

---

# Sadržaj

---

Predgovor .....	xv
Uvod .....	xxi
Na korici .....	xxv
<b>Poglavlje 1: Jasan kod</b> .....	<b>1</b>
Neka bude kod .....	2
Loš kod .....	2
Ukupan trošak nereda .....	3
Veliko redizajnijske na pomolu .....	4
Stav .....	5
Početni problem .....	5
Umetnost jasnog koda .....	6
Šta je jasan kod? .....	6
Škole mišljenja .....	11
Mi smo autori .....	12
Pravilo izviđača .....	13
Prethodište i principi .....	13
Zaključak .....	14
Bibliografija .....	14
<b>Poglavlje 2: Smisljena imena</b> .....	<b>15</b>
Uvod .....	15
Koristite imena koja otkrivaju namenu .....	16
Izbegavajte dezinformacije .....	17
Pravite smisljene razlike .....	18
Koristite izgovorljive nazive .....	19
Koristite pretraživa imena .....	20

Izbegavajte kodiranje.....	21
Mađarska notacija.....	21
Prefiksi .....	21
Interfejsi i implementacije .....	22
Izbegavajte mentalno mapiranje.....	22
Imena klase .....	23
Nazivi metoda .....	23
Ne budite slatki.....	23
Izaberite jednu reč po konceptu .....	24
Ne igrajte se rečima.....	24
Koristite imena prema gotovim rešenjima .....	25
Koristite imena koja se odnose na problem koji rešavate.....	25
Dodajte smisljeno značenje .....	25
Nemojte dodavati nepotrebno značenje.....	28
Završne reči .....	28
<b>Poglavlje 3: Funkcije .....</b>	<b>29</b>
Mala! .....	32
Blokovi i uvlačenja .....	33
Činite jednu stvar .....	33
Sekcije u okviru funkcija.....	34
Jedan nivo apstrakcije po funkciji.....	34
Čitanje koda od vrha ka dnu: <i>Pravilo silaska</i> .....	35
Switch naredbe .....	35
Koristite opisna imena.....	37
Argumenti funkcije .....	38
Uobičajeni jednoargumentni oblici.....	39
Argumenti pokazivači .....	39
Dvoargumentne fukcije .....	40
Troargumentne funkcije .....	40
Argument objekti.....	41
Spisak argumenata.....	41
Glagoli i ključne reči.....	41
Bez nuspojava.....	42
Izlazni argumenti.....	43
Razlaganje komandi upita .....	43
Dajte prednost izuzecima a ne vraćanju poruke o grešci.....	44
Izdvajanje Try/Catch blokova .....	44
Obrada grešaka je jedna stvar .....	45
Magnetna zavisnost klase Error.java .....	45

Ne ponavljajte se.....	46
Strukturirano programiranje .....	46
Kako pišete ovakve funkcije?.....	47
Zaključak.....	47
SetupTeardownIncluder .....	48
Bibliografija.....	50
<b>Poglavlje 4: Komentari .....</b>	<b>51</b>
Komentari ne popravljaju loš kod.....	52
Izražavajte se kodom .....	53
Dobri komentari .....	53
Pravni komentari .....	53
Informativni komentari .....	53
Objašnjenje namere.....	54
Razjašnjenje.....	55
Upozorenje na posledice .....	55
Komentari TODO .....	56
Naglašavanje.....	57
Generatori dokumentacije (Javadocs) u javnim API-jima .....	57
Loši komentari .....	57
Mrmljanje .....	57
Suvišni komentari.....	58
Komentari koji vode na pogrešan put.....	60
Obavezni komentari.....	61
Komentari u vidu dnevnika .....	61
Komentari koji predstavljaju „šum“.....	62
Zastrašujući „šum“ .....	64
Ne koristite komentar kada možete koristiti funkciju ili promenljivu.....	64
Oznake položaja.....	64
Komentari na zatvarajućim zagradama.....	65
Prateća objašnjenja .....	65
Kod pretvoren u komentar .....	66
HTML komentari .....	66
Informacije od opšteg značaja .....	67
Previše informacija.....	67
Neočigledna veza.....	68
Funkcije zaglavlja .....	68
Javadocs u kodu koji nije javan .....	68
Primer .....	68
Bibliografija.....	72

<b>Poglavlje 5: Formatiranje</b> .....	73
Svrha formatiranja.....	74
Vertikalno oblikovanje .....	74
Novinska metafora.....	75
Vertikalna otvorenost između pojmova.....	76
Vertikalna gustina .....	77
Vertikalna udaljenost.....	78
Vertikalni redosled .....	82
Horizontalno formatiranje.....	83
Horizontalna otvorenost i gustina .....	83
Horizontalno poravnanje .....	84
Uvlačenja .....	86
Jednoredni opseg .....	87
Timska pravila.....	87
Pravila oblikovanja ujka Boba.....	88
<b>Poglavlje 6: Objekti i strukture podataka</b> .....	91
Apstrakcija podataka.....	91
Antisimetrija podataka/objekata.....	93
Zakon Demetera .....	96
Sudar vozova .....	96
Hibridi .....	97
Sakrivajuća struktura .....	97
Objekti za prenos podataka .....	98
Aktivni zapisi .....	99
Zaključak.....	99
Bibliografija.....	99
<b>Poglavlje 7: Rad sa greškama</b> .....	101
Koristite izuzetke, a ne povratne kodove .....	102
Napišite svoj Try-Catch-Finally kod.....	103
Koristite neproverene izuzetke.....	104
Obezbedite kontekst sa izuzecima .....	105
Definišite klase izuzetaka prema potrebama pozivaoca .....	105
Definišite normalan protok .....	107
Ne vraćajte Null .....	108
Ne prosleđujte Null.....	109
Zaključak.....	110
Bibliografija.....	110

<b>Poglavlje 8: Granice</b> .....	111
Korišćenje koda drugih proizvođača.....	112
Istraživanje i spoznaja granica .....	114
Spoznaja log4j.....	114
Testovi za učenje su bolji nego besplatni.....	116
Korišćenje koda koji još ne postoji.....	116
Jasne i čiste granice .....	118
Bibliografija.....	118
<b>Poglavlje 9: Testiranje</b> .....	119
Tri zakona programiranja zasnovanog na testiranju.....	120
Održavanje testova jasnim .....	121
Testovi omogućavaju sve .....	122
Jasni testovi .....	122
Jezik za testiranje za određeni domen .....	125
Dvostruki standard .....	126
Jedna tvrdnja po testu.....	128
Jedan koncept po testu.....	129
F.I.R.S.T. ....	130
Zaključak.....	131
Bibliografija.....	131
<b>Poglavlje 10: Klase</b> .....	133
Organizacija klase .....	134
Enkapsulacija.....	134
Klase treba da budu male! .....	134
Princip jednodgovornosti.....	137
Kohezija .....	138
Održavanje rezultata kohezije u mnogo malih klasa .....	139
Organizovanje za promene .....	145
Izolovanje od promene.....	148
Bibliografija.....	149
<b>Poglavlje 11: Sistemi</b> .....	151
Kako biste izgradili grad? .....	152
Odvojite izgradnju sistema od njegove upotrebe.....	152
Razdvajanje glavnog.....	153
Abstract Factory .....	153
Uvođenje zavisnosti .....	154
Proporcionalno uvećanje .....	155
Nadležnosti unakrsnog preseka .....	158

Java proksiji.....	159
Čist Java AOP radni okvir.....	161
AspectJ aspekti.....	164
Test systemske arhitekture.....	164
Optimizujte donošenje odluka.....	165
Koristite standarde mudro, onda kada dodaju <i>vidljivu</i> vrednost.....	165
Sistemima je potreban određeni jezik domena.....	166
Zaključak.....	166
Bibliografija.....	167
<b>Poglavlje 12: Nastajanje.....</b>	<b>169</b>
Biti jasan pomoću dizajna u nastajanju.....	169
Jednostavno dizajniranje, pravilo 1: Izvršava sve testove.....	170
Jednostavno dizajniranje, pravila 2–4: Refaktorisanje.....	170
Bez duplikata.....	171
Izražajno.....	173
Što manji broj klasa i metoda.....	174
Zaključak.....	174
Bibliografija.....	174
<b>Poglavlje 13: Paralelnost.....</b>	<b>175</b>
Zašto paralelnost?.....	176
Mitovi i zablude.....	177
Izazovi.....	177
Principi odbrane paralelnosti.....	178
Princip jedinstvene odgovornosti.....	178
Posledica: Ograničite opseg podataka.....	178
Zaključak: Koristite kopije podataka.....	179
Posledica: Niti moraju biti što je moguće više nezavisne.....	179
Upoznajte svoju biblioteku.....	180
Kolekcije koda sigurne za rad sa nitima.....	180
Upoznajte svoje modele izvršenja.....	180
Proizvođač-korisnik.....	181
Čitaoci-upisivači.....	181
Filozofi za stolom.....	182
Pazite na zavisnosti između sinhronizovanih metoda.....	182
Održavajte sinhronizovane delove malim.....	183
Pisanje tačnog koda za isključivanje je teško.....	183
Testiranje koda za nitni rad.....	183
Lažne propuste tretirajte kao kandidate za probleme.....	184
Prvo učinite da radi kod koji nije nitni.....	184

Učinite da kod sa nitima bude uklopljiv .....	184
Učinite da kod sa nitima bude prilagodljiv .....	185
Radite sa više niti nego procesora .....	185
Izvršavajte kod na različitim platformama .....	185
Pokušajte da „opteretite“ kod da biste ga naterali na grešku .....	185
Ručno kodiranje .....	186
Automatizovano menjanje .....	186
Zaključak .....	187
Bibliografija .....	188
<b>Poglavlje 14: Uzastopno usavršavanje .....</b>	<b>189</b>
Implementacija Argova .....	190
Kako sam to uradio? .....	197
Args: Gruba skica .....	197
Tako da sam stao .....	210
O inkrementiranju .....	211
Argumenti String .....	213
Zaključak .....	253
<b>Poglavlje 15: JUnit iznutra .....</b>	<b>255</b>
JUnit radni okvir .....	256
Zaključak .....	270
<b>Poglavlje 16: Refaktorisanje klase SerijalDate .....</b>	<b>271</b>
Prvo, učinite da radi .....	272
Zatim ispravite .....	274
Zaključak .....	288
Bibliografija .....	289
<b>Poglavlje 17: „Mirisi“ i heuristika .....</b>	<b>291</b>
Komentari .....	292
C1: <i>Neodgovarajuće informacije</i> .....	292
C2: <i>Nepotreban komentar</i> .....	292
C3: <i>Suvišan komentar</i> .....	292
C4: <i>Loše napisan komentar</i> .....	293
C5: <i>Isključen kod</i> .....	293
Okruženje .....	293
E1: <i>Za izgradnju je potrebno više od jednog koraka</i> .....	293
E2: <i>Testovi zahtevaju više od jednog koraka</i> .....	293

Funkcije.....	294
F1: <i>Previše argumenata</i> .....	294
F2: <i>Izlazni argumenti</i> .....	294
F3: <i>Argumenti pokazivači</i> .....	294
F4: <i>Mrtva funkcija</i> .....	294
Uopšteno.....	294
G1: <i>Više jezika u jednoj izvornoj datoteci</i> .....	294
G2: <i>Očigledno ponašanje je neimplementirano</i> .....	294
G3: <i>Netačno ponašanje u graničnim slučajevima</i> .....	295
G4: <i>Zaobilaženje bezbednosti</i> .....	295
G5: <i>Udvostručavanje</i> .....	295
G6: <i>Kod na pogrešnom nivou apstrakcije</i> .....	296
G7: <i>Osnovne klase zavise od njihovih derivata</i> .....	297
G8: <i>Previše informacija</i> .....	297
G9: <i>Mrtav kod</i> .....	298
G10: <i>Vertikalno odvajanje</i> .....	298
G11: <i>Nedoslednost</i> .....	298
G12: <i>Nered</i> .....	299
G13: <i>Veštačko povezivanje</i> .....	299
G14: <i>Osobina zavisti</i> .....	299
G15: <i>Argumenti selektori</i> .....	300
G16: <i>Nejasna namera</i> .....	301
G17: <i>Pogrešno postavljena odgovornost</i> .....	301
G18: <i>Neprikladna statika</i> .....	302
G19: <i>Koristite objašnjavajuće promenljive</i> .....	302
G20: <i>Imena funkcija treba da kažu šta one rade</i> .....	303
G21: <i>Razumite algoritam</i> .....	303
G22: <i>Učinite logičke zavisnosti fizičkim</i> .....	304
G23: <i>Dajte prednost polimorfizmu u odnosu na If/Else         ili Switch/Case</i> .....	305
G24: <i>Pridržavajte se standardnih konvencija</i> .....	306
G25: <i>Zamenite čarobne brojeve sa imenovanim konstantama</i> .....	306
G26: <i>Budite precizni</i> .....	307
G27: <i>Struktura pre konvencije</i> .....	307
G28: <i>Enkapsuliranje uslova</i> .....	307
G29: <i>Izbegavajte negativne uslove</i> .....	308
G30: <i>Funkcije bi trebalo da urade jednu stvar</i> .....	308
G31: <i>Skrivena vremenska povezivanja</i> .....	309
G32: <i>Nemojte biti proizvoljni</i> .....	310



G33: <i>Granični uslovi enkapsuliranja</i> .....	310
G34: <i>Funkcije treba da se spuštaju samo na jedan nivo apstrakcije</i> .....	310
G35: <i>Čuvajte podatke koji se mogu konfigurisati na visokim nivoima</i> .....	312
G36: <i>Izbegavajte tranzitivnu navigaciju</i> .....	313
Java.....	313
J1: <i>Izbegavajte duge spiskove uvoza koristeći džokere</i> .....	313
J2: <i>Ne nasledujte konstante</i> .....	314
J3: <i>Konstante naspram Enumeratora</i> .....	315
Imena.....	316
N1: <i>Izaberite opisna imena</i> .....	316
N2: <i>Izaberite imena na odgovarajućem nivou apstrakcije</i> .....	317
N3: <i>Koristite standardnu nomenklaturu gde god je to moguće</i> ....	318
N4: <i>Jednoznačna imena</i> .....	318
N5: <i>Koristite dugačka imena za velike opsege</i> .....	319
N6: <i>Izbegavajte kodiranje</i> .....	319
N7: <i>Imena treba da opisuju sporedna dejstva</i> .....	319
Testovi .....	319
T1: <i>Nedovoljni testovi</i> .....	319
T2: <i>Koristite alat za prikrivanje!</i> .....	320
T3: <i>Ne preskačite jednostavne testove</i> .....	320
T4: <i>Zanemareni test je pitanje o nejasnoći</i> .....	320
T5: <i>Testiranje graničnih uslova</i> .....	320
T6: <i>Detaljno testiranje u blizini grešaka</i> .....	320
T7: <i>Obrasci otkazivanja se otkrivaju</i> .....	320
T8: <i>Obrasci ispitivanja mogu se otkriti</i> .....	320
T9: <i>Testovi treba da budu brzi</i> .....	321
Zaključak.....	321
Bibliografija.....	321
<b>Dodatak A: Paralelnost II</b> .....	<b>323</b>
Primer klijent/server.....	323
Server.....	323
Dodavanje niti.....	324
Nadgledanje servera .....	325
Zaključak .....	327
Mogući putevi izvršenja.....	327
Broj putanja .....	328
Kopanje dublje .....	329
Zaključak .....	332

Upoznajte svoju biblioteku .....	332
Izvršni radni okvir .....	332
Neblokirajuća rešenja .....	333
Nenitne bezbedne klase .....	334
Zavisnosti između metoda mogu pokvariti paralelan kod .....	335
Tolerisanje otkazivanja .....	336
Zaključavanje na strani klijenta .....	336
Zaključavanje na strani servera .....	338
Povećanje propusnosti .....	339
Proračun protoka sa jednom niti .....	340
Proračun protoka sa više niti .....	340
Zastoj .....	341
Uzajamno isključivanje .....	342
Zaključaj i sačekaj .....	342
Nema preko reda .....	342
Kružno čekanje .....	342
Prekid uzajamnog isključivanja .....	343
Prekidanje zaključavanja i čekanja .....	343
Prekidanje prekorednosti .....	344
Prekid kružnog čekanja .....	344
Testiranje višenitnog koda .....	344
Alati za podršku testiranja višenitnog koda .....	347
Zaključak .....	348
Vežba: Primeri punog koda .....	348
Klijent/server bez niti .....	348
Klijent/server pomoću niti .....	352
<b>Dodatak B: org.jfree.date.SerialDate .....</b>	<b>355</b>
<b>Dodatak C: Uporedne reference poglavlja i heuristika .....</b>	<b>427</b>
<b>Epilog .....</b>	<b>429</b>
<b>Indeks .....</b>	<b>431</b>

---

# Predgovor

---

Jedan od naših omiljenih slatkiša ovde u Danskoj je Ga-Jol, čiji jaki miris sladića je savršen dodatak našem vlažnom i često hladnom vremenu. Deo šarma Ga-Jola za nas Dance jesu mudre ili duhovite izreke ispisane na poklopcu svake kutije. Jutros sam kupio paketić poslastice i otkrio da nosi ovu staru dansku izreku:

*Ærlighed i små ting er ikke nogen lille ting.*

„Iskreno, male stvari nisu male stvari.“ Bio je to dobar predznak u skladu sa onim što sam ovde već hteo da kažem. Male stvari su bitne. Ovo je knjiga o poniznim brigama čija je vrednost daleko od male.

*Bog je u detaljima*, rekao je arhitekta Ludwig mies van der Rohe. Ovaj citat podseća na savremene argumente o značaju arhitekture u razvoju softvera, posebno u Agile svetu. Bob i ja povremeno se upuštamo u žestok dijalog o tome. I da, Ludwig mies van der Rohe je ukazvao na korisnost i bezvremenost građevina koje predstavljaju najbolju arhitekturu. S druge strane, lično je odabirao kvake za vrata svake kuće koju je projektovao. Zašto? Jer su male stvari važne.

Tokom naše „rasprave“ o razvoju zasnovanom na testiranju, Bob i ja smo otkrili da se slažemo da softverska arhitektura ima važno mesto u razvoju, mada verovatno imamo različite predstave o tome šta to konkretno znači. Takve razlike su, međutim, relativno nevažne, jer možemo prihvatiti zdravo za gotovo da odgovorni profesionalci na početku projekta odvoje *neko* vreme za razmišljanje i planiranje. Pojmovi projektovanja zasnovanog na testiranju i kodu nestali su kasnih 1990-ih. Ipak, pažnja prema detaljima postala je još više važna osnova profesionalizma, više nego što je to velika vizija. Prvo, kroz praksu na detaljima profesionalci stiču stručnost i sigurnost za rad na velikim stvarima. Drugo, najmanja a pomalo neuredna konstrukcija, vrata koja se ne zatvaraju čvrsto ili blago zakrenuta pločica na podu, ili čak neuredan radni sto, u potpunosti kvare šarm celine. O tome se radi kada je u pitanju jasan i čist kod.

Ipak, arhitektura je samo jedna metafora razvoja softvera, a posebno za onaj deo softvera koji isporučuje početni *proizvod* u istom smislu kao što arhitekta isporučuje netaknutu zgradu. U današnje doba Scruma and Agile fokus je na brzom stavljanju

*proizvoda* za tržište. Želimo da fabrika radi maksimalnom brzinom u proizvodnji softvera. To su ljudske fabrike: razmišljanje, osećanje za kodiranje koje se radi na osnovu prethodnih proizvoda i zahtevi korisnika da bi stvorili proizvod. Metafora proizvodnje snažno odgovara takvim razmišljanjima. Proizvodni aspekti japanske auto-proizvodnje, sveta montažnih pokretnih traka, nadahnjuju Scrum.

Ipak, čak i u auto industriji, najveći deo posla nije u proizvodnji, već u održavanju – ili njegovom izbegavanju. U softveru, 80% ili više onoga što radimo je čin popravke a naziva se sa izmišljenom reči „održavanje“. Umesto da se bavimo tipičnim zapadnjačkim fokusom na *proizvodnji* dobrog softvera, mi više razmišljamo kao kućni majstor u građevinskoj industriji ili automehaničari u automobilskoj industriji. Šta bi japansko rukovodstvo reklo na *to*?

Otrprilike 1951. godine na japanskoj sceni se pojavio pristup kvalitetu nazvan Održavanje potpune produktivnosti (Total Productive Maintenance, TPM). Njegov fokus je na održavanju, a ne na proizvodnji. Jedan od glavnih stubova TPM-a je skup takozvanih 5S principa. 5S je skup disciplina – i ovde poučno koristim termin „disciplina“. Ovi 5S principi su u stvari osnove Leana – još jedna zvučna reč na zapadnoj sceni i sve istaknutija reč u softverskim krugovima. Ovi principi nisu opcija. Kao što ujak Bob govori u svojoj prvoj stvari, dobra softverska praksa zahteva takvu disciplinu: fokus, prisustvo uma i razmišljanje. Ne radi se uvek samo o tome kako da se postigne optimalna brzina rada fabričkih mašina. Filozofija 5S sadrži ove koncepte:

- *Seiri* ili organizacija („sortirajte“). Znajte gde su stvari – korišćenje pristupa kao što je pogodno imenovanje – je presudno. Mislite da imenovanje nije važno? Pročitajte u narednim poglavljima.
- *Seiton*, ili urednost („sistemizovati“). Postoji stara američka izreka: *Mesto za sve i sve na svom mestu*. Deo koda treba da bude tamo gde očekujete da ga nađete – i, ako nije, treba da preradite kod da bi pomenuti deo bio tamo.
- *Seiso* ili čišćenje (mislite na „sijati“): Na svom radnom mestu ne držite delove žice, ulja, ostatke i otpad. Šta autori kažu o tome da zagađujete svoj kod komentarima i isključenim redovima kodova koje beleže istoriju ili želje za budućnost? Otarasite ih se.
- *Seiketsu*, ili standardizacija: Grupa se slaže o tome kako održati radno mesto čistim. Da li mislite da ova knjiga govori o tome da imate dosledan stil kodiranja i skup pravila za rad u grupi? Odakle dolaze ti standardi? Nastavite sa čitanjem.
- *Shutsuke*, ili disciplina (samodisciplina). To znači da imate disciplinu da pratite pravila prakse koja utiča na vaš rad i spremni ste da se menjate.

Ako prihvatite izazov – da, izazov – čitanja i primene ove knjige, shvatićete i ceniti poslednju tačku. Ovde konačno stižemo do korena odgovornog profesionalizma u profesiji koja bi trebalo da se bavi životnim ciklusom proizvoda. Dok održavamo automobile i druge mašine pod TPM-om, održavanje kvarova – čekanje na greške na površini – je izuzetak. Umesto toga, idemo na viši nivo: svakodnevno pregledamo mašine i zamenujemo potrošne delove pre nego što otkazu ili uradimo ekvivalent promeni ulja na 10.000 kilometara da bismo sprečili habanje. Kôd refaktorišemo nemilosrdno. Možete

poboljšati još jedan nivo, kao što je TPM pokret inoviran pre više od 50 godina: pravite mašine koje su na prvom mestu što održivije. Učiniti vaš kod čitljivim je podjednako važno kao i njegovo izvršenje. Krajnja praksa, uvedena u TPM krugove oko 1960. godine, jeste fokusiranje na uvođenje čitavih novih mašina ili zamenu starih. Kao što nas Fred Brooks opominje, verovatno bismo morali da ponovo napišemo od nule velike komade softvera svakih sedam godina ili da ga očistimo od lošeg koda. Možda bi trebalo da ažuriramo Brooksovu vermensku konstantu na redosled nedelja, dana ili sati umesto godina. U tome leži detalj.

Postoji velika moć u detaljima, ali postoji nešto ponizno i duboko u ovom pristupu životu, kao što to možemo stereotipno očekivati od bilo kojeg pristupa koji ima japanske korene. Ali to nije samo istočni pogled na život; engleska i američka narodna mudrost pune su takvih opomena. Gornji citat *Seiton* izašao je iz pera jednog ministra iz Ohaja koji je urednost doslovno posmatrao „kao lek za svaki stepen zla“. A šta je sa *Seiso*? *Čistoća je pola zdravlja*. Koliko god da je lepa kuća, neuredan radni sto oduzima joj sjaj. Šta kažete na *Shutsuke* o malim stvarima? *Onaj ko je veran u malom, veran je i u velikom*. Kako stojite sa željom da preradite program na vreme, ojačavajući svoju poziciju za naredne „velike“ odluke, umesto da ih odlažete? *Stići na vreme štedite vreme. Korano rani dve sreće grabi. Sve što možete uraditi danas ne ostavljajte za sutra*. (Takav je bio izvorni smisao fraze „poslednji odgovorni trenutak“ u Leanu, sve dok nije pala u ruke softverskih konsultanata.) Šta kažete na zajedništvo malih, pojedinačnih napora u veliku celinu? *Rastu moćni hrastovi iz malih žirova*. Ili kako integrisati jednostavan preventivni rad u svakodnevni život? *Bolje lečiti nego sprečiti. Jedna jabuka na dan, tera doktora iz kuće van*. Jasan i čist kod poštuje duboke korene mudrosti ispod naše popularne kulture ili naše kulture onakvu kakva je nekada bila, ili bi trebalo da bude, i može biti sa pažnjom prema detaljima.

Čak i u velikoj arhitektonskoj literaturi pronalazimo izreke koje ukazuju pažnju detaljima. Pomislite na kvake mesa van der Rohea. To je *seiri*. To je pažnja za svako ime promenljive. Trebalo bi da imenujete promenljivu koristeći istu brigu sa kojom imenujete prvorodeno dete.

Kao što svaki vlasnik kuće zna, takvoj brizi i stalnom održavanju čistoće nikada se ne nazire kraj. Arhitekta Christopher Alexander – otac obrazaca i jezika obrazaca – svaki čin projektovanja vidi kao mali, lokalni čin popravke. On smatra da je izrada fine strukture jedini zadatak arhitekta; veći oblici se mogu prepustiti obrascima i namenama predloženim od strane stanovnika. Projektovanje nije samo dodavanje novih soba kući, već i pažnja posvećena krečenju, zameni dotrajalih tepiha ili dogradnji kuhinjskog sudopera. Većina umetnosti odjekuje analognim osećanjima. U potrazi za drugima koji smatraju da je Bog u detaljima, našli bismo se u dobrom društvu francuskog autora iz 19. veka Gustava Flauberta. Francuski pesnik Paul Valery savetuje nas da pesma nikada nije gotova i traži neprestanu preradu, a prestanak rada na njoj je napuštanje. Takva preokupacija detaljima je zajednička svim koji teže savršenstvu. Možda je ovde malo novog, ali čitajući ovu knjigu naći ćete se pred izazovom da prihvatite dobre discipline koje ste odavno predali apatiji ili želji za spontanošću i prostom „reagovanju na promene“.

Nažalost, obično ne smatramo takvu brigu ključnim temeljem umeća programiranja. Rano napuštamo svoj kodeks, ne zato što je učinjeno, već zato što se naš sistem vrednosti fokusira više na spoljašnji izgled, nego na suštinu onoga što isporučujemo.

Ova nepažnja nas na kraju košta: *Sve dođe na naplatu*. Istraživanje je pokazalo da ni u industriji, ni u akademskim krugovima, nema unižavanja sebe da bi se kod održavao jasnim i čistim. U mojim danima radeći u Organizaciji za istraživanje softverske proizvodnje Bell Labs (zaista *proizvodnja!*) imali smo prateće nalaze koji su sugerisali da je dosledan stil uvlačenja redova koda jedan od statistički najznačajnijih pokazatelja niske gustine grešaka. Želimo da arhitektura ili programski jezik ili neki drugi visoki pojam budu uzrok kvaliteta; kao ljudi čiji navodni profesionalizam dugujemo savladanom alatu i visokim metodama projektovanja, osećamo se uvređeni zbog vrednosti koju te fabričke mašine, odnosno programeri, dodaju jednostavnom doslednom primenom stila uvlačenja. Citiram svoju vlastitu knjigu od pre 17 godina, takav stil razlikuje izvanrednosti od obične sposobnosti. Japanski pogled na svet razume presudnu vrednost svakodnevnog radnika, i više, vrednost razvojnih sistema koji su zasnovani na jednostavnim, svakodnevnim radnjama tih radnika. Kvalitet je rezultat milionskih ličnih postupaka pažnje – ne zbog neke sjajne metode koja je došla sa neba. To što su ovi postupci jednostavni, ne znači da su pojednostavljeni, a teško da znači i da su laki. Ipak, predstavljaju tkanje izvanrednosti, i više od toga, lepote, u bilo kojem ljudskom poduhvatu. Njihovo ignorisanje nije u potpunosti ljudsko.

Naravno, i dalje sam zagovornik šireg razmišljanja, a posebno vrednosti arhitektonskih pristupa koji su ukorenjeni u znanja dubokih domena i upotrebljivosti softvera. Knjiga nije u tome – ili, bar, nije očigledno o tome. Ova knjiga ima suptilniju poruku čiju dubinu ne treba potceniti. Povezana je sa ljudima koji se bave kodom, poput Petera Sommerlada, Kevlina Henney i Giovannija Aspronija. „Program je projekat“ i „Jednostavan kod“ su njihove mantre. Dok moramo da vodimo računa da je interfejs program i da njegove strukture imaju puno toga da kažu o našoj programskoj strukturi, ključno je da kontinuirano zauzmemo skromni stav da projektovanje živi u kodu. I dok prerada u metafori proizvodnje dovodi do troškova, prepravka dizajna dovodi do vrednosti. Naš kod treba da posmatramo kao prelepu artikulaciju plemenitih napora dizajna – dizajna kao procesa, a ne statičke krajnje tačke. U kodu su arhitektonske metrike spajanja i kohezije. Ako slušate kako Larry Constantine opisuje povezanost i koheziju, on govori u smislu koda – a ne o apstraktnim konceptima koji bi se mogli naći u UML-u. Richard Gabriel nas u svom eseju, „Abstraction Descant“, savetuje da je apstrakcija zlo. Kôd je protiv zla, a jasan kôd je verovatno božanski.

Vraćajući se svojoj maloj kutiji Ga-Jola, mislim da je važno primetiti da danska mudrost savetuje da ne samo da treba da obratimo pažnju na male stvari, već i treba da budemo *iskreni* u malim stvarima. To znači da budemo iskreni prema kodu, iskreni prema kolegama o stanju našeg koda i, pre svega, iskreni prema sebi u vezi sa našim kodom. Da li smo dali sve od sebe da „kamp ostavimo čistijim nego što smo ga zatekli“? Da li smo preradili svoj kôd pre prijavljivanja? To nisu periferne brige, već brige koje stoje direktno u centru agilnih vrednosti. U Scrumu je preporučena praksa da ponovno

refaktorisanje bude deo koncepta „Gotovo“. Ni arhitektura ni jasan kod ne insistiraju na savršenstvu, samo na iskrenosti i pružanju najboljeg što možemo. *Grešiti je ljudski; oprostiti, božanski*. U Scrumu sve činimo vidljivim. Provlačimo prljav veš. Iskreni smo o stanju našeg koda jer kôd nikada nije savršen. Postajemo potpuniji ljudi, vredni božanskog i bliži toj veličini u detaljima.

U našoj profesiji očajnički nam je potrebna sva pomoć koju možemo dobiti. Ako čist pod prodavnice smanjuje nesreće, a dobro organizovani alati povećavaju produktivnost, onda sam za sve to. Što se tiče ove knjige, ona je najbolja pragmatična primena Lean principa na softveru koji sam ikada video u štampanoj knjizi. Nisam ništa manje očekivao od ove male grupe mislećih ljudi koji se godinama zajedno trude ne samo da postanu bolji, već i da daruju svoje znanje softverskoj industriji u delima kao što je ovo koje je sada u vašim rukama. Ostavlja svet malo boljim nego što sam ga zatekao pre nego što mi je ujak Bob poslao rukopis.

Pošto sam završio ovaj predgovor sa dubokim uvidima, krećem da čistim svoj sto.

**James O. Coplien**  
Mørdrup, Denmark

---

# Uvod

---

Jedina valjana mera kvaliteta softvera je: Šta je ovo/minut



(c) 2008 Focus Shift

Reprodukovano sa ljubaznom dozvolom Thoma Holwerda.

[http://www.osnews.com/story/19266/WTFs\\_m](http://www.osnews.com/story/19266/WTFs_m)

Koja vrata predstavljaju vaš kod? Koja vrata predstavljaju vaš tim ili vašu kompaniju? Zašto smo u toj sobi? Da li je ovo samo uobičajena provera koda ili smo pronašli niz užasnih problema ubrzo pošto ste kod pustili u primenu? Otklanjamo li greške u panici, opsednuti kodom za koji smo mislili da radi? Da li nas klijenti masovno napuštaju, a



menadžeri nam dišu za vrat? Kako da budemo sigurni da ćemo biti iza pravih vrata kad stvari postanu teške? Odgovor je: zanatstvo.

Postoje dva dela za učenje zanata: znanje i rad. Morate steći znanje o principima, obrascima, praksama i heuristikama koje zanatlija poznaje, a to znanje morate usaditi i u svoje prste, oči i osećanje radeći naporno i vežbajući.

Mogu da vas naučim fiziku vožnje bicikla. Zaista, klasična matematika je relativno jednostavna. Gravitacija, trenje, ugaoni moment, centar mase i tako dalje mogu se prikazati sa manje od jedne stranice jednačina. Tim formulama mogao bih da vam dokažem da je vožnja biciklom praktična i pružio bih vam svo znanje koje vam je potrebno da biste ga vozili. Ali, ipak kada biste prvi put seli na bicikl, odmah biste pali.

Kodiranje se ne razlikuje od vožnje bicikla. Mogli bismo da zapišemo sve dobre principe za pisanje jasnog i čistog koda i onda da vam poverimo da programirate (drugim rečima, da vam dozvolimo da padnete kada sednete na bicikl), ali kakvi bismo onda mi bili nastavnici kada bismo tako učinili, i kakvi biste vi bili učenici?

Ne. To nije način na koji će vas ova knjiga učiti.

Učenje pisanja jasnog koda je *težak posao*. To zahteva više od znanja o principima i obrascima. Morate da se *preznojite* nad tim. Morate sami da vežbate i gledate kako ne uspevate. Morate da gledate kako drugi to rade i ne uspevaju. Morate da vidite kako se spotiču i vraćaju nazad. Morate da vidite njihovu muku u donošenju odluka i cenu koju plaćaju za donošenje pogrešnih odluka.

Budite spremni na naporni rad dok čitate ovu knjigu. Ova knjiga nije lako štivo koje možete pročitati u avionu pre nego što sleti. Ova knjiga će vas naterati da radite *i naporno radite*. Kakvim se poslom bavite? Čitaćete kod – mnogo koda. I izazivaćete vas da razmišljate šta je tačno u kodu a šta nije. Od vas će se tražiti da nas pratite dok razdvajamo module i ponovo ih sastavljamo. Za ovo će trebati vremena i truda; ali mislimo da će biti vredno toga.

Knjigu smo podelili na tri dela. U prvih nekoliko poglavlja opisani su principi, obrasci i prakse pisanja jasnog koda. U ovim poglavljima postoji prilično malo koda, pa će biti izazov pročitati ih. Pripremiće vas za drugi deo. Ako knjigu odložite nakon što pročitate prvi deo, želimo vam srećan put!

Drugi deo knjige je teži posao. Sastoji se od nekoliko studija slučaja stalno rastuće složenosti. Svaka studija slučaja je vežba čišćenja nekog koda – transformacije koda koji ima nekih problema – u kod koji ima manje problema. Detalji u ovom delu su *intenzivni*. Moraćete da prelazite između narativa i listinga programa. Moraćete da analizirate i razumete kod sa kojim radimo i da razumete naša obrazloženja za svaku promenu koju učinimo. Odvojte malo vremena jer *ovo bi moglo da potraje*.

Treći deo ove knjige predstavlja naplatu. To je jedno poglavlje koje sadrži spisak heuristike i „mirisa“ prikupljenih tokom stvaranja studija slučaja. Dok smo prolazili i čistili kod u studijama slučaja, dokumentovali smo svaki razlog za naše postupke za heuristiku ili za „miris“. Pokušali smo da razumemo sopstvene reakcije na kôd koji smo čitali i menjali i naporno smo radili da shvatimo zašto osećamo šta osećamo i zašto radimo to što radimo. Rezultat je baza znanja koja opisuje način na koji razmišljamo kada pišemo, čitamo i čistimo kod.

Ova baza znanja ima ograničenu vrednost ako ne radite pažljivo čitajući studije slučaja u drugom delu ove knjige. U tim studijama slučaja pažljivo smo označili svaku promenu koju smo izvršili uz upućivanje na heuristiku. Ove prednje reference su prikazane u uglastim zagradama, kao što je ovo: [H22]. To vam omogućava da vidite *kontekst* u kojem su te heuristike primenjene i napisane! Nisu heuristike same po sebi toliko vredne, to je *odnos između te heuristike i diskretnih odluka koje smo doneli dok smo čistili kod u studijama slučaja*.

Da bismo vam dodatno pomogli u tim odnosima, na kraju knjige smo dali unakrsnu referencu koja prikazuje broj stranice za svako upućivanje napred. Možete da ih upotrebite da potražite gde je primenjena određena heuristika.

Ako pročitate prvi i treći deo i preskočite studije slučaja, tada ćete pročitati još jedno lako štivo o pisanju dobrog softvera. Ali ako odvojite vreme da proučite studije slučaja, prateći svaki sitan korak, svaku pravovremenu odluku, ako se postavite na naše mesto i naterate se da razmišljate duž iste staze kojom smo i mi mislili, tada ćete steći mnogo bogatije razumevanje tih principa, obrazaca, praksi i heuristika. Sve to više neće biti „znanje“ iz lakog štiva. Biće vam u telu, prstima i srcu. Sve to će postati deo vas na isti način na koji bicikl postaje produžetak vaše volje kada naučite da ga vozite.

## Zahvalnosti

Zahvaljujem se svojim dvema umetnicama, Jeniffer Kohnke i Angeli Brooks. Jennifer je odgovorna za zapanjujuće i kreativne slike na početku svakog poglavlja, kao i za portrete Kent Becka, Vard Cunninghama, Bjarne Stroustrupa, Ron Jeffriesa, Grady Boocaha, Dave Thomasa, Michael Feathersa i mene.

Angela je odgovorna za pametne slike koje krasi unutrašnjost svakog poglavlja. Tokom godina napravila je za mene priličan broj slika, uključujući mnoge slike u *Agile Software Development: Principles, Patterns, and Practices*. Ona je takođe moje prvo dete sa kojim sam veoma zadovoljan.

Posebnu zahvalnost upućujem mojim recenzentima Bobu Bogettiju, Georgeu Bullocku, Jeffreyu Overbeiu, a posebno Mattu Heusseru. Bili su brutalni. Bili su okrutni. Oni su bili neumoljivi. Snažno su me gurali da napravim neophodna poboljšanja.

Zahvaljujem svom izdavaču, Chris Guzikovskom, na podršci, ohrabrenju i radošnom izražavanju. Hvala i redakciji u Pearsonu, uključujući Rainu Chrobak što me je tretirala iskreno i tačno.

Hvala Micah Martinu i svim momcima iz 8th Lights ([www.8thlight.com](http://www.8thlight.com)) na proverama i ohrabrenju.

Hvala svim mentorima, prošlim, sadašnjim i budućim, uključujući: Bob Koss, Michael Feathers, Michael Hill, Erik Meade, Jeff Langr, Pascal Roy, David Farber, Brett Schuchert, Dean Wampler, Tim Ottinger, Dave Thomas, James Grenning, Brian Button, Ron Jeffries, Lowell Lindstrom, Angelique Martin, Cindy Sprague, Libby Ottinger, Jo-leen Craig, Janice Brown, Susan Rosso i ostalima.

Hvala Jimu Newkirku, mom prijatelju i poslovnom partneru, koji me je naučio više nego što mislim da jeste. Hvala Kent Becku, Martin Fowleru, Ward Cunninghamu, Bjarnr Stroustrupu, Grady Boochu, i svim drugim mojim mentorima i zemljacima. Hvala John Vlissidesu što je bio sa nama kada je trebalo. Zahvaljujem momcima u Zebri što su mi dozvolili da dižem dreku o tome koliko dugačka treba da bude funkcija.

I, konačno, hvala vam što ste pročitali ove zahvalnice.