Sadržaj

	Predgovor	vii
	Priča o Pravilima	xi
	Kako se ne složiti sa Pravilima	xv
Pravilo 1	Što je moguće jednostavnije, ali ne i previše pojednostavljeno	1
Pravilo 2	Greške su zarazne	15
Pravilo 3	Dobro ime je najbolja dokumentacija	31
Pravilo 4	Generalizacija traži tri primera	43
Pravilo 5	Prva lekcija optimizacije je ne optimizujte	59
	Interludij: U kome se kritikuje prethodno poglavlje	71
Pravilo 6	Revizije koda su dobre iz tri razloga	75
Pravilo 7	Eliminišite slučajeve neuspeha	81
Pravilo 8	Kôd koji se ne izvršava ne radi	97
Pravilo 9	Napišite kôd koji se može sažeti	109
Pravilo 10	Lokalizujte složenost	125
Pravilo 11	Da li je duplo bolje?	141
Pravilo 12	Velikim timovima su potrebne jake konvencije	149
Pravilo 13	Pronađite kamičak koji je pokrenuo lavinu	159
Pravilo 14	Četiri vrste koda	173
Pravilo 15	Iščupajte korov	187
Pravilo 16	Radite unazad, od rezultata, a ne unapred	193
Pravilo 17	Ponekad je veći problem lakše rešiti	213
Pravilo 18	Neka vaš kôd priča svoju priču	225

Pravilo 19 | Preradite paralelno 233

Pravilo 20 | Uradite matematiku 253

Pravilo 21 | Ponekad samo treba da zakucate eksere 263

| Zaključak: Uvedite sopstvena Pravila 271

Dodatak A | Čitanje C++ programa za Python programere 275

Dodatak B | Čitanje C++ programa za JavaScript programere 293

| Indeks 307

Predgovor

Dobro došli u *Pravila programiranja*, skup pravila koja se lako pamte i primenjuju i koja će vam pomoći da napišete bolji kôd. Programiranje je teško, ali pridržavanje Pravila ga čini malo lakšim.

Evo nekoliko saveta za čitanje knjige:

Svako pravilo je nezavisno. Ako vidite zanimljivo pravilo u sadržaju i želite da skočite pravo na njega u sredinu knjige, slobodno. Takav način čitanja je u potpunosti podržan.

- S obzirom na to, predlažem da počnete sa *Pravilom 1, „Što je moguće jednostavnije, ali ne suviše pojednostavljeno“*. To je dobra postavka za ostatak Pravila.
- Svi primeri u knjizi su napisani na jeziku C++. Ako ste Python ili JavaScript programer, pročitate *Dodatak A, „Čitanje C++-a za Python programere“ Dodatak B, „Čitanje C++-a za JavaScript programere“*, pre nego što uđemo predaleko u Pravila. Dva dodatka služe kao Kemen iz Rosete da prevedu C++ u koncepte na koje ste navikli. Ako je vaše iskustvo sa nekim drugim jezikom i smatrate da je C++ primere teško pratiti, onda predlažem fenomenalnu veb stranicu Rosetta Code (*https://oreil.ly/Rr2BL*).
- Ako ste C++ programer, imajte na umu da sam uprostio nekoliko stvari u primerima koda da bih ih učinio lakšim za čitanje programerima koji ne koriste C++. Na primer, primeri koriste cele brojeve sa preznakom na nekoliko mesta gde bi celi brojevi bez predznaka bili više tipični za C++ program, a isključio sam upozorenja o implicitnoj konverziji između vrednosti sa predznakom i bez predznaka. Takođe sam kompajliao primere sa implicitnim „using std“ da bih izbegao gomilu ometajućih „std::“ referenci.

I na kraju, pišem velikim slovom *Pravilo* kada govorim o stvarnom pravilu u knjizi. Ako vidite *pravilo*, to je samo uobičajeno staro pravilo, a ne zvanično označeno. Razlika između dva smisla te reči bila bi zbunjujuća bez upotrebe velikog slova.

Nadam se da ćete uživati u onome što sledi! Mislim da ćete otkriti korisne misli koje će vam pomoći da izoštrite svoje veštine programiranja.

Devojke koje kodiraju

Sav honorar od ove knjige ide za Girls Who Code (Devojke koje kodiraju) (*https://oreil.ly/QyCTX*), organizaciji koja naporno radi da pomogne mladim ženama da otkriju koliko programiranje može biti korisno. Kada sam diplomirao na fakultetu, više od trećine diplomiranih na računarskim naukama bile su žene; ovih dana je oko petine. Mislim da bi nam svima bilo bolje da postoji balans. Verovatno mislite i vi. A podrška za Girls Who Code kroz donacije ili volontiranje je korak ka tome da ta nada postane stvarnost.

Konvencije koje se koriste u ovoj knjizi

U ovoj knjizi se koriste sledeće tipografske konvencije:

Italic
: Označava nove termine, URL adrese, adrese e-pošte, nazive datoteka i tipove datoteka.

`Konstantna širina`
: Koristi se za listinge programa, kao i unutar pasusa za upućivanje na elemente programa kao što su imena promenljivih ili funkcija, baze podataka, tipovi podataka, promenljive okruženja, izjave i rezervisane reči.

Upotreba primera koda

Dodatni materijal (kôd primera, vežbe itd.) dostupan je za preuzimanje na *https://oreil.ly/rules-of-programming-code*.

Ako imate tehničko pitanje ili problem sa korišćenjem primera koda, pošaljite e-poštu na *bookquestions@oreilly.com*.

Ova knjiga je tu da vam pomogne da završite svoj posao. Generalno, ako se uz ovu knjigu nudi primer koda, možete ga koristiti u svojim programima i dokumentaciji. Ne morate da nas kontaktirate za dozvolu osim ako ne reprodukujete značajan deo koda. Na primer, za pisanje programa koji koristi nekoliko delova koda iz ove knjige nije potrebna dozvola. Prodaja ili distribucija primera iz O'Reilly knjiga zahteva dozvolu. Da biste odgovorili na pitanje citiranjem ove knjige i citiranjem primera koda, nije potrebna dozvola. Ugrađivanje značajne količine primera koda iz ove knjige u dokumentaciju vašeg proizvoda zahteva dozvolu.

Cenimo, ali ne zahtevamo, navođenje. Navođenje obično sadrži naslov, autora, izdavača i ISBN. Na primer: „*The Rules of Programming* autora Chris Zimmerman (O'Reilly). Copyright 2023 Chris Zimmerman, 978-1-098-13311-5."

Ako smatrate da je vaša upotreba primera koda van korektne upotrebe ili gore navedene dozvole, slobodno nas kontaktirajte na *permissions@oreilly.com*.

Kako da nas kontaktirate

Komentare i pitanja u vezi ove knjige uputite izdavaču:

MIKRO KNJIGA
+381 11 7702-883
redakcija@mikroknjiga.rs

Izrazi zahvalnosti

Prvo, zahvaljujem se svojoj divnoj i talentovanoj ženi Lauri, koja me je ohrabrila da potrošim vreme na pisanje ove knjige umesto da radim sve druge korisne stvari koje sam mogao da radim.

Veliko hvala svim ljudima koji su pomogli u razvoju Pravila u ovoj knjizi. To uključuje sve bivše i sadašnje Sucker Punch kodere, pošto ste svi dali svoj doprinos bez obzira na to da li ste ili niste nameravali, ali posebnu zahvalnost zaslužuju Apoorva Bansal, Chris Heidorn, David Meyer, Eric Black, Evan Christensen, James McNeill, Jasmin Patry, Nate Slottow, Matt Durasoff, Mike Gaffney, Ranjith Rajagopalan, Rob McDaniel, Sam Holley, Sean Smith, Wes Grandmont i William Rossiter.

I hvala ne-Sucker Punchersima koji su pružili pogled izvan šume: Adam Barr, Andreas Fredriksson, Colin Bryar, David Oliver, Max Schubert, Mike Gutmann i Seth Fine.

Posebno se zahvaljujemo neustrašivim čitaocima koji su prošli kroz svako od Pravila, a to su: Adrian Bentley, Bill Rockenbeck, Jan Miksovsky, and Julien Merceron. Zvanično svima vama dugujem uslugu.

I na kraju, hvala svima u timu O'Reilly koji me je strpljivo podučavao u mojim nespretnim pokušajima da napišem ovu knjigu: Charles Roumeliotis, Gregory Hyman, Libby James, Mary Treseler, Sara Hunter, Suzanne Huston, i naročito Sarah Grey, koja je vama čitaocima učinila ogromnu uslugu tako što je izbacila većinu šala koje sam insistirao da dodam.

Priča o Pravilima

Pravila programiranja su rođena iz ogorčenja.

Proveo sam oko deceniju vodeći programske timove u Microsoftu, a zatim suosnivao kompaniju za video igre Sucker Punch 1997. Obe kompanije su bile uspešne – velikim delom zbog svoje sposobnosti da regrutuju i razviju vrhunske programerske timove. U Sucker Punchu, to je dovelo do 25-godišnjeg niza uspešnih igara. Postojale su tri igre *Sly Cooper*, koje su omogućile deci svih uzrasta da iskuse uzbudljiv život glavnog kradljivca rakuna Sly Coopera i njegovih prijatelja. Bilo je pet *inFamous* igara, koje su igračima dale supermoći i izbor da ih koriste za dobro ili zlo. A tu je i ono što je do sada naš veliki opus, *Ghost of Tsushima*, gde igrači igraju usamljenog samuraja koji se bori protiv invazije na Japan 1274. godine.[1]

Veliki deo strategije regrutovanja i u Microsoftu i u Sucker Punchu je zapošljavanje pametnih mladih programera, a zatim njihovo obučavanje na način profesionalnih programera. Ova praksa je bila neosporno uspešna, ali je vodila i do određene frustracije.

Stalno sam nailazio na isti problem iznova i iznova. Doveli bismo nove programere u tim, često sa tek završenim fakultetom. Pregledao bih neku novu funkciju koju su planirali da dodaju u kôd, obično za rešavanje veoma jednostavnog problema – da bih otkrio da su napisali kôd koji pokušava da reši mnogo veći problem, onaj u kome je veoma jednostavan i konkretan problem samo mali podslučaj.

Grrr! Nije nam potrebno da se reši taj veći problem, svakako ne sada! Kao i uvek rešenje većeg problema je osrednje rešenje jednostavnog problema koga imamo – komplikovano za korišćenje, komplikovano za razumevanje i moguće je da krije mnogo grešaka. Ali kada bi im u recenziji koda rekli[2] – da nam nije potrebno rešavanje većeg problema, da treba da pokušaju da reše problem koji razumeju – bilo bi uzalud. Nastavljali su to da rade.

---

1 Oštrooki među vama su možda primetili da sam izostavio *Rocket: Robot on Wheels*, prvu igru Sucker Puncha. To je zato što je retko ko od vas nju igrao; ako ste jedan od retkih, imate moju zahvalnost.

2 Sav kôd projekta Sucker Puncha prolazi recenziju koda. Pogledajte Pravilo 6 za više detalja.

Zbog frustracije, rekao sam dosta. „U redu. Evo novog pravila. Dok nemate tri primera problema, nije vam dozvoljeno da napišete opšte rešenje".

Na moje iznenađenje i oduševljenje, ovo je zaista dalo rezlutat! Pretvaranje opšte filozofije u specifično pravilo sa specifičnim kriterijumima bio je efikasan način da se poruka prenese. Naravno, većina naših novih programera bi jednom napravila grešku preuranjene generalizacije, ali im je pravilo pomoglo da izbegnu da je ponovo naprave. Pomoglo im je i da prepoznaju kada je *došlo* vreme da generalizuju. Manje od tri primera? Ne generalizujte. Tri ili više? Počnite da tražite mogućnosti.

Pravilo je funkcionisalo jer ga je bilo lako zapamtiti, a situacije u kojima se primenjuje bilo je lako prepoznati. Kada su koderi mogli da vide da su počeli da prelaze granice dobro definisanog problema, mogli su da naprave korak unazad, izbroje broj konkretnih primera te vrste problema na koje bi naišli i doneli bolju odluku o tome da li generalizovati ili ne. Pisali su bolji kôd.

Vremenom smo pronašli druge važne delove filozofije Sucker Puncha koji bi mogli da se destiluju u lako pamtljive fraze – aforizme, da budemo precizni. Duga je istorija aforizama – tih kratkih, sadržajnih izjava koje obuhvataju neku suštinsku istinu. Kladim se da možete da izvučete popriličan broj iz sećanja. Mogao bih da vas ograničim na aforizme *u vezi sa pticama*, a vi biste ipak mogli da smislite bar dva! Ponudiću nekoliko:

- Ne brojite svoje piliće pre nego što se izlegu.[3]
- Bolje vrabac u ruci nego golub na grani.
- Ptica koja se prva izleže uzima crva.
- Ne stavljajte sva jaja u jednu korpu.

Aforizmi postoje jer funkcionišu. Oni su viralni, u savremenom smislu – hiljadama godina aforizmi „inficiraju" ljude delićima mudrosti.[4] Nije iznenađujuće što su efikasan način zaražavanja novih članova tima sa filozofijom kodiranja Sucker Punch.

Dakle, mic po mic, ono što je nekada bilo jedno pravilo preraslo je u listu pravila: Pravila programiranja opisana u ovoj knjizi. Ona predstavljaju mnoge od najvažnijih aspekata inženjerske kulture Sucker Puncha: stvari za koje verujemo da

3 U svom originalnom obliku, iz *Novi soneti i lepi pamfleti (New Sonnets and Pretty Pamphlets)* Thomas Howella (1570): „Ne računaj svoje neizležene kokoške..." (Count not thy chickens that unhatched be...) Mala demonstracija trajnosti dobrog aforizma.

4 Samu reč *aforizam* je skovao Hipokrat oko 400. godine pre nove ere. Pa, strogo govoreći, reč koju je skovao bila je Ἀφορισμός. To je bio naslov njegove knjige pravila za medicinsku dijagnozu i lečenje, od kojih su neka i dalje prisutna milenijumima kasnije, poput aforizma 13 iz odeljka 6 – „napad kijanja će izlečiti slučaj štucanja". Tako istinito.

su dovele do našeg uspeha, ideje koje novi koderi u timu moraju da usvoje da bi bili efikasni. Stvari na koje čak i stariji koderi poput mene treba da se podsete bar s vremena na vreme!

Svako od poglavlja koje sledi opisuje Pravilo, sa mnogo primera koji ilustruju ideju iza njega. Nakon čitanja poglavlja, trebalo bi da imate jasnu predstavu o praksi kodiranja koju Pravilo podstiče i situacijama u kojima se primenjuje.

Da li su Pravila isto tako zarazna i u obliku knjige? Hajde da otkrijemo.

Kako se ne složiti sa Pravilima

Nadam se da nećete glatko proći kroz svih 21 Pravila.

Ako ljubazno klimate glavom uz svako od Pravila i uz primere koje koristim – „O da, to ima smisla, ovaj primer mi je poznat, već sam razmišljao o tome, samo sam koristio drugačiji jezik da to opiše" – onda, to je neuspeh.

Nadam se da ću vam dati stvari za razmišljanje. Idealno, jedan ili dva nova uvida. Možda ću uspeti da imenujem neki nejasan osećaj koji ste imali, ili da vam dam jasan primer nečega što niste uspeli da razumete. Možda ću vam čak dati nešto sasvim novo za razmatranje.

Ali je i verovatno da ćete naići na jednu ili dve ideje sa kojima se ne slažete. Možda mislite da sam potpuno pogrešio u vezi nečega – da je neko od Pravila loše.

To je dobro! Pronalaženje Pravila sa kojim se u potpunosti ne slažete je prilika. Odbaciti ga odmah bila bi greška.

Obećavam da dotično Pravilo nije sasvim pogrešno – ali može biti ispravno za *nas* dok je istovremeno pogrešno za *vas*. Razumevanje zašto je to tako će vam pomoći da razumete i ojačate sopstvenu filozofiju programiranja. To znači da razumete razlike između Sucker Puncha i vašeg sopstvenog tima, jer te razlike čine Pravilo važnim delom naše kulture, ali lošim uklapanjem sa vašom. Pravila koriste primere iz video igara koje smo stvorili u Sucker Punchu, i koja treba da osvetle neke stvari koje programiranje video igara čine drugačijim. Mnoga od ostalih su obrađena u poslednjem poglavlju, „Zaključak: Uvedite sopstvena Pravila".

Evo procesa pomirenja koji mi je bio koristan kada sam naišao na neku izjavu o filozofiji kodiranja koja nije u skladu sa mojim iskustvom:

1. Pronađite istinu u izjavi, a ne samo nedostatke. Možda se ne slažem sa izjavom, ali to je verovatno zbog mojih sopstvenih pretpostavki. Koje su okolnosti pod kojima bi izjava bila tačna?
2. Rešite problem i sa druge strane. Koje su okolnosti pod kojima bi moj suprotan stav o izjavi bio lažan? Koja je razlika u okolnostima koje menjaju istinitost izjave?

3. Imajte na umu da se okolnosti menjaju. Izjava može biti pogrešna za vas sada, ali ispravna na vašem sledećem projektu. Identifikovali ste situaciju u kojoj bi vaša filozofija morala da se promeni; budite oprezni za mogućnost da ste zalutali upravo u tu situaciju.

Prošao sam kroz ovaj proces mnogo puta. Evo primera: razvoj vođen testom (TDD, test-driven development). To nije u skladu sa našim iskustvom u Sucker Punchu; ipak nam je istina u tome očigledna. Videćete reference u Pravilima na načine na koje naše okolnosti čine TDD nezgodnim, ali znamo da se te okolnosti mogu promeniti. Mi smo oprezni; ako se stvari promene, promeniće se i naša filozofija.

Tako da se nadam da ćete naći vrednost u Pravilima koja vam najviše smetaju... ali razumem ako odlučite da sledite drugačiji kurs akcije, onaj koji se pripisuje Dorothy Parker:

Ovo nije roman koji treba olako odbaciti. Treba ga baciti velikom snagom.

Ako je tako, predlažem da ciljate na nešto meko.