

1 - Uvod

*Ne pretvaram se da znamo sve odgovore.
Ali pitanja svakako zaslužuju da se o njima razmisli.*

Artur Klark

Vežite se...

Nakon što nauče osnove programiranja i upoznaju veći broj opštih i specifičnih algoritama, mladi programeri često imaju utisak da su naučili da prave softver. Ali, stvari stoje drugačije. Time što je naučeno programiranje, samo je obezbeđena ulaznica u svet razvoja softvera. Put do dobrih rezultata je dugačak i ispunjen brojnim drugim aktivnostima, a ne samo programiranjem.

Mnogo softvera se piše ad-hok – bez velikih planova i namera; za jednokratnu upotrebu, da bi se rešio neki tekući problem. U takvim slučajevima pojmovi „softver“ i „program“ se često ne razlikuju – sve što pišemo je jedan jednostavan program, čije pisanje često ne zahteva ni posebne tehnike ni disciplinu u radu. Zato ponekad može da se učini da nema mnogo razlike između pravljenja softvera i programiranja, ali to je veoma daleko od istine. Već u slučajevima kada je potrebno da se napiše ne samo jedan jednostavan program nego više njih, i to tako da njihova upotreba bude jasna, logična i međusobno usklađena, stvari postaju složenije; a kada poraste i složenost programa koje pišemo, onda prelazimo u sasvim drugi prostor problema. Što je softver složeniji to njegovo pravljenje ima više različitih činilaca, bez kojih softver ne može da se efikasno i kvalitetno pravi, upotrebljava i održava.

Obično se najpre uočava da je neophodno da se pravi i održava neki vid dokumentacije. Dokumentacija je, pre svega, sredstvo komunikacije, kojim se prenose informacije o upotrebi softvera od razvijalaca prema korisnicima, ali i razmenjuju informacije o strukturi i implementaciji softvera među njegovim

razvijaočima. Iz ugla programera, dokumentaciju čine i formalne specifikacije delova softvera, različiti projekti, planovi, definicije interfejsa i drugo.

Ispostavlja se da je programiranje samo jedna od brojnih aktivnosti koje se preduzimaju tokom razvoja softvera. Štaviše, programiranje u okviru kompleksnijeg procesa razvoja softvera postaje mnogo drugačije i složenije od programiranja koje smo prethodno upoznali – nije više dovoljno da se napiše program, nego pri tome moraju da se prate određena pravila, a delovi programa koje pišemo moraju da budu u skladu sa nekim ustanovljenim principima.

Pored programiranja, pravljenje softvera obuhvata i sve one pripremne korake, koji prethode pisanju programa, kao i sve one postupke koji tokom ili nakon pisanja programa pomažu da se programiranje izvede što kvalitetnije i koji obezbeđuju da napisani programi predstavljaju što dugotrajniju vrednost. U nekom jednostavnijem slučaju, ti dodatni poslovi bi možda mogli da se ne uoče, zato što ih je malo ili su sasvim jednostavni, ali u složenijim slučajevima razvoja softvera, ne samo da je tih dodatnih poslova mnogo, nego oni čak postaju veoma složeni i obimni, tako da se relativno često događa da se njima posvećuje više pažnje i vremena nego programiranju.

Različite aktivnosti se tokom rada povezuju, kombinuju i prepliću, pa ih nije uvek lako ni prepoznati ni nabrojati. Skup tih aktivnosti zavisi od ugla posmatranja i primenjene razvojne metodologije, ali i specifičnosti problema koje rešavamo, načina grupisanja različitih poslova u celine, složenosti projekta na kome radimo i mnogih drugih faktora. Radi ilustracije, mogli bismo, na primer, da izdvojimo sledeće aktivnosti:

- prikupljanje informacija o ciljevima i potrebama;
- analiziranje problema;
- prepoznavanje i oblikovanje zahteva;
- projektovanje softvera;
- oblikovanje korisničkog interfejsa;
- implementiranje softvera;
- testiranje softvera;
- održavanje softvera;
- planiranje resursa;
- planiranje i praćenje toka razvoja;
- staranje o kvalitetu softvera;
- upravljanje razvojnim procesom;
- uređivanje tehničke i korisničke dokumentacije;

- pravljenje i uređivanje fotografija, ilustracija, video i zvučnih zapisa za potrebe softverskog projekta;
- pripremanje i pakovanje proizvedenog softvera za distribuiranje i instaliranje
- i drugo.

Ovaj spisak bi mogao da bude kraći ali i duži. Na primer, testiranje je navedeno kao posebna aktivnost iako bi moglo da se svrsta u staranje o kvalitetu softvera, a delimično i u implementiranje softvera. Nasuprot tome, testiranje bi moglo i da se podeli na različite vrste testiranja i da se zameni sa više različitih aktivnosti.

Na ovom spisku naizgled nema programiranja – ono namerno nije eksplicitno navedeno, već se pretpostavlja da je sadržano u okviru implementiranja softvera. To je urađeno da bi se dodatno istakla činjenica da je programiranje samo jedan od činilaca razvoja softvera. Štaviše, u nekim posebnim slučajevima imamo primere razvoja softvera u kojima uopšte nema programiranja. Na primer, ako upakujemo gotove programe, određena uputstva za konfigurisanje i neke dodatne sadržaje (slike, tekstove, video ili audio zapise), onda možemo da dobijemo nov softverski proizvod, a da pri tome nismo napisali nijedan nov red programa. Važno je da razumemo da postoje i takvi projekti, iako se u daljem tekstu bavimo isključivo razvojem softvera u kome je najvažniji deo pravljenje programskih elemenata.

Softversko inženjerstvo

Preduzimanje velikog broja različitih aktivnosti pri pravljenju softvera ima za posledicu da u tom poduhvatu moraju da učestvuju razvijaoci softvera koji imaju veliki broj različitih specijalnosti. Brojne aktivnosti i rad većeg broja ljudi nije lako organizovati, pa je neophodno da se u tom radu primeni sistematičan pristup, koji počiva na primeni naučnih rezultata, ali i dobrih prethodnih iskustava – inženjerstvo.

Inženjerstvo je praktična primena naučnih saznanja i matematike pri rešavanju problema i pravljenju proizvoda. Pri tome se aktivno koriste prethodna iskustva,oličena u pravilima ili standardima, a pomoću kojih se obezbeđuje odgovarajući nivo kvaliteta napravljenog rezultata. U skladu s tim se definiše i termin *softversko inženjerstvo* [IEEE 2017]:

Softversko inženjerstvo je primena sistematskog, disciplinovanog i merljivog pristupa razvoju, funkcionisanju i održavanju softvera; odnosno primena inženjerstva na softver

IEEE

Iz navedene definicije se vidi da se softversko inženjerstvo odlikuje sistematičnim i disciplinovanim inženjerskim pristupom problemu proizvodnje softvera. Pri tome je i sveobuhvatno, u smislu da se odnosi na *sve* aktivnosti koje se preduzimaju pri proizvodnji softvera.

Da bismo se lakše nosili sa složenošću proizvodnje softvera i softverskog inženjerstva, obično se problemi posmatraju iz različitih aspekata. Tri najvažnija pristupa su iz aspekta procesa, metoda i kvaliteta¹. Svaki od ovih aspekata nam pruža odgovarajući deo odgovora na pitanja razvoja softvera, ali nam tek svi zajedno pružaju kompletne odgovore na probleme sa kojima se suočavamo.

Jedan od najvažnijih aspekata softverskog inženjerstva predstavlja upravljanje razvojnim procesom i njegovo redovno praćenje i nadziranje. Možemo da kažemo da nam posmatranje problema iz ugla procesa pruža odgovore na pitanja „Šta radimo?“ i „Kada to radimo?“.

Odvijanje različitih aktivnosti tokom proizvodnje softvera se naziva *proces razvoja softvera*, ili *razvojni proces*. Termin *proces* se upotrebljava kako da označi celokupan razvojni proces, tako i da označi odvijanje manjeg broja aktivnosti (ili čak pojedinačnih aktivnosti) u nekom segmentu razvojnog procesa. Pri planiranju i upravljanju razvojnim procesom mora da se ima u vidu šta su preduslovi za odvijanje određenih aktivnosti i šta su rezultati tih aktivnosti. Sagledavanjem svih potrebnih aktivnosti stiču se uslovi za planiranje rasporeda njihovog odvijanja.

Drugi važan aspekt softverskog inženjerstva čini planiranje i odabiranje skupa metoda, tehnika i alata koji će se upotrebljavati na nekom projektu. Ako nam je upravljanje procesom odredilo šta se i kada radi, onda nam planiranje metoda, tehnika i alata odgovara na pitanje „Kako to radimo?“.

Pri obavljanju svake od aktivnosti, koja čini razvojni proces, mogu da se upotrebljavaju različite *tehnike* i odgovarajući *alati*. Da bi rezultati jedne aktivnosti bili upotrebljivi u nekoj drugoj, neophodno je da se usklade alati i tehnike koji se koriste u tim aktivnostima. Za obavljanje istog posla često mogu da se upotrebe različite tehnike, kao što i neke tehnike mogu da se upotrebljavaju u većem broju različitih aktivnosti. U razvoju softvera se često upotrebljava i termin *metod*. U nekim slučajevima se ne pravi značajna razlika između termina metod i tehnika. Ipak, uglavnom je uobičajeno da se terminu metod dodeljuje nešto šire značenje, pa možemo da kažemo da metod predstavlja specifičan način primene jedne ili više tehnika u okviru nekog dela razvojnog procesa ili pojedinačne aktivnosti. Na primer,

¹ U literaturi se ovi aspekti nazivaju i slojevima softverskog inženjerstva. Takođe, metodi se negde zamenjuju alatima i tehnikama. U svakom slučaju, suština je ista.

specifičan način primene tehnike testiranja jedinica koda predstavlja osnovu metoda razvoja vođenog testovima.

Treći aspekt softverskog inženjerstva je staranje o kvalitetu. Ne bismo mnogo pogrešili ako bismo rekli da je staranje o kvalitetu najvažniji aspekt softverskog inženjerstva, a da ostali aspekti postoje samo da bi staranje o kvalitetu moglo da se ostvari u punoj meri. Staranje o kvalitetu je često najprepoznatljivija razlika između sistematičnog i disciplinovanog razvoja softvera i amaterskog pisanja programa. Dok je u amaterskim poduhvatima često dovoljno da program „proradi“ i isporuči odgovarajuće rezultate, u komercijalnoj proizvodnji softvera je od presudnog značaja da rezultat zadovolji definisane kriterijume kvaliteta. Bez staranja o kvalitetu, svi drugi činioци softverskog inženjerstva postaju bezvredni.

Da bi razvojni proces mogao da se lakše razume i planira, a zbog velikog broja različitih aktivnosti o kojima mora da se vodi računa, obično pokušavamo da ovaj problem malo pojednostavimo tako što grupišemo aktivnosti u nekoliko većih celina. Ima više načina da se to uradi. Jedna od mogućih podela je prema osnovnom poslu kome pripadaju aktivnosti, pa tako razlikujemo sledeće celine:

- **planiranje softvera** – često se naziva i specifikacija softvera; obuhvata sve aktivnosti koje se odnose na prikupljanje informacija o ciljevima, sagledavanje i definisanje zahteva i potrebnih karakteristika softvera, procenu potrebnih resursa i druge poslove neophodne za odgovaranje na pitanje *šta se pravi*;
- **projektovanje i implementiranje softvera** – obuhvata sve aktivnosti u vezi projektovanja i implementiranja, uključujući i programiranje; ponekad se naziva i *razvoj softvera*;
- **staranje o kvalitetu** – sve aktivnosti koje imaju za cilj obezbeđivanje potrebnog nivoa kvaliteta, ali i proveravanje da li je taj nivo kvaliteta ostvaren; u velikoj meri se prepiće sa drugim celinama;
- **održavanje softvera** – aktivnosti koje imaju za cilj da produže vrednost softvera, kroz prilagođavanje izmenjenim zahtevima korisnika ili tržišta; često se tu ubrajaju i aktivnosti u vezi sa pripremanjem softvera za rad, konfigurisanjem, obučavanjem korisnika i slično.

Nešto drugačiji pristup predstavlja grupisanje poslova u tzv. „okvirne aktivnosti“, uz konstatovanje da se u većim razvojnim projektima one odvijaju iterativno, u razvojnim ciklusima [Pressman 2020]:

- **komunikacija** – razmena informacija sa klijentom i prikupljanje potrebnih informacija da bi moglo da se započne planiranje; cilj je razumevanje ciljeva i potreba klijenta;

- **planiranje** – pravljenje okvirnog plana realizacije projekta, sa analizom rizika i okvirnom procenom potrebnih resursa; pravljenje okvirnog rasporeda aktivnosti;
- **modeliranje** – projektovanje novog proizvoda; preciznost i obim projekta zavise od specifičnosti konkretnog proizvoda;
- **konstrukcija** – izgradnja softvera obuhvata detaljno projektovanje i implementiranje; savremene metodologije podrazumevaju da se tokom konstrukcije izvodi i značajan deo testova kvaliteta;
- **pripremanje za rad** – proizveden softver mora da se pripremi za rad; preduzimaju se različite aktivnosti u zavisnosti od vrste i složenosti softvera.

Možemo da uočimo neke sličnosti i razlike između ovih podela, koje mogu da nam pomognu da razumemo složenost i obim razvojnog procesa. Na primer, u prvoj podeli nije navedena „komunikacija“, već se pretpostavlja da je ona sastavni deo „planiranja“, ali se danas uglavnom očekuje da se komunikacija odvija tokom čitavog razvojnog procesa; u prvoj podeli su „projektovanje i implementiranje“ objedinjeni, a u drugoj su podeljeni na „modeliranje“ i „konstrukciju“; prva podela ima izdvojenu grupu „staranje o kvalitetu“, a u drugoj se ona ne pojavljuje kao samostalna grupa, već se podrazumeva da se staranje o kvalitetu prepiće sa svim ostalim aktivnostima; prva podela prepoznaje „održavanje“, a u drugoj se pretpostavlja da je održavanje softvera samo još jedan razvojni ciklus, koji čine sve navedene okvirne aktivnosti.

Razvoj softvera

Termin *razvoj softvera* se koristi još češće od softverskog inženjerstva². Razvoj softvera i softversko inženjerstvo imaju mnogo toga zajedničkog i relativno često možemo da upotrebimo bilo koji od ova dva termina, bez bojazni da će doći do nesporazuma. Ipak, za razliku od termina softversko inženjerstvo, čija je upotreba uglavnom ujednačena, termin razvoj softvera se u literaturi i praksi upotrebljava u različitim kontekstima i sa različitim značenjem.

Kada govorimo o konkretnom softverskom projektu, razvoj softvera, u najširem smislu, obuhvata doslovno sve aktivnosti koje se odnose na pravljenje softvera, pa i sam razvojni proces. Upotrebljava se i kao sinonim za pravljenje softvera ili proizvodnju softvera. U skladu s tim, ako se govorи o disciplini razvoja softvera,

² Na primer, Google pronađi oko 117.000.000 rezultata ako tražimo „software engineering“ i oko 202.000.000 rezultata ako tražimo „software development“.

onda možemo da kažemo da ona, u najširem smislu, predstavlja sinonim za softversko inženjerstvo.

Nasuprot tome, u najužem smislu, razvoj softvera se odnosi samo na poslove projektovanja i implementiranja softvera, pa i disciplina razvoj softvera obuhvata samo poslove koji čine projektovanje i implementiranje softvera. Takvo određenje ovog termina je prilagođeno klasičnim razvojnim metodologijama, u kojima su različiti poslovi bili jasno razdvojeni po fazama. Tada su i poslovi projektovanja i implementacije bili jasno odvojeni i međusobno i od ostalih poslova, kako vremenski tako i po specijalnostima subjekata koji su obavljali ove poslove. Međutim, takvo organizovanje razvojnog procesa nam danas donosi više problema nego koristi, pa se zato i termin razvoj softvera relativno retko koristi sa ovakvo uskim značenjem.

U daljem tekstu ćemo da podrazumevamo da u kontekstu razvojnog projekta termin razvoj softvera ima najšire značenje, a da se disciplina razvoj softvera odnosi prvenstveno na aktivnosti koje su neposredno ili relativno blisko povezane sa projektovanjem i implementiranjem softvera. U odnosu na ranije prepoznate grupe poslova, jasno je da je fokus discipline razvoja softvera prvenstveno na grupi „projektovanje i implementiranje softvera“ (ili na grupama „modeliranje“ i „konstruisanje“, zavisno od toga koju podelu razmatramo). Međutim, važno je da primetimo da i u ostalim grupama poslova postoje brojne aktivnosti koje su relativno blisko povezane sa projektovanjem i implementiranjem i koje se takođe razmatraju kao činioci razvoja softvera. Na primer, definisanje interfejsa komponenti se svrstava i u planiranje i u projektovanje; testiranje jedinica koda se jednakostvorno u programiranje i u staranje o kvalitetu; sve aktivnosti u vezi sa projektovanjem i implementiranjem su zastupljene i u održavanju softvera i drugo.

Takvo tumačenje discipline razvoja softvera, koje je negde između dva ekstrema, danas preovladava i u literaturi i u praksi. Posebno je zgodno što je ono prilagođeno savremenim agilnim razvojnim metodologijama. Danas se različite razvojne aktivnosti prepliću i vremenski i sadržajem, a od razvijalaca se očekuje da imaju široka znanja i da su sposobljeni za mnoge aktivnosti, koje nisu neposredno povezane sa projektovanjem i programiranjem, ali sa njima imaju dodirnih tačaka.

Softverski inženjeri i razvijaoci softvera

Posledica razlikovanja softverskog inženjerstva i razvoja softvera bi trebalo da bude i odgovarajuće razlikovanje termina *softverski inženjer* i *razvijalac softvera*, ali savremena praksa tu nije dosledna. Ovi pojmovi se često izjednačavaju, ali se u drugim prilikama nailazi i na raznovrsna tumačenja njihovih razlika.

Savremene metodologije u velikoj meri počivaju na visokom nivou stručnosti i samostalnosti svih članova razvojnog tima, pa je kvalitetno i široko obrazovanje jedan od osnovnih kriterijuma za angažovanje razvijalaca. U savremenoj praksi se zvanje softverskog inženjera često dodeljuje praktično svim razvijaocima koji rade na

razvoju programskih elemenata softvera, po čemu se oni razlikuju od razvijalaca koji se bave drugim aspektima projekta (na primer oblikovanjem korisničkog interfejsa, tehničkom pripremom dokumentacije, pravljenjem i uređivanjem ilustracija, video ili zvučnih zapisa i slično).

U skladu sa ranije usvojenim značenjem termina razvoj softvera, u daljem tekstu ćemo podrazumevati da je razvijalac softvera svako ko se bavi poslovima projektovanja i implementiranja softvera, kao i poslovima koji sa njima imaju dodira.

Metodologije

U razvoju softvera se koriste mnoge tehnike i aktivnosti, ali one svejedno nisu dovoljne, pa se zato neprekidno usavršavaju i razvijaju nove. Veliki broj alata, tehnika i aktivnosti otežava snalaženje među njima i odabiranje odgovarajućih metoda za rešavanje nekog konkretnog problema. Dodatni problem je što pri izboru načina rešavanja jednog problema moramo da vodimo računa i o tome kako ćemo rešavati neke druge probleme u sklopu istog projekta, zato što će njihovi ulazni sadržaji i rezultati morati da se uskladjuju.

Da bi se olakšalo donošenje pojedinačnih odluka tokom rada na projektu, uobičajeno je da se na samom početku razvojnog procesa, najpre ustanove neki skup principa rada i odgovarajući skupovi metoda i procesa, koje ćemo kasnije dosledno primenjivati tokom čitavog projekta. Takvo lokalizovanje donošenja nekih značajnih odluka na samom početku projekta ima i dobre i loše strane. Naravno, dobro je što ćemo kasnije lakše i brže da donosimo odluke. Dobro je i što će biti lakše da se uskladjuju metodi i procesi između različitih delova projekta. Ali loše je što onda moramo da napravimo izbor već na samom početku projekta, kada možda još uvek nismo svesni svih pojedinačnih karakteristika problema sa kojim se suočavamo. Štaviše, dok pri rešavanju pojedinačnih problema možemo da biramo metode i procese iz nekog manjeg skupa onih koji se odnose na tu vrstu problema, pravljenje opštег odabira na počeku projekta predstavlja daleko teži i obimniji posao.

Prirodno se postavlja pitanje da li bismo mogli da ponovimo sličnu stvar – kao što smo pokušali da olakšamo izbor metoda za pojedinačne probleme, prebacujući ih na nivo projekta, možda bismo mogli da olakšamo i izbor na nivou projekta tako što bismo ga napravili objedinjeno za veći skup projekata? Iz takvog pristupa izrasta koncept metodologije.

Metodologija razvoja softvera (ili *razvojna metodologija*) je preporučeni skup metoda i procesa, koji su međusobno usklaćeni i omogućavaju uspešnu realizaciju razvojnog projekta.

Metodologija predstavlja okvir u kome se odvijaju planiranje razvojnog procesa i odabiranje načina rešavanja pojedinačnih problema. Osnovna uloga metodologije je da olakša sistematičan pristup razvoju softvera, tako što na određeni način sužava izbor i preporučuje puteve i alate kojima se valja služiti. Metodologija može da bude

veoma stroga i da precizno određuje i redosled koraka i način njihovog odvijanja i svaki pojedinačan alat. Nasuprot tome, postoje i manje stroge metodologije, koje propisuju samo neke okvirne principe kojih bi trebalo da se pridržavamo tokom razvoja, a ostavljaju relativno veliku slobodu pri izboru konkretnih alata. Takve metodologije se uobičajeno nazivaju *okvirnim metodologijama*, pa se čak kaže i da nisu prave metodologije ili da su *nekompletne*.

Razvoj i oblikovanje metodologija je vid spoja računarskih nauka i softverksog inženjerstva. Tokom istorije računarstva nastalo je mnogo različitih metodologija. Svaka od njih je osmišljena da bi u datim okolnostima neka vrsta problema mogla da se reši bolje nego do tada postojećim metodologijama. Vremenom su se menjale okolnosti u kojima se razvija softver, pa su se menjale i metodologije. Neke od najvažnijih vrsta metodologija su procesne, strukturne, objektno-orientisane i agilne.

Procesne metodologije su u centar pažnje stavljače životni ciklus softvera i posvećivanje procesima u domenu. Modeliranje procesa je na kraju imalo za rezultat algoritme, na osnovu kojih su se pravili programi. Životni ciklus je imao različite oblike, od kojih je najpoznatiji tzv. *model vodopada*³, po kome su se ključni poslovi (na primer: istraživanje, analiziranje, planiranje, projektovanje, implementiranje, testiranje, pripremanje za rad) odvijali u definisanom redosledu i u jasno razdvojenim fazama, tako da je projekat prolazio kroz te faze u jednom smeru, kao što voda teče niz vodopad.

Strukturne metodologije su prepoznavale da se softverski sistemi grade nad nekoliko vrsta strukturnih elemenata (subjekti, procesi, podaci i interfejsi) i usmeravale su razvoj tako da u centru pažnje bude ostvarivanje dobre unutrašnje strukture sistema. Imale su dosta sličnosti, ali i razlika, u odnosu na procesne metodologije, pa su vremenom nastale i tzv. kombinovane metodologije, koje su predstavljale kombinaciju procesnih i strukturnih.

Objektno-orientisane metodologije (OOM) su nastale kao vid nadgradnje objektno-orientisanog programiranja (OOP). Kao što je kod OOP akcenat na opisivanju objekata i klasa u domenu programa, tako je kod OOM akcenat na opisivanju objekata i klasa u domenu problema. Teži se da se svi elementi posmatranog problema opišu u terminima objekata, klase, metoda i interfejsa, sa idejom da se tako, uz primenu enkapsulacije, objedinjeno opisuju i struktura i procesi. Na sličan način se modeliraju posmatrani problemi u prostoru domena i implementacija u prostoru rešenja. Ovakve metodologije su danas veoma

³ Čak je toliko poznat da se često koristi i pogrešan termin *metodologija vodopada*. Ovaj termin nije dobar zato što je životni ciklus vodopada zastupljen u više različitih metodologija, t.j. ne podrazumeva neki konkretan skup metoda.

zastupljene u praksi. Veliki broj novih projekata se zasniva na primeni neke od objektno-orientisanih metodologija.

Agilne metodologije su nastale usled potrebe da se dinamika razvoja softvera prilagodi dinamici promena u domenu za koji se softver pravi. Osnovni problem sa prethodnim metodologijama je bio što razvoj softvera može da traje veoma dugo, a u međuvremenu mogu da se promene okolnosti, pa usled toga i ciljevi i zahtevi koji se postavljaju pred razvojni projekat. Agilne metodologije počivaju na pretpostavci da će se zahtevi menjati i sve ostale aspekte u značajnoj meri podređuju toj pretpostavci. Neke od agilnih metodologija predstavljaju samo okvirne metodologije, pa se u praksi primenjuju uz kombinovanje sa drugim metodologijama (najčešće OOM), iz kojih se preuzimaju metodi i procesi.

Jedna od najvažnijih karakteristika agilnih metodologija je da su dobro prilagođene održavanju softvera. Štaviše, one kao jedan od važnijih kriterijuma kvaliteta prepoznaju lakoću održavanja softvera, odnosno njegovu trajnost. Za razliku od fizički opipljivih vrsta proizvoda, softver nije podložan starenju. Građevine, mašine i računarski hardver mogu da se oštete pod uticajem spoljašnjih faktora i mogu da se troše dok obavljaju svoju funkciju, pa se sa protokom vremena lakše i češće kvare. Softver ima sasvim drugačiju prirodu, pa na njega protok vremena nema neposrednog uticaja. Ali ako softver ne stari, to ne znači da je večan. Softver se ne troši, ali mu kvalitet ipak postepeno opada [Pressman 2020]. Vremenom se menjaju uslovi u kojima se softver upotrebljava i njegova vrednost postepeno opada – menjaju se računarski sistemi na kojima se koristi, ili se menja oblik ulaznih ili izlaznih podataka, ili se menjaju spoljašnje okolnosti tako da korisnici od njega očekuju više nego što on može da ponudi. Zbog toga softver povremeno mora da se usklađuje sa potrebama i okolnostima – tj. da se *održava*. Svaki pojedinačan ciklus održavanja softvera je veoma sličan razvojnom procesu – potrebno je prvo da se isplanira šta će da se menja, pa da se isprojektuje i implementira i na kraju da se proveri da li je sve urađeno u skladu sa planovima i potrebama.

Često se upotrebljava i termin *klasične metodologije*, ali sa dva različita značenja. Kada se govori o agilnim metodologijama, termin klasične metodologije se obično odnosi na sve metodologije koje nisu agilne. Međutim, ovaj termin se često upotrebljava i da predstavi sve metodologije koje su prethodile objektno-orientisanim metodologijama. Razlika između ova dva značenja je samo u tome da li se OO metodologije koje nisu agilne smatraju za klasične ili ne. Uporedo sa ovim terminom koristi se i termin *tradicionalne metodologije*, sa istim značenjima.

Polećemo!

Ova knjiga se bavi izabranim temama iz oblasti razvoja softvera. Najviše pažnje je posvećeno problemima projektovanja i implementiranja softvera, ali je značajan deo prostora posvećen i nekim opštijim metodološkim pitanjima.

Danas se smatra da je neophodno da praktično svi razvijaoci softvera raspolažu solidnim poznavanjem osnovnih tema iz oblasti struktturnog projektovanja. Zato je oblast projektovanja softvera zastupljena u obimu koji omogućava čitaocima da naprave prve korake u toj oblasti i da zahvaljujući stečenim znanjima budu u stanju da oblikuju i pišu kvalitetnije programe. To svakako nije dovoljno da bi neko mogao da se nazove projektantom softvera, ali bi trebalo da posluži kao odskočna daska za dalje usavršavanje.

Preplitanje različitih metoda i procesa u savremenom razvoju suočava razvijaoce sa obavezom da dobro razumeju sve različite aspekte posla koji obavljaju, a da bi se dobro razumelo kombinovanje aspekata razvoja, kao i da bi se razumeo širi kontekst razvoja softvera, neophodno je poznavanje osnovnih metodoloških tema.

Predstavljeni su izabrani metodi i tehnike implementiranja softvera. Jedan od važnijih kriterijuma izbora je bio da su svi ti metodi i pripadajuće tehnike relativno bliski sa staranjem o kvalitetu softvera. Sa druge strane, izloženi sadržaji su povezani i međusobno i sa izloženim temama iz oblasti projektovanja softvera, tako da predstavljaju jednu zaokruženu i relativno čvrsto povezanu celinu.

Razvoj softvera zahteva disciplinu i sistematičnost u radu. Zbog toga neke od tehnika mogu da izgledaju kao dosadne liste pravila ili koraka koje je potrebno naučiti i zatim preduzimati. Međutim, svaki razvojni projekat je priča za sebe, sa svojim specifičnim problemima i prioritetima, zbog čega mnoge tehnike i principi ne mogu da se na isti način primenjuju u svakom kontekstu. Zato cilj izučavanja predstavljenih tehika i principa nije da se odgovarajuća pravila ili postupci nauče napamet i zatim dosledno i nekritički primenjuju, nego da se razume zašto i kako su ta pravila i postupci oblikovani i kako možemo da ih prilagođavamo specifičnostima konkretnih projekata.

Pored aktivnosti koje čine razvoj softvera, softversko inženjerstvo obuhvata i brojne specifične poslove, kao što su poslovi planiranja i kontrole kvaliteta celovitog proizvoda, poslovi oko konfigurisanja i puštanja softvera u rad, brojni tehnički poslovi na usklađivanju aktivnosti, poslovi staranje o resursima i vođenja projekata i drugo. U ovoj knjizi se uglavnom nećemo baviti takvim specifičnim činiocima softverskog inženjerstva. Za temeljnije izučavanje ovih tema, ali i softverskog inženjerstva kao celovite i zaokružene oblasti, upućujemo čitaoce, pre svega, na [Pressman 2020], a zatim i na [Sommerville 2016, Pfleeger 2006, Popović 2019].

ÚPPŘÍPŘEML