# Sadržaj

# 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*?

Otprilike 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 („sistematizovati"). 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 zamenjujemo potrošne delove pre nego što otkažu 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*. *Ko rano 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 miesa van der Rohea. To je *seiri*. To je pažnja za svako ime promenljive. Trebalo bi da imenujete promenljivu koristeći istu brigu sa kojom imenujete prvorođeno 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 arhitekte; 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 prihatite 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 koncepatima 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


Reprodukovano sa ljubaznom dozvolom Thoma Holwerda.
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 Boocha, Dave Thomasa, Michael Feathersa i mene.

Angela je odgovorna za pametne slike koje krase unutrašnjost svakog poglavlja. Tokom godina napravila je za mene priličan broj slika, uključujući mnoge slike u *Agile Software Develpment: 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 Guzikowskom, na podršci, ohrabrenju i radosnom 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) 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, Joleen 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.