savet u knjizi.

Savet 3 Imate agenciju

Da li je vaše radno okruženje neodgovarajuće? Da li vam je posao dosadan? Pokušajte to da ispravite. Međutim, nemojte samo pokušavati. Kao što je Martin Fowler rekao: „Možete da promenite vašu organizaciju ili promenite vašu organizaciju”.[1]

1 http://wiki.c2.com/?ChangeYourOrganization

Ako vam se čini da vas tehnologija „zaobilazi“, odvojite malo vremena da biste izučili novine koje izgledaju interesantne. Investirate u sebe, pa je jedino razumno da to radite van radnog vremena.

Želite da radite na daljinu? Da li ste se negde raspitali? Ako vam je odgovoreno odrečno, pronađite nekoga ko će vam dati potvrdan odgovor.

Softverska industrija obezbeđuje neverovatan skup mogućnosti. Budite proaktivni i iskoristite ih.

Povezani odeljci uključuju:

- Temu 4, „*Supa od kamenja i kuvane žabe*“, na stranici 8
- Temu 6, „*Portfolio vašeg znanja*“, na stranici 13

2 Mačka je pojela moj izvorni kod

Najveća slabost je strah da se pokažete slabi.

J.B. Bossuet, Politics from Holy Writ, 1709

Jedan od temelja pragmatične filozofije je ideja preuzimanja odgovornosti za sebe i svoje akcije, radi napretka u karijeri, učenja i obrazovanja, projekata i svakodnevnog posla. Pragmatični programeri su zaduženi za svoju karijeru i ne plaše se da priznaju greške. To sigurno nije najprijatniji aspekt programiranja, ali će se desiti - čak i u najboljim projektima. Uprkos detaljnom testiranju, dobroj dokumentaciji i solidnoj automatizaciji, može se javiti greška. Isporuke kasne. Javljaju se nepredviđeni tehnički problemi.

Ovakve nevolje se dešavaju i pokušavamo da ih rešimo što je moguće profesionalnije. To znači biti iskren i direktan. Možemo da budemo ponosni na svoje sposobnosti, ali se moramo suočiti i sa svojim nedostacima - neznanjem i greškama.

Poverenje tima

Pre svega, potrebno je da tim može da vam veruje i da se na vas osloni - i vi treba da možete da se oslonite na tim. Poverenje u timu je apsolutno potrebno za kreativnost i kolaboraciju u skladu sa literaturom istraživanja.[2] U zdravom okruženju, zasnovanom na poverenju, možete slobodno da izrazite svoje mišljenje, prezentujete svoje ideje i oslonite se na članove tima koji se, zauzvrat, mogu osloniti na vas.

2 Na primer, vidite dobru analizu na sajtu Trust and team performance: A meta-analysis of main effects, moderators, and covariates, http://dx.doi.org/10.1037/apl0000110

Zamislite visokotehnološki, tajanstveni nindža tim koji se infiltrira u jazbinu zlikovaca. Nakon više meseci planiranja i delikatnog izvršenja, infiltrirali ste na licu mesta. Sada je na vas red da postavite mrežu laserskog navođenja: „Žao mi je - ja nemam laser! Mačka se igrala crvenom tačkom i ostavio sam ga kod kuće."

Takvu povredu poverenja je možda teško ispraviti.

Preuzmite odgovornost

Odgovornost je nešto sa čime se aktivno slažete. Obavezujete se da ćete osigurati da će nešto biti dobro urađeno, ali ne morate nužno da imate direktnu kontrolu nad svakim aspektom izvršenja zadatka. Osim što dajete sve od sebe, treba da analizirate situaciju i otkrijete rizike koji su van vaše kontrole. Imate pravo da ne preuzmete odgovornost za nemoguću situaciju, ili situaciju u kojoj su rizici preveliki, ili su etičke implikacije previše nejasne. Treba da donesete odluku na osnovu sopstvenih vrednosti i prosuđivanja.

Kada prihvatite odgovornost za ishod, treba da očekujete da ćete se smatrati odgovornim za to. Kada pogrešite (jer svi grešimo) ili pogrešno nešto procenite, priznajte iskreno i pokušajte da ponudite neke opcije.

Nemojte da krivite nekoga ili nešto drugo, ili da pronalazite izgovore. Nemojte za sve kriviti prodavca, programski jezik, upravu ili saradnike. Neki od njih i svi oni će imati ulogu u grešci, ali vi treba da obezbedite rešenje, a ne izgovor.

Ako postoji rizik da vam prodavac ne izađe u susret, trebalo bi da imate plan za nepredviđene slučajeve. Ako se vaš medijum velikog kapaciteta topi, noseći i ceo izvorni kod sa sobom, a nemate rezervnu kopiju, to je vaša greška. Ako kažete svom šefu da je „maca pojela izvorni kod", to neće smanjiti štetu.

Savet 4 Obezbedite opcije, nemojte da izmišljate loše izgovore!

Pre nego što pristupite bilo kome da biste rekli zašto nešto ne može da se uradi, da će izvršenje kasniti ili je neuspešno, zaustavite se i poslušajte sebe. Govorite gumenoj patki na vašem monitoru, ili mački. Da li vaš izgovor zvuči smisleno ili glupo? Kako će to zvučati vašem šefu?

Preslušajte razgovor u vašoj glavi. Šta je ono što će verovatno druga osoba reći? Da li će upitati: „Da li si probao ovo..." ili „Da li si razmotrio ono?" Kako ćete odgovoriti? Pre nego što pođete i kažete lošu vest, da li postoji još nešto što biste mogli da isprobate? Ponekad ćete jednostavno znati šta će vam neko reći, pa ga nemojte ni uznemiravati.

Umesto izgovora, obezbedite opcije. Nemojte da kažete da ne može nešto da se uradi; objasnite šta može da se preduzme da bi bilo nađeno rešenje. Na primer, ako je opcija da kod treba da se izbriše, recite to i objasnite vrednost prerade koda (vidite Temu 40, „*Prerada*“, na stranici 209).

Treba li da odvojite vreme za kreiranje prototipa za određivanje najboljeg načina za nastavak rada (vidite Temu 13, „*Prototipi i post-it beleške*“, na stranici 56)? Treba li da uvedete bolje testiranje (vidite Temu 41, „*Test za kod*“, na stranici 214, i „*Nemilosrdno i kontinualno testiranje*“, na stranici 275) ili automatizaciju da biste sprečili da se problem ponovo javi?

Možda su vam potrebni dodatni resursi da biste izvršili ovaj zadatak. Ili možda treba da odvojite više vremena za susrete sa korisnicima? Ili je problem u vama: treba li da naučite neku tehniku ili tehnologiju detaljnije? Da li bi knjiga ili obuka pomogle? Nemojte se plašiti da priznate da vam je pomoć potrebna.

Pokušajte da isključite loše izgovore pre nego što ih naglas izgovorite. Ako morate, prvo recite svojoj mački.

Povezani odeljci uključuju

- Temu 49, „*Pragmatični timovi*“, na stranici 264

Izazovi

- Kako reagujete kada vam neko (bankarski službenik, automehaničar ili službenik) dolazi sa lošim izgovorom? Šta tada mislite o tim ljudima i njihovim kompanijama?
- Kada govorite „*Ne znam*“, obavezno nakon toga dodajte „*... ali ću saznati*“. To je odličan način da priznate da ne znate, a da, zatim, preuzimate odgovornost kao profesionalac.

3 Entropija softvera

Iako je razvoj softvera imun na skoro sve fizičke zakone, povećanje u *entropiji* nas teško pogađa. *Entropija* je termin iz fizike koji se odnosi na količinu „poremećaja" u sistemu. Nažalost, zakoni termodinamike ukazuju da entropija u univerzumu teži ka maksimumu. Kada se poremećaj poveća u softveru, to nazivamo „truljenje" softvera. Neki ljudi mogu ga nazivati optimističnijim terminom „tehnički dug", sa podrazumevanom idejom da ga jednog dana vrate. Verovatno ga neće vratiti.

Koji god naziv da koristimo, i „dug" i „truljenje" mogu nekontrolisano da se prošire.

Postoje mnogi faktori koji mogu doprineti „truljenju" softvera. Najvažnija je, čini se, psihologija, ili kultura, na poslu na projektu. Čak i ako ste jedini u timu, psihologija projekta može da bude veoma delikatna. Bez obzira na dobro osmišljene planove i najbolje ljude, projekat i dalje može da pretrpi propadanje u toku svog „životnog ciklusa". Ipak postoje i drugi projekti, koji se, bez obzira na prevelike poteškoće i konstantne zastoje, uspešno bore protiv prirodne sklonosti ka poremećaju i prilično dobro rešavaju ovaj problem.

Šta je bitno?

U gradovima su neke zgrade lepe i čiste, dok su druge trule. Zašto? Istraživači na polju kriminala i propadanja grada otkrili su fascinantan pokretački mehanizam, koji veoma brzo pretvara čistu, netaknutu, nastanjenu zgradu u razorenu i napuštenu ruinu.[3]

Polomljen prozor.

Jedan razbijen prozor, koji nije popravljen u dugom vremenskom periodu, u stanovnicima izaziva osećaj napuštenosti - da vlastima nije stalo do njihove zgrade. Pa se, zatim, polomi još jedan prozor. Ljudi počinju da bacaju smeće. Pojavljuju se grafiti. Započinje ozbiljno strukturalno oštećenje. U relativno kratkom vremenskom periodu zgrada postaje oštećena toliko da vlasnik ne želi da je popravi i napuštenost postaje stvarnost.

Zašto je to važno? Psiholozi su izvršili istraživanja[4] koja prikazuju da beznađe može biti zarazno. Razmislite o virusu gripa u bliskim četvrtima. Ignorisanje jasno loše situacije povećava ideje da možda ništa ne može da se ispravi, da nikoga nije briga, da je sve osuđeno na propast; sve negativne misli koje mogu da se prošire među članovima tima stvaraju začaranu spiralu.

3 Vidite *The police and neighborhood safety (WH82)*

4 Vidite *Contagious depression: Existance, specifičnost za simptome depresije i uloge traženja uverenja (Joi94)*

Savet 5 Nemojte živeti sa polomljenim prozorom

Ne ostavljajte polomljene prozore (loš dizajn, loše odluke ili loš kod) nepopravljene. Popravite ih što pre. Ako nemate dovoljno vremena da ih odgovarajuće popravite, onda ih *popravite privremeno*. Možda možete da komentarišete prekršeni kod, ili prikažete poruku „Nije implementirano" ili zamenite lažne podatke. Preduzmite *neke* akcije da biste sprečili dalju štetu i pokazali da se trudite da bude pronađeno rešenje za trenutnu situaciju.

Videli smo u nekim kompaniijama čiste, funkcionalne sisteme koji se prilično brzo pogoršavaju kada prozori počnu da pucaju. Postoje drugi faktori koji mogu izazvati „truljenje" softvera; zapostavljanje ubrzava „truljenje" više od bilo kog drugog faktora.

Možda mislite da niko nema vremena da se šeta okolo i čisti polomljeno staklo projekta. Ako je tako, onda bolje planirajte nabavku kontejnera ili potražite drugo susedstvo. Ne dozvolite da entropija pobedi.

Prvo, ne činite zlo

Andy je imao poznanika koji je bio veoma bogat - njegova kuća je bila besprekorna, popunjena neprocenjivim antikvitetima, *umetničkim delima* i tako dalje. Jednog dana, tapiserija koja je visila preblizu kaminu se zapalila. Vatrogasci su požurili u pomoć - da spasu njegovu kuću. Međutim, pre nego što su uvukli svoja velika, prljava creva za vodu u kuću, zaustavili su se uz vatru koja divlja da bi postavili prostirku između ulaznih vrata i izvora vatre.

Nisu želeli da uprljaju tepih.

Sada to zvuči prilično ekstremno. Sigurno je prvi prioritet vatrogasaca da ugase vatru, bez obzira na kolateralnu štetu. Međutim, oni su jasno procenili situaciju, bili su sigurni da mogu da ugase vatru i potrudili su se da ne izazovu nepotrebnu štetu na imovini. Tako treba postupati i za softver: ne izazivajte kolateralnu štetu samo zato što postoji kriza neke vrste. I jedan polomljeni prozor je previše.

Jedan polomljeni prozor (loše dizajniran deo koda ili loša odluka o upravljanju sa kojom tim treba da živi u toku trajanja projekta) je sve što je potrebno za početak pada. Ako se nađete u situaciji da radite na projektu sa nekoliko slomljenih prozora, veoma je lako početi razmišljati: „Ceo ostatak koda je loš, samo ću pratiti primer". Nije važno ako je projekat do sada bio dobar. U originalnom eksperimentu koji vodi do „teorije polomljenog prozora" napušteni automobil stajao je nedelju dana netaknut. Međutim, kada je jedan prozor razbijen, u roku od nekoliko sati automobil je bio rastavljen i prevrnut.

Isto tako, ako se nađete u projektu u kojem je kod izuzetno lep - jasno napisan, dobro dizajniran i elegantan, verovatno ćete se dodatno potruditi da ga ne uneredite, isto kao i vatrogasci. Čak i ako postoji vatra koja divlja (rok, datum izdavanja, demo prikaz na sajmu, itd), ne želite da budete prvi koji će napraviti nered i naneti dodatnu štetu.

Samo kažite sebi: „Nema polomljenih prozora“.

Povezani odeljci uključuju

- Temu 10, „*Ortogonalnost*“, na stranici 39
- Temu 40, „*Prerada*“, na stranici 209
- Temu 44, „*Imenovanje*“, na stranici 238

Izazovi

- Pomozite ojačavanje tima anketiranjem susedstva projekta. Izaberite dva ili tri polomljena prozora i razgovarajte sa kolegama o problemima koji su se javili.
- Možete li da kažete kada je prozor prvi put razbijen? Kakva je vaša reakcija? Ako je to rezultat nečije odluke, ili ukaza uprave, šta možete da uradite?

4 Supa od kamenja i kuvane žabe

Trojica vojnika koji su se vraćali kući iz rata bili su gladni. Kada su pred sobom videli jedno selo, raspoloženje im se popravilo - bili su uvereni da će im seljani ponuditi obrok. Međutim, kada su stigli u selo, naišli su na zaključana vrata i zatvorene prozore. Nakon mnogo godina ratovanja, seljani nisu imali dovoljno hrane i čuvali su ono što su imali.

Vojnici su prokuvali lonac vode i pažljivo postavili u njega tri kamena. Zadivljeni seljani su došli da gledaju.

„Ovo je supa od kamenja“, objasnili su vojnici. „Da li je to sve što stavljate u nju?“, upitali su seljani. „Apsolutno - mada neki kažu da je čak i ukusnija ako se doda nekoliko šargarepa...“. Jedan seljanin je otrčao i vratio se ubrzo sa korpom šargarepa iz svog skladišta.

Nekoliko minuta kasnije, seljani su ponovo upitali: „Da li je to sve?“

„Pa“, rekoše vojnici, „nekoliko krompira bi poboljšalo supu.“ Otrčao je drugi seljanin po krompir...

U toku sledećeg sata, vojnici su izlistali više sastojaka koji su poboljšali supu: govedinu, praziluk, so i začine. Svaki put bi drugi seljanin otrčao do svog ličnog skladišta.

Na kraju su skuvali veliki lonac supe. Vojnici su uklonili kamenje i seli su sa celim selom da uživaju u prvom finom obroku koji mesecima nisu jeli.

U priči o supi od kamenja postoji nekoliko pouka. Vojnici su prevarili seljane, iskoristivši njihovu radoznalost da bi od njih dobili hranu. Međutim, ono što je još važnije, vojnici deluju kao katalizatori, spajajući selo da bi mogli zajedno da kreiraju nešto što nisu mogli sami da urade – postignut je sinergistički rezultat. Na kraju svi pobeđuju.

S vremena na vreme, možda ćete želeti da oponašate te vojnike.

Možete se naći u situaciji u kojoj tačno znate šta treba da se uradi i kako da to uradite. Ceo sistem se prikazuje pred vašim očima - znate da je to u redu. Međutim, tražite dozvolu da rešite ceo problem i susrešćete se sa kašnjenjima i praznim pogledima. Ljudi će formirati odbore, za budžet se zahteva odobrenje i sve postaje komplikovano. Svako će voditi sopstvene resurse. Ponekad se to zove „početni zamor“.

Vreme je da izvadimo kamenje. Uradite ono što *možete* razumno da tražite. Dobro programirajte. Kada završite, pokažite ljudima i omogućite im da se dive. Zatim, recite: „Naravno, bilo bi bolje da smo dodali...“. Pravite se da to nije važno. Sedite i čekajte da ljudi sami zatraže da dodate funkcionalnost koju ste prvobitno želeli. Njima će biti lakše da se pridruže stalnom uspehu. Pokažite im tračak budućnosti i nateraćete ih da se okupe.[5]

Savet 6 Budite katalizator za promenu

5 Dok to radite, može da vas uteši linija pripisana kontraadmiralu dr Grace Hopperu: „Lakše je tražiti oproštaj, nego dobiti dozvolu“.’

Strana seljana

Sa druge strane, priča o supi od kamenja takođe govori o blagoj i postepenoj prevari. Govori o prejakom fokusiranju. Seljani su razmišljali o kamenju, a zaboravili su na ostatak sveta. Svakodnevno svi mi nasedamo na tako nešto.

Svi smo mi videli simptome. Projekti nam polako i neumoljivo izmiču. Većina Softverske katastrofe započinju najčešće kao previše male da bi se primetile, a većina prekoračenja u projektu se dešava svakodnevno. Sistemi odstupaju od svojih specifikacija funkciju po funkciju, dok se dodaje „zakrpa" za „zakrpom" u kod, sve dok ne ostane ništa od originala. Često akumulacija sitnica rastura moral i timove.

Savet 7 Zapamtite veliku sliku

Iskreno, ovo nikada nismo probali, ali, navodno, ako uzmete žabu i ubacite je u vrelu vodu, ona će iskočiti. Međutim, ako stavite žabu u hladnu vodu, koju, zatim, postepeno zagrevate, žaba neće primetiti sporo povećanje temperature i ostaće mirna dok se ne skuva.

Imajte na umu da je problem žabe drugačiji od problema razbijenog prozora koji smo opisali u Temi 3, „*Entropija softvera*", na stranici 6. U odeljku „Teorija razbijenog prozora" ljudi gube volju da se bore protiv entropije, jer smatraju da nikoga nije briga – jednostavno, žaba ne primećuje promenu.

Nemojte da budete kao žaba. Pratite veliku sliku. Stalno pregledajte ono što se oko vas dešava, a ne samo ono što vi lično radite.

Povezani odeljci uključuju

- Temu 1, „*To je vaš život*", na stranici 2
- Temu 38, „*Programiranje po slučajnosti*", na stranici 197

Izazovi

- Tokom pregleda nacrta prvog izdanja John Lakos je pitao: „Vojnici progresivno obmanjuju seljane, ali promena koju sprovode svima čini dobro. Međutim, progresivnim obmanjivanjem žabe nanosite štetu. Možete li da odredite da li pravite supu od kamenja ili supu od žabe kada pokušavate da sprovedete promenu? Da li je odluka subjektivna ili objektivna?"
- Brzo, bez gledanja, odgovorite koliko se sijalica nalazi na plafonu iznad vas? Koliko je izlaza u prostoriji? Koliko je ljudi? Postoji li nešto van konteksta, nešto što izgleda da tu ne pripada? Ovo je vežba za svesnost

situacije, tehnika koju sprovode ljudi, od dečaka i devojčica izviđača, do pripadnika ratne mornarice.

Neka vam pređe u naviku da gledate i primećujete svoje okruženje. Uradite isto za projekat.

5 Dovoljno dobar softver

Težeći za boljim, često izostavljamo ono što je dobro.

Shakespeare, Kralj Lir 1.4

Postoji stari vic o kompaniji koja je naručila 100.000 IC-ja od japanskog proizvođača. Deo specifikacije bila je i stopa oštećenja: jedan čip na 10.000. Nekoliko nedelja kasnije stigla je poručena roba: jedna velika kutija sa 99.990 IC-ja i mala sa samo 10. Na maloj kutiji bila je nalepnica na kojoj je pisalo: „Ovo su oštećeni IC-ji".

Kada bismo bar imali ovu vrstu kontrole nad kvalitetom... Međutim, stvarni svet nam jednostavno ne dozvoljava da proizvedemo mnogo toga što je zaista savršeno, posebno ne softver bez greške. Vreme, tehnologija i temperament su protiv nas.

Međutim, to ne mora biti frustrirajuće. Kao što je Ed Yourdon opisao u članku u *IEEE Software, When good-enough software is best [You95]*, možete se disciplinovati za pisanje softvera koji je dovoljno dobar za vaše korisnike, za buduće administratore i za vaš mir. Otkrićete da ste produktivniji i da su vaši korisnici srećniji. I možda ćete otkriti da su vaši programi, u stvari, bolji zbog kraće inkubacije.

Pre nego što nastavimo, treba da kvalifikujemo ono što ćemo reći. Fraza „dovoljno dobar" ne podrazumeva neispravan ili loše napisan kod. Svi sistemi moraju da ispune zahteve svojih korisnika da bi bili uspešni i da zadovolje osnovne performanse, privatnost i standarde bezbednosti. Jednostavno se zalažemo za to da se korisnicima pruži mogućnost da učestvuju u procesu donošenja odluka kada je ono što ste proizveli dovoljno dobro za njihove potrebe.

Uključite korisnike u kompromise

Normalno, vi pišete softver za druge ljude. Ovo je vežba za svesnost situacije, tehnika koju sprovode ljudi, od dečaka i devojčica izviđača, do pripadnika ratne mornarice.[6] Međutim, da li ih ikada upitate koliko dobar softver žele? Ponekad ne postoji izbor. Ako radite na pejsmejkerima, autopilotu ili biblioteci niskog nivoa koja će biti široko rasprostranjena, zahtevi će morati da budu stroži, a vaše opcije ograničenije.

6 Ovo je trebalo da bude šala!

Međutim, ako radite na potpuno novim proizvodima, imaćete drugačija ograničenja. Ljudi koji se bave marketingom treba da održe obećanje, krajnji korisnici su možda već napravili planove zasnovane na rasporedu isporuke, a vaša kompanija će sigurno imati ograničenja u novčanom toku. Bilo bi vrlo neprofesionalno ignorisati ove zahteve korisnika za dodavanje nove funkcije u program ili za još jedan pregled koda. Isto tako je neprofesionalno obećati nemoguće vremenske rokove i iseći neke osnovne inženjerske poslove da bi bili ispoštovani rokovi.

Opseg i kvalitet sistema koji proizvodite trebalo bi da budu opisani kao deo sistemskih zahteva.

Savet 8 Neka kvalitet bude pitanje zahteva

Često ćete biti u situacijama u kojima su uključeni kompromisi. Iznenađujuće je da mnogi korisnici radije u *današnje* vreme koriste softver sa nekim grubim ivicama, nego da čekaju godinu dana da se pojavi sjajna verzija sa svim funkcijama (a ono što će biti potrebno za godinu dana može se u potpunosti razlikovati). Mnoga IT odeljenja sa slažu da korisnici radije koriste softver koji nije profinjen. Dobar softver danas je često poželjniji od savršenog softvera sutra. Ako vašim korisnicima ranije date nešto da se poigraju, njihove povratne informacije će često dovesti do boljeg krajnjeg rešenja (vidite Temu 12, „*Oznake tragača*“, na stranici 50).

Znajte kada da se zaustavite

Na neke načine programiranje je kao slikanje. Započećete sa praznim platnom i određenim osnovnim sirovim materijalima. Da biste odredili šta sa njima možete da uradite, upotrebićete kombinaciju nauke, umetnosti i zanata. Skiciraćete oblik i odslikati osnovno okruženje, koje ćete, zatim, popuniti detaljima. Konstantno ćete se vraćati i kritički pregledati ono što ste uradili. S vremena na vreme ćete baciti platno i početi iznova.

Međutim, umetnici će vam reći da će sav težak posao biti uništen ako ne znate kada treba da se zaustavite. Ako dodajete sloj na sloj, detalj preko detalja, *slika postaje izgubljena u bojama*.

Nemojte da uništite dobar program preteranim ulepšavanjem. Nastavite i ostavite da kod malo „odstoji“. Možda nije savršen. Ne brinite: nikada ne može biti savršen (u Poglavlju 7, „*U toku kodiranja*“, biće reči o filozofijama za razvoj koda u nesavršenom svetu).

Povezani odeljci uključuju

- Temu 45, „*Zahtevi*“, na stranici 244
- Temu 46, „*Rešavanje nemogućih slagalica*“, na stranici 252

Izazovi

- Pogledajte softverske alatke i operativne sisteme koje redovno koristite. Možete li da pronađete neki dokaz da su organizacije koje isporučuju softver i/ili programeri isporučili softver za koji znaju da nije savršen? Kao korisnik, da li biste radije čekali da isprave sve greške, imali složen softver i prihvatili neke greške ili biste izabrali jednostavniji softver sa manje grešaka?
- Razmotrite efekat modularizacije na isporuku softvera. Da li će biti potrebno više ili manje vremena da se kreira čvrsto povezan monolitni blok softvera potrebnog kvaliteta u poređenju sa sistemom koji je dizajniran kao veoma labavo povezani moduli ili mikroservisi? Koje su prednosti i mane svakog pristupa?
- Možete li da zamislite popularni softver koji ima višak funkcija, odnosno, softver koji sadrži mnogo više funkcija nego što ćete ikada upotrebiti, a svaka funkcija predstavlja više mogućnosti za greške i bezbednosnu ranjivost i čini da teže pronađete funkcije koje koristite i upravljate njima. Da li ste u opasnosti da upadnete i sami u zamku da kreirate softver sa viškom funkcija?

6 Portfolio vašeg znanja

Ulaganje u znanje uvek plaća najbolju kamatu.

Benjamin Franklin

Ah, dobri stari Ben Franklin - nikada nije bio na gubitku zbog jezgrovite propovedi. Ako bismo mogli rano da legnemo u krevet i rano da se probudimo, bili bismo odlični programeri - zar ne? Ptica ranoranilica može dobiti crva ranoranioca, ali šta se dešava sa crvom ranoraniocem?

Međutim, u ovom slučaju Franklin se zaista zapitao šta se dešava sa crvom. Vaše znanje i iskustvo su najvažnije svakodnevno profesionalno bogatstvo.

Nažalost, to *bogatstvo se troši*.[7] Vaše znanje postaje zastarelo kada se razvijaju nove tehnike, jezici i okruženja. Promena tržišta može vaše iskustvo učiniti zastarelim ili nevažnim. S obzirom na sve brži tempo promena u tehnološkom društvu, to može da se desi prilično brzo.

Dok se smanjuje vrednost vašeg znanja smanjuje se i vrednost vaše kompanije ili klijenta. Mi želimo da sprečimo da se to desi.

Vaša mogućnost da naučite novine je najvažnija strategija. Kako da naučite kako da učite i kako da znate šta da učite?

Portfolio vašeg znanja

Mi mislimo na sve činjenice koje programeri znaju o računarstvu, domenima aplikacija u kojima rade, svom iskustvu i svom portfoliju znanja. Upravljanje *portfolijom znanja* je veoma slično upravljanju portfolijom finansija:

1. Ozbiljni investitori vrše investicije redovno – to im je navika.
2. Diverzifikacija je ključ dugoročnog uspeha.
3. Pametni investitori uravnotežuju svoje portfolije između konzervativnih i visokorizičnih investicija sa velikim nagradama.
4. Investitori pokušavaju da kupuju jeftino i prodaju skupo, uz maksimalan dobitak.
5. Portfolije treba periodično pregledati i ponovo uravnotežiti.

Da biste imali uspešnu karijeru, morate da investirate u svoj portfolio znanja, koristeći iste ove smernice.

Dobra vest je da je upravljanje ovom vrstom investicija veština isto kao i bilo koja druga - može da se nauči. Trik je da naterate sebe da investirate na početku i steknete naviku. Razvijte rutinu koju ćete pratiti dok je mozak ne internalizuje - tada ćete automatski „upijati“ nova znanja.

Izgradnja portfolija

Investirajte redovno

> Kao i za finansijske investicije, morate redovno da investirate u portfolio znanja, čak i ako je to samo mala količina znanja. Navika je isto toliko važna kao i suma, pa planirajte da upotrebite konzistentno vreme i mesto. Nekoliko jednostavnih ciljeva je izlistano u sledećem odeljku.

7 Bogatsvo koje se troši je nešto čija vrednost nestaje vremenom. Primeri uključuju skladište puno banana i ulaznicu za fudbalsku utakmicu.

Raznolikost

Što je vaše znanje raznolikije, vredniji ste. Osnovno je da poznajete karakteristike određene tehnologije koju trenutno koristite. Međutim, nemojte se tu zaustavljati. Računarstvo se brzo menja - najnovija tehnologija danas možda će sutra biti skoro beskorisna (ili nepotrebna). Što više tehnologija poznajete, bolje ćete se prilagođavati promenama. I ne zaboravite sve ostale veštine koje su vam potrebne, uključujući i one iz netehničkih oblasti.

Upravljanje rizikom

Tehnologija postoji duž spektra, od rizičnih sa potencijalno visokim nagradama, do niskorizičnih standarda sa malim nagradama. Nije dobra ideja da investirate sav svoj novac u visokorizične akcije koje iznenada mogu da se sruše, niti bi trebalo da investirate sav novac konzervativno i propustite potencijalne mogućnosti. Nemojte staviti sva tehnička jaja u jednu korpu.

Kupujte jeftino, prodajte skupo

Učenje novih tehnologija pre nego što postanu popularne može da bude isto toliko teško kao i pronalaženje potcenjenih akcija, ali nagrada može biti podjednako dobra. Učenje Jave kada je prvi put predstavljena možda je bilo rizično u to vreme, ali za one koji su je rano naučili prilično se brzo isplatilo kada je postala oslonac u industriji.

Pregled i ponovno uravnoteženje

Softverska industrija je veoma dinamična. Tehnologija koju ste počeli da istražujete prošlog meseca možda je do sada već zaboravljena. Možda treba da isključite tehnologiju baze podataka koju niste neko vreme koristili. Ili možda možete da budete bolje pozicionirani za novi posao ako isprobate neki drugi jezik.

Od svih ovih smernica, najvažnija je i najjednostavnija:

Savet 9 Investirajte redovno u portfolio znanja

Ciljevi

Sada, kada imate neke smernice šta i kada da dodate portfoliju znanja, koji je najbolji način da steknete intelektualni kapital pomoću kojeg ćete finansirati svoj portfolio? Evo nekoliko predloga:

Naučite najmanje jedan novi jezik svake godine

Različiti jezici rešavaju iste probleme na različite načine. Učenjem nekoliko različitih pristupa možete proširiti vaša razmišljanja i izbeći da ostanete

zaglavljeni na jednom mestu. Osim toga, učenje više jezika je lako, zahvaljujući ogromnoj količini dostupnog softvera.

Pročitajte po jednu tehničku knjigu svakog meseca

Iako na Vebu postoje mnogobrojni kratki eseji i povremeno pouzdani odgovori, za bolje razumevanje treba vam obimna knjiga. Potražite tehničke knjige o interesantnim temama koje se odnose na vaš aktuelni projekat.[8] Kada vam to pređe u naviku, pročitajte jednu knjigu mesečno. Nakon što savladate tehnologije koje trenutno koristite, proširite se i naučite nešto što se ne odnosi na projekat.

Čitajte i netehničke knjige

Važno je da imate na umu da računare koriste ljudi čije potrebe pokušavate da zadovoljite. Vi radite sa ljudima, zapošljavaju vas ljudi i hakuju vas ljudi. Nemojte zaboraviti ljudsku stranu jednačine, jer to zahteva potpuno različit skup veština (mi ironično ove veštine nazivamo lakim, ali ih je, u stvari, veoma teško naučiti).

Idite na časove

Potražite neke interesantne kurseve na lokalnom ili online koledžu ili univerzitetu, ili možda na sledećem sajmu ili konferenciji.

Učestvujte u lokalnim korisničkim grupama ili na sastancima

Izolacija može biti veoma loša za vašu karijeru; otkrijte šta rade ljudi van vaše kompanije. Nemojte samo da idete na sastanak i slušate, već aktivno učestvujte.

Eksperimentišite sa različitim okruženjima

Ako ste koristili samo Windows, odvojite malo vremena za Linux. Ako ste koristili samo makefiles i editor, isprobajte sofisticiraniji IDE koji sadrži moderne funkcije.

Ostanite „u toku“

Pročitajte novosti i postove online o tehnologijama koje se razlikuju od aktuelnog projekta. To je odličan način da otkrijete kakva iskustva drugi ljudi imaju sa tehnologijom, koji konkretan žargon koriste i tako dalje.

Važno je da nastavite da investirate. Kada naučite neki novi jezik ili tehnologiju, nastavite. Naučite još jedan jezik ili jednu tehnologiju.

Nije važno da li ćete ikada upotrebiti bilo koju od ovih tehnologija u projektu ili čak da li ćete ih uneti u rezime. Proces učenja će proširiti vaša razmišljanja, otvoriće vam nove mogućnosti i predstaviće nove načine rada. Važno je unakrsno ispitivanje ideja; pokušajte da primenite lekcije koje ste naučili u aktuelnom projektu.

8 Možda smo pristrasni, ali odlična selekcija je dostupna na adresi https://pragprog.com.

Čak i ako projekat ne koristi konkretnu tehnologiju, možda možete da pozajmite neke ideje. Upoznajte, na primer, orijentaciju objekta i pokušajte da napišete drugačije proceduralne programe. Razjasnite paradigmu funkcionalnog programiranja i pisaćete objektno-orijentisani kod na drugi način.

Mogućnosti za učenje

Čitate mnogo, obavešteni ste o svim najnovijim dostignućima u svojoj oblasti (što nije lako) i neko vam postavi pitanje, a vi nemate nikakvu ideju šta da odgovorite, i to slobodno priznate.

Nemojte da se tu zaustavite. Otkrivanje odgovora smatrajte ličnim izazovom. Raspitajte se. Pretražite Veb - i u delu nauke, a ne samo u delovima korisnika.

Ako ne možete sami da pronađete odgovor, saznajte ko može. Nemojte odustajati. Razgovarajte sa drugim ljudima, jer će vam to pomoći da izgradite ličnu mrežu, i možda ćete se iznenaditi otkrivanjem rešenja drugih, nepovezanih problema na tom putu. I stari portfolio se samo povećava...

Za ovo čitanje i istraživanje je potrebno vreme, a vremena je već malo. Stoga, treba da planirate unapred. Uvek imajte nešto za čitanje. I dok čekate pregled kod lekara ili zubara, možete nešto pročitate, ali obavezno ponesite svoj e-čitač, ili ćete biti primorani da čitate stari članak iz 1973. godine o Papua Novoj Gvineji.

Kritičko razmišljanje

Razmišljate *kritički* o onome što pročitate ili čujete. Treba da se uverite da je znanje u vašem portfoliju tačno i bez uticaja prodavca ili medija. Čuvajte se fanatika koji insistiraju da je njihova dogma jedini odgovor – može, ali ne mora da bude primenljiva za vas ili vaš projekat.

Nikada nemojte da potcenite moć komercijalizma. Samo zato što veb pretraživač lista stavku na prvom mestu ne znači da je i najbolje poklapanje; provajder sadržaja može da plati da bi bio postavljen na prvo mesto. Samo zato što knjižara postavi knjigu na dobro vidljivo mesto ne znači da je knjiga dobra, ili čak popularna; možda je plaćeno da se tamo postavi.

Savet 10 Kritički analizirajte ono što pročitate ili čujete

Kritičko razmišljanje je disciplina samo po sebi i savetujemo vas da pročitate i ispitate sve što možete o njemu. U međuvremenu, evo za početak nekoliko pitanja koja treba da postavite i da o njima razmislite.

Pet pitanja „zašto“

> Omiljeni konsultantski trik: upitajte „zašto“ najmanje pet puta i nastojte da pronađite odgovor. Ponavljajte kao da ste četvorogodišnjak (ali pristojan). Možda ćete na ovaj način moći da se približite uzroku.

Kome je ovo od koristi?

> Možda zvuči cinično, ali praćenje novca može da bude veoma korisna putanja za analizu. Prednosti za nekog drugog čoveka ili drugu organizaciju mogu se uskladiti sa vašim.

Šta je kontekst?

> Sve se dešava u sopstvenom kontekstu i zbog toga često ne postoje rešenja „jedna veličina odgovara svima“. Pronađite članak ili knjigu o „najboljoj praksi“. Dobro pitanje koje treba razmotriti je „za koga je najbolja?“. Koji su preduslovi i kakve su posledice, kratkoročno i dugoročno?

Kada ili gde bi ovo funkcionisalo?

> Pod kojim uslovima? Da li je prekasno? Prerano? Nemojte odmah prestati da razmišljate šta će se zatim dogoditi.

Zašto je ovo problem?

> Da li postoji osnovni model? Kako funkcioniše osnovni model?

Nažalost, postoji veoma malo jednostavnih odgovora. Međutim, zahvaljujući svom proširenom portfoliju i primeni kritičke analize tehničkih članaka koje ćete pročitati, možete da razumete složene odgovore.

Povezani odeljci uključuju

- Temu 1, „*To je vaš život*“, na stranici 2
- Temu 22, „*Dnevna knjiga inženjerstva*“, na stranici 100

Izazovi

- Počnite učenje novog jezika ove nedelje. Uvek programirate u istom starom jeziku? Isprobajte Clojure, Elixir, Elm, F#, Go, Haskell, Python, R, ReasonML, Ruby, Rust, Scala, Swift, TypeScript ili nešto drugo što vam se dopada ili što izgleda da bi moglo da vam se dopadne.[9]
- Počnite da čitate novu knjigu (ali prvo završite čitanje ove!). Ako vršite veoma detaljnu implementaciju i kodiranje, pročitajte knjigu o dizajnu i arhitekturi. Ako radite dizajn visokog nivoa, pročitajte knjigu o tehnikama kodiranja.
- Razgovarajte o tehnologiji sa ljudima koji nisu uključeni u vaš aktuelni projekat, ili koji ne rade za istu kompaniju. Umrežite se u kafiću vaše kompanije ili možete da potražite nekog entuzijastu na nekom lokalnom sastanku.

7 Komunicirajte!

Verujem da je bolje da me nadziru nego da me previde.

Mae West, Belle of the Nineties, 1934

Možda možete da naučite lekciju od gospođe West. Nije važno samo ono što imate, već i kako ćete to upakovati. Ako imate najbolje ideje i najbolji kod ili ako razmišljate najpragmatičnije, to je na kraju sterilno ako ne umete da komunicirate sa drugim ljudima. Dobra ideja je „siroče“ ako izostane efikasna komunikacija.

Kao programeri, treba da komuniciramo na mnogo nivoa. Provodimo sate na sastancima, slušamo i pričamo. Mi radimo sa krajnjim korisnicima, pokušavajući da razumemo njihove potrebe. Pišemo kod koji mašini prenosi naše namere i dokumentuje naše razmišljanje za buduće generacije programera. Mi pišemo predloge i dopise, tražeći i opravdavajući resurse, izveštavamo o statusu i preporučujemo nove pristupe. I svakodnevno radimo u našim timovima da bismo zagovarali naše ideje, modifikovali postojeću praksu i preporučili novu. Veliki deo dana provodimo u komunikaciji.

Tretirajte engleski jezik (ili vaš maternji jezik) kao još jedan programski jezik. Pišite na svom maternjem jeziku kao da pišete kod: poštujte DRY princip, ETC, automatizaciju i tako dalje (o DRY i ETC principima projektovanja biće reči u sledećem poglavlju).

9 Nikada niste čuli za ove jezike? Ne zaboravite, znanje je bogatstvo koje se troši, a isto tako se troše i popularne tehnologije. Lista najnovijih i eksperimentalnih jezika

Savet 11 Engleski je samo još jedan programski jezik

Sastavili smo listu dodatnih ideja koje smatramo korisnim.

Upoznajte publiku

Vi komunicirate samo ako prenosite ono što želite da prenesete - sam razgovor nije dovoljan. Da biste to uradili, treba da razumete potrebe, interesovanja i mogućnosti publike. Svi smo mi bili prisutni na sastancima na kojima neki razvojni „zaluđenik“ dugim monologom opisuje potpredsedniku marketinga prednosti neke tajnovite tehnologije. To nije komunikacija: to je samo govor i veoma je dosadan.[10]

Recimo da želite da promenite sistem za daljinsko praćenje i da upotrebite nezavisnog posrednika za poruke za širenje obaveštenja o statusu. Možete da predstavite ovo ažuriranje na više različitih načina, u zavisnosti od publike. Krajnji korisnici će ceniti što njihovi sistemi mogu interoperabilno raditi sa drugim servisima koji koriste posrednika. Vaše marketinško odeljenje će moći da iskoristi ovu činjenicu za povećanje prodaje. Razvojni i operativni menadžeri će biti zadovoljni, jer su briga za taj deo sistema i njegovo održavanje sada tuđi problemi. Na kraju, programeri mogu da uživaju što će iskusiti nove API-e i možda će pronaći neku novu namenu za posrednika za poruke. Ako pronađete odgovarajuću korist za svaku grupu, oni će biti uzbuđeni zbog vašeg projekta.

Kao i kod svakog drugog tipa komunikacije, trik je da sakupite povratne informacije. Nemojte samo čekati pitanja: postavite pitanje i vi. Pratite govor tela i izraze lica. Jedna od pretpostavki neuro-lingvističkog programiranja je „Smisao vaše komunikacije je odgovor koji dobijate“. Stalno povećavajte znanje publike dok komunicirate.

Treba da znate šta želite da kažete

Verovatno najteži deo formalnijih stilova komunikacije koja se koristi u poslu je da kažete upravo ono što želite. Pisci fantastike obično prave detaljan nacrt knjige pre nego što započnu pisanje, ali ljudi koji pišu tehničku dokumentaciju često zadovoljno sedaju uz tastaturu i unose:

1. Uvod

i započnu pisanje onoga što im padne napamet.

10 Reč annoy (dosada) potiče od stare francuske reči enui, što takođe znači „dosada“.

Isplanirajte ono što želite da kažete. Napišite koncept. Zatim se upitajte: „Da li ovo prenosi ono što želim da kažem mojoj publici na način koji joj odgovara?“. Ispravljajte dok ne bude tako.

Ovaj pristup funkcioniše za mnogo više elemanata, a ne samo za dokumente. Kada se suočite sa važnim sastankom ili ćaskate sa glavnim klijentom, zapišite ideje koje želite da prenesete i isplanirajte nekoliko strategija za njihovo prenošenje.

Sada, kada znate šta želi vaša publika, treba to da isporučite.

Izaberite momenat

Petak je, šest sati popodne, u nedelji kada su dolazili revizori, najmlađe dete vašeg šefa je u bolnici, napolju pada kiša i put do kuće je garantovana „noćna mora“. Ovo verovatno nije najbolji trenutak da zatražite nadgradnju memorije za vaš laptop.

Kao deo razumevanja onoga što publika želi da čuje, treba da otkrijete koji su njeni prioriteti. Presretnite menadžera koji je upravo pretrpeo kritiku od šefa zbog gubitka nekog izvornog koda i imaćete boljeg slušaoca za vaše ideje o skladištima izvornog koda. Učinite ono što govorite relevantnim za vreme i za sadržaj. Ponekad je potrebno samo jednostavno pitanje „Da li je ovo dobro vreme da razgovaramo o ...?“.

Izaberite stil

Prilagodite stil prenosa tako da odgovara vašoj publici. Neki ljudi žele formalni brifing „samo činjenice“. Drugi vole duge, sveobuhvatne razgovore pre nego što pređu na posao. Koji je njihov nivo veštine i iskustva u ovoj oblasti? Da li su stručnjaci? Novajlije? Da li im je potrebno držanje za ruku ili samo brz rezime? Ako niste sigurni, pitajte.

Međutim, ne zaboravite da ste vi polovina komunikacijske transakcije. Ako neko kaže da mu je potreban pasus koji opisuje nešto, a vi ne vidite način na koji biste mogli da obezbedite da traženi pasus ne bude dugačak nekoliko stranica, kažite mu tako. Ne zaboravite, i ta vrsta povratne informacije je forma komunikacije.

Učinite da izgleda dobro

Vaše ideje su važne. One zaslužuju „vozilo“ dobrog izgleda koje će ih preneti publici.

Previše programera (i njihovih menadžera) koncentriše se samo na sadržaj kada izrađuju pisane dokumente. Mi mislimo da je to pogrešno. Svaki kuvar (ili gledalac Food Networka) će vam reći da možete da „robujete“ u kuhinji satima, a da upropastite trud lošom prezentacijom.

U današnje vreme ne postoji izgovor za izradu štampanih dokumenata lošeg izgleda. Savremeni softver može da dovede do zadivljujućih rezultata, bez obzira da li pišete koristeći Markdown ili procesor reči. Potrebno je da naučite samo nekoliko osnovnih komandi. Ako koristite procesor reči, upotrebite njegove stilove za konzistentnost (vaša kompanija je, verovatno, već definisala stilove koje možete da upotrebite). Naučite kako da podesite zaglavlja i fusnote stranice. Pogledajte primer dokumenata koji su uključeni u paket da biste videli stil i raspored. *Proverite poštovanje pravopisa* - prvo automatski, a zatim ručno.

Uključite publiku

Često otkrijemo da su dokumenti koje izrađujemo na kraju manje važni od procesa koji prolazimo da bismo ih izradili. Ako je moguće, uključite čitaoce za rane nacrte dokumenta. Zatražite od njih povratne informacije. Izgradićete dobar radni odnos i verovatno ćete u tom procesu izraditi i bolji dokument.

Budite slušalac

Postoji jedna tehnika koju morate da koristite ako želite da vas ljudi slušaju: slušajte vi njih. Čak i ako imate sve informacije, čak i ako ste na formalniom sastanku na kojem vi stojite pred 20 ljudi, ako ih ne saslušate, oni neće slušati vas.

Postavljanjem pitanja ohrabrite ljude da govore ili ih zamolite da pokrenu raspravu sopstvenim rečima. Pretvorite sastanak u dijalog i efikasnije ćete izneti svoje mišljenje. Ko zna, možda ćete čak i naučiti nešto.

Javljajte se ljudima

Ako nekome postavite pitanje, mislićete da je nekulturan ako vam ne odgovori. Međutim, koliko često vi ne odgovorite ljudima kada vam pošalju e-mail ili dopis, tražeći informacije ili neku akciju? U žurbi svakodnevnog života veoma je lako da zaboravite da pošaljete odgovor. Uvek odgovorite na e-mailove i glasovne poruke, makar i sa jednostavnom porukom „Javiću vam se kasnije“. Ako informišete ljude, mnogo lakše će vam oprostiti povremeni propust, jer će imati osećaj da ih niste zaboravili.

Savet 12 Važno je šta kažete i način na koji to kažete

Osim ako radite u vakuumu, treba da možete da komunicirate. Što je komunikacija efikasnija, bićete uticajniji.

Dokumentacija

Na kraju, postoji i komunikacija pomoću dokumentacije. Obično, programeri ne razmišljaju mnogo o dokumentaciji. U najboljem slučaju to je nesrećna potreba, a u najgorem slučaju tretira se kao zadatak niskog prioriteta, u nadi da će menadžment zaboraviti na dokumentaciju po završetku projekta.

Pragmatični programeri prihvataju dokumentaciju kao integralni deo celokupnog procesa razvoja. Pisanje dokumentacije možete da olakšate ako ne duplirate rad ili dodatno vreme i držite dokumentaciju pri ruci - u samom kodu. U stvari, treba primeniti sve pragmatične principe za dokumentaciju kao i za kod.

Savet 13 Ugradite dokumentaciju, nemojte je uključivati

Veoma je lako kreirati dokumentaciju dobrog izgleda iz komentara u izvornom kodu - preporučujemo da dodate komentare u module i eksportovane funkcije da biste drugim programerima olakšali posao kada ga budu koristili.

Međutim, to ne znači da se slažemo sa ljudima koji kažu da svaka funkcija, struktura podataka, deklaracija tipa i tako dalje treba da ima svoj komentar. U stvari, ovaj tip mehaničkog pisanja komentara otežava održavanje koda; postoje dve stavke koje treba da ažurirate kada izvršite neku promenu. Ograničite komentare za stavke koje nisu API da biste opisali zašto je nešto urađeno, namenu i cilj. Kod već prikazuje kako je nešto urađeno, pa je komentar suvišan - i to je kršenje DRY principa.

Komentarisanje izvornog koda pruža savršenu mogućnost za dokumentovanje onih neuhvatljivih bitova projekta koji ne mogu na drugom mestu da se dokumentuju: koji su kompromisi inženjerstva, zašto su odluke donete, koje su druge alternative isključene i tako dalje.

Rezime

- Treba da znate šta želite da kažete.
- Upoznajte publiku.
- Izaberite momenat.
- Izaberite stil.
- Učinite da dobro izgleda.
- Uključite publiku.
- Budite slušalac.
- Javljajte se ljudima.
- Čuvajte kod i dokumentaciju zajedno.

Povezani odeljci uključuju:

- Temu 15, „*Procena*“, na stranici 65
- Temu 18, „*Editovanje*“, na stranici 81
- Temu 45, „*Zahtevi*“, na stranici 244
- Temu 49, „*Pragmatični timovi*“, na stranici 264

Izazovi

- Postoji nekoliko dobrih knjiga koje sadrže odeljke o komunikacijama unutar timova, uključujući „*The Mythical Man-Month: Essays on Software Engineering [Bro96]*“ i „*Peopleware: Productive Projects and Teams [DL13]*“. Pokušajte da ih pročitate u narednih 18 meseci. Osim toga, u knjizi „*Dinosaur Brains: Dealing with All Those Impossible People at Work [BR89]*“ opisan je emotivni prtljag koji svi mi nosimo u radno okruženje.
- Sledećeg puta kada bude trebalo da prikažete prezentaciju ili napišete dopis zalažući se za neku poziciju, pre nego što započnete, pokušajte da pratite savete u ovom odeljku. Konkretno, identifikujte publiku i ono što želite da prenesete. Ako je pogodno, razgovarajte sa svojom publikom nakon toga i proverite koliko su bile tačne vaše procene njenih potreba.

Online komunikacija

Sve što smo rekli u vezi komunikacije u pisanoj formi primenjuje se i na e-mail, postove društvenih medija, blogove i tako dalje. Posebno se e-mail razvio do tačke u kojoj predstavlja temelj poslovne komunikacije; koristi se za raspravu o ugovorima, za rešavanje sporova i kao dokaz na sudu. Međutim, događa se da ljudi koji nikada ne bi poslali otrcani papirni dokument često šalju grozne, nekoherentne e-mail poruke širom sveta.

Naši saveti su jednostavni:

- Proverite tekst pre nego što pritisnete SEND.
- Proverite poštovanje gramatike i potražite neke slučajne automatske ispravke.
- Format treba da bude jednostavan i jasan.
- Citirajte minimalno. Niko ne voli da primi e-mail poruku od 100 linija na koju ste dodali „Slažem se".
- Ako citirate e-mail poruke drugih ljudi, obavezno to navedite i umetnite citat (nemojte ga slati kao prilog). Isto važi i za citate na platformama društvenih medija.
- Nemojte se ponašati kao trol, ako ne želite da vam se to kasnije vrati i da vas proganja. Ako nešto ne biste rekli nekome u lice, nemojte reći ni online.
- Proverite listu primalaca pre nego što pošaljete e-mail. Postalo je uobičajeno kritikovati šefa pomoću poslovnog e-maila bez provere da li se on nalazi na CC listi. Nemojte da kritikujete šefa korišćenjem e-maila.

Kao što su brojne velike korporacije i političari otkrili, e-mail i postovi društvenih medija traju zauvek. Pokušajte da obratite istu pažnju na e-mailove kao i na bilo koji pisani dopis ili izveštaj.