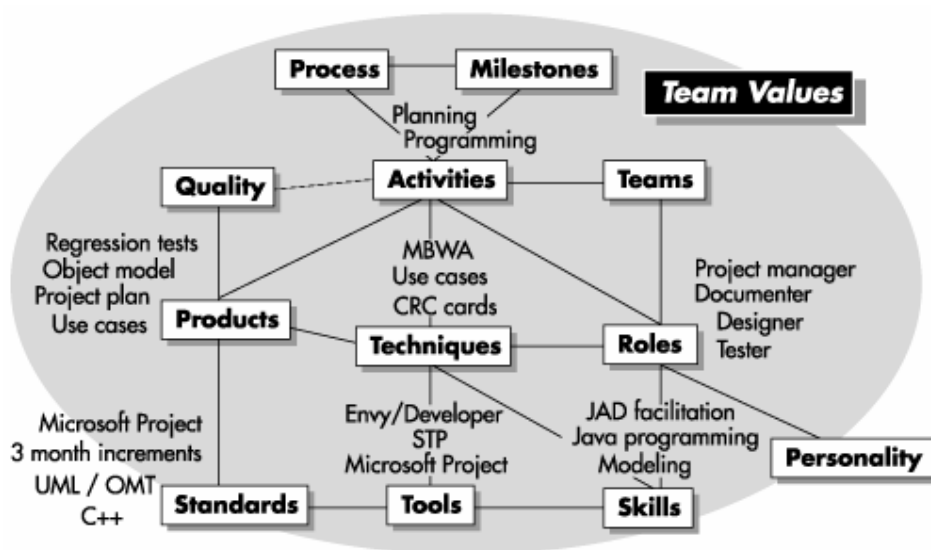


16 Metodologije za programiranje i razvoj softvera

Metodologija se može definisati kao “skup međusobno povezanih metoda i tehnika“, gde se pod metodama podrazumevaju „sistematične procedure“ slične tehnikama.

Metodologije za razvoj softvera bave se „metodama i tehnikama“ za razne elemente koji se pojavljuju u procesu razvoja softvera. Na sledećoj slici prikazani su najvažniji elemenati razvoja softvera od kojih zavisi uspešnost svakog softverskog poduhvata.



Slika 1. Elementi metodologije razvoja softvera

Gornja slika može biti interpretirana na sledeći način.

Softverski timovi (Teams) koji se sastoje od menadžera, dizajnera, testera, dokumentalista i sl. (Roles) koji poseduju potrebna znanja i veštine (Skills), svojim svakodnevnim aktivnostima (Activities) kao što su planiranje i programiranje kroz procese (Process) i ključne rezultate (Milestones), a korišćenjem različitih tehnika (Techniques) i alata (Tools) stvaraju nove softverske proizvode (Products) određenog kvaliteta (Quality) i po „de fakto“ ili „de jure“ standardima (Standards). Naravno, ne treba zaboraviti da su timovi sastavljeni od ljudi koji imaju svoje osobenosti (Personality) o kojima takođe treba voditi računa.

Kada se proces razvoja (proizvodnje) softvera shvati na prikazani način, onda postaje jasnije zašto je potrebno da se taj razvoj odvija prema nekoj unapred usvojenoj metodologiji. Odsustvo metodologije bi za slučaj svakog malo složenijeg softverskog poduhvata vodilo ka haosu sa negativnim posledicama.

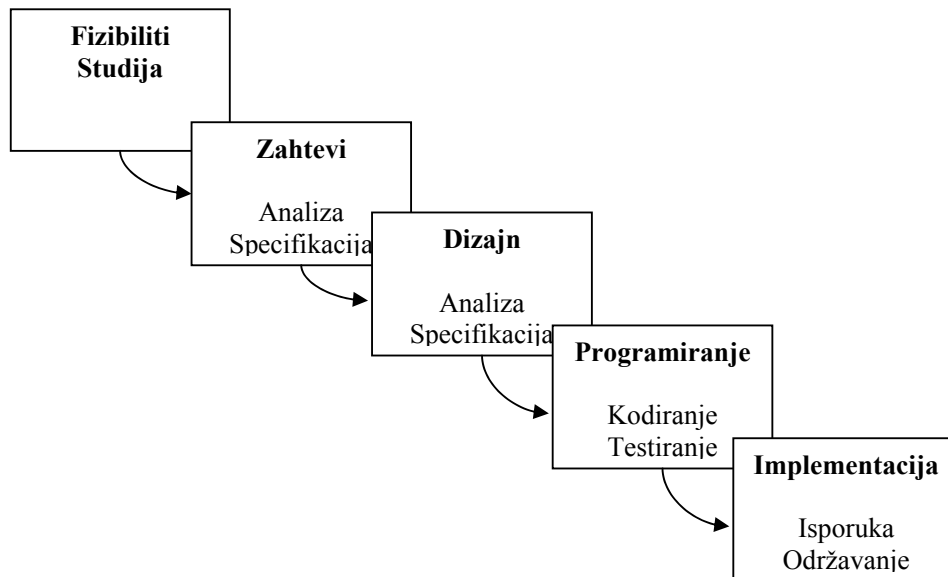
Životni ciklus razvoja softvera

Tokom vremena razvijeno je više modela za razvoj softverskih sistema poznatim pod nazivom SDLC (System Development Life Cycle) modeli.

SDLC modeli imaju za cilj da daju jedan metodičan pristup za razvoj informacionih sistema i to za sve faze razvoja od izrade studije izvodljivosti (feasibility study) , sve do održavanja gotovih aplikacija.

Ukratko ćemo navesti neke od najpoznatijih SDLC modela za razvoj softvera.

Vodopad (Waterfall) model: To je klasičan SDLC model kod kojeg se faze u razvoju softvera linearno i sekvencijalno odvijaju jedna za drugom bez preklapanja faza i/ili vraćanja (iteracije) na prethodno završenu fazu. To se može islustrovati sledećom slikom:



Slika 2 Vodopad model

Fizibiliti

Fizibiliti stufijom se određuje da li вреди uopšte započeti rad na nekom softverskom projektu. Ako je odluka da se projekat realizuje onda se već u samoj fizibiliti studiji daju osnove plana izvođenja projekta i procena budžeta (i drugih resursa) potrebnih za realizaciju.

Zahtevi

Zahtevi za izradu novog ili modifikaciju postojećeg sistema predstavljaju detaljan opis željene funkcionalnosti i drugih karakteristika softverskog sistema koji će biti razvijen. Zahtevi moraju biti što precizniji i u pisanoj formi, kako bi se na kraju projekta moglo ustanoviti da li realizovani sistem odgovara zahtevima.

Dizajn

Dizajn se može podeliti na tri dela:
Design focuses on:

- Viši nivo kojim se određuje koji su sve programski moduli potrebni, šta su njihovi ulazi/izlazi i kako oni interaguju međusobno i sa drugim softverom i operativnim sistemom.
- Niži nivo na kojem se definiše način rada programskih modula, koji algoritmi i modeli će biti korišćeni, koje su programske biblioteke potrebne i sl.
- Dizajn podataka kao što je interfejs za ulaz/izlaz podataka, strukture podataka koje će se koristiti i sl.

Do kojih će se detalja ići pri dizajnu je stvar izbora. Ako iamo veoma detaljan dizajn, pisanje programskog koda će biti mnogo lakše ali će naknadne promene ići mnogo teže, dok je za slučaj grubog (manje detaljnog) dizajna mnogo više posla ostavljeno za pisanje koda, ali je i jednostavnija njegova naknadna promena. Inad svega je važno da dizajn bude dobro dokumentovan i da u dokumentaciji jasno piše zašto su napravljene one odluke kod kojih je bilo više opcija. Takav pristup olakšava uključivanje novih programera na projekat, a takođe olakšava dalji razvoj sistema dodavanjem novih funkcija i karakteristika.

Programiranje i testiranje

U ovoj fazi se dizajn pretvara u programski kod. Programski alati kao što su kompajleri i dibageri se pri tome koriste za generisanje izvornog koda dobrog kvaliteta a time i celokupne softverske aplikacije. Testiranje manjih delova (modula) je pogodan način za kontrolu kvaliteta i pronalaženje grešaka što je ranije moguće. Testiranje sistema kao celine vrši se da se proveru da li sistem radi na ciljnim platformama, i da li njegovo ponašanje odgovara zahevima koji su postavljeni na početku projekta.

Održavanje

Od trenutka kada je softverski sistem isporučen korisnicima počinje i potreba za njegovim održavanjem. Mogu se pojaviti greške prouzrokovane pogrešno unetim podacima od strane korisnika (takve podatke uvrstiti u plan testiranja), ili zbog neočekivanog i/ili nepravilnog korišćenja softvera (takve slučajeve uvrstiti u dokumentaciju). korisnici, takođe, mogu zahtevati i dodatne funkcije koje nisu uključene u tekuću reviziju softvera, mogu tražiti da softver radi brže, ili čak postaviti pred razvojni tim i veće probleme. Proces rzvoja softvera mora biti prilagođen za promene koje takđe moraju proći kroz sve gore navedene faze.

Iterativni Razvoj (Iterative Development)

Ovde se propisuje inicijalna konstrukcija malog (pa sve većeg) dela softverskog projekta kojom se pomaže da svi oni koji su uključeni u razvoj otkriju probleme i pitanja pre nego što oni postanu suviše ozbiljni. Iterativni procesi su pogodni za komercijalni pristup razvoju jer omogućavaju da se zadovolje potrebe budućih korskika softvera koji ne znaju da definišu šta im je potrebno. Iterativne metode

uključuju i druge ideje kao što su agilni softver I ekstremno programiranje, o čemu će biti reči kasnije.

Brzi razvoj aplikacija (Rapid Application Development - RAD): Ovaj metod baziran je na konceptu da se softverski proizvod može dobiti brže i bolje organizovanjem radionica (workshops) interesnih grupa i na taj način generišu softverski zahtevi.

Zajednički razvoj (Joint application development - JAD): Ovaj model podrazumeva uključivanje korisnika u dizajn I razvoj aplikacije kroz seriju kolaborativnih vorkšopova koji se nayivaju JAD sesijama.

Prototipski razvoj (Prototyping Model): U ovom metodu se najpre uradi prototip sistema koji predstavlja samo jednu ranu aproksimaciju finalnog sistema. Zatim se prototip testira I unapređuje sve do nivoa konačno prihvatljivog prototipa na osnovu kojeg se onda razvija željeni softverski proizvod.

Sinhronizuj i stabilizuj (Synchronize-and-Stabilize): Ovaj model se zasniva na timovima koji rade paralelno na individualnim modulima zajedničke aplikacije, povremeno (često) sinhronizuju svoj kod sa kodovima ostalih timova do postizanja stabilnog sitema, a zatim nastavljaju dalji razvoj. Ovaj metod se generalno koristi u Majkrosoftvu i predstavlja osnovu Majkrosoftvove metodologije poznate kao "Microsoft Solution Framework" (MSF).

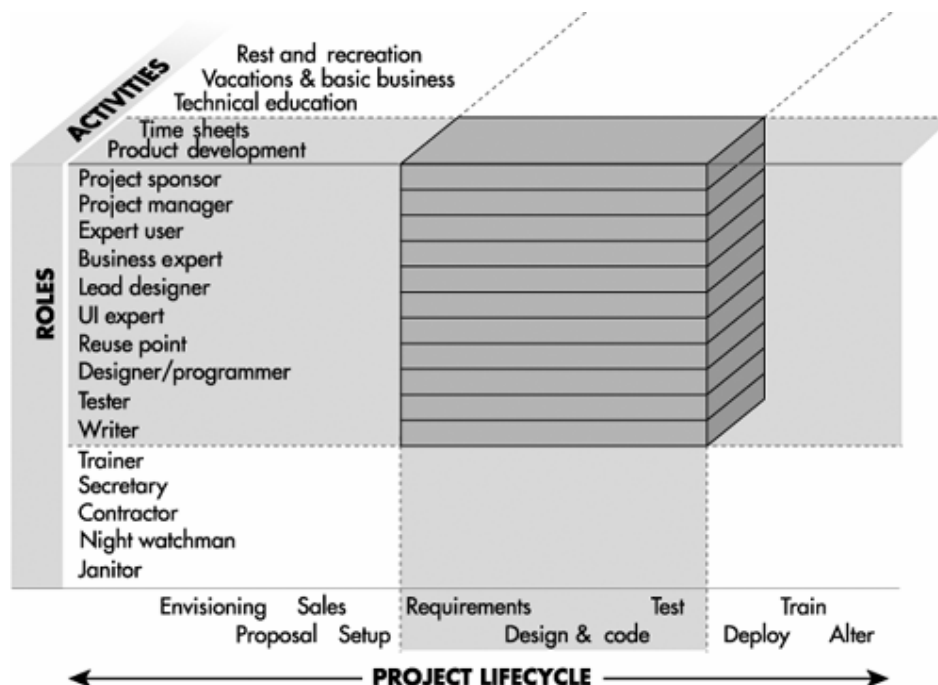
Spiralni razvoj (Spiral Model): Ovaj model razvoja softvera kombinuje vodopad i prototip modele. On je pogodan za velike, skupe i komplikovane projekte.

Organizija rada na softverskih projektima

Razvoj i proizvodnja softvera zahteva visko obrazovane stručnjake raznih profile, najsavremenija sredstva rada kao i dobru organizaciju rada. Analiziraćemo aktivnosti i uloge članova softverskog tima, organizaciju timova i potrebne uslove rada.

Aktivnosti i uloge softverskih timova

Sledeća slika pokazuje kako se aktivnosti i uloge softverskih timova dovode u vezu sa životnim ciklusom razvoja softvera. Liste aktivnosti i uloga su samo ilustrativne i naravno ne iscrpljuju sve slučajeve.



Slika . Tri dimenzije projekta. Metodologije se bave samo delom projektnih aktivnosti.

Opisaćemo sada neke od standardnih aktivnosti i uloga prikazanih na gornjoj slici.

Aktivnosti

Razvoj proizvoda (Product Development): Ovo je centralna aktivnost u softverskim projektima. Obuhvata već pomenute procese analize, dizajna, kodiranja, testiranja, održavanja, pisanja tehničke dokumentacije i sl.

Tehnička edukacija (Technical Education): Informacione tehnologije se veoma brzo razvijaju. Praćenje tako brzog razvoja zahteva poseban tretman u obuci i treningu članova softverskog tima. Specijalistički kursevi, seminari, konferencije, vorkšopovi su samo neki od oblika permanentnog obrazovanja neophodnog da softverski stručnjaci održe visok nivo svoje profesionalnosti.

Odmor i rekreacija (Rest and Recreation) Rad na softveru zahteva izuzetno naporno mentalno angažovanje. Borba sa vremenom za što raniji izlazak na tržište sa proizvodom koji se razvija može da bude veoma stresna. Zato softverske kuće pokušavaju da ove izuzetne napore svojih zaposlenih kompenziraju raznim oblicima komfornih radnih okruženja, sportskih i rekreacionih aktivnosti. Nije čudo što je u Kaliforniji sedište najvećih softverskih kompanija (Google, na primer) jer je Kalifornija poznata kao deo USA sa izuzetnom ponudom za rekreaciju i razonodu.

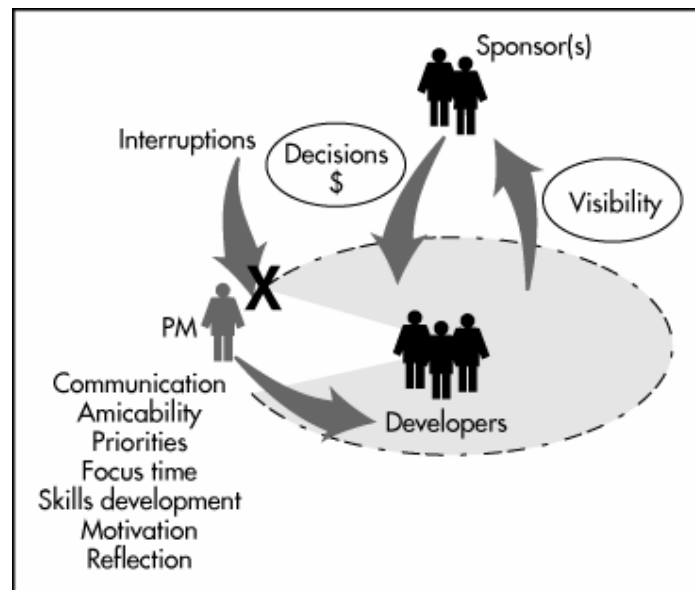
Uloge

Dizajneri/Programeri: Ovo je suštinska uloga svakog razvoja softvera. To su kreatori sistema. Oni definišu arhitekturu sistema, vrše izbor platformi, alata, metoda, algoritama, proizvode kod (source code). Može postojati više nivoa prema iskustvu (seniori, juniori, početnici, itd.) ili prema oblastima (GUI, biznis logika, baze podataka, itd).

Testeri: Testeri su veoma važna uloga kojom se postiže više efekata. Pored kontrole kvaliteta softvera koji se razvija, testeri mogu da pomognu i u podizanju ukupne produktivnosti tima, da ukažu ne samo na reške u funkcionisanju već i da, svojim primedbama, pomognu dizajn korisničkog interfejsa, i/ili ukupne funkcionalnosti sistema.

Pisci: Pisci su oni koji proizvode hiljade stranica teksta kojim se opisuje proizvod., od prvog dokumenta kojim se proizvod najavljuje (“white paper”), pa sve do opisa sistema, uputstava za korišćenje, marketinških prezentacija i reklamiranja.

Menadžer projekta (Project Manager): To je osoba koja je najviše odgovorna za uspeh projekta. On planira aktivnosti, prati izvršenje plana, komunicira sa svim zainteresovanim stranama u projektu (članovima softverskog tima, menadžmentu firme, kornicima, itd.). Sldeća slika ilustruje rad menadžera projekta.

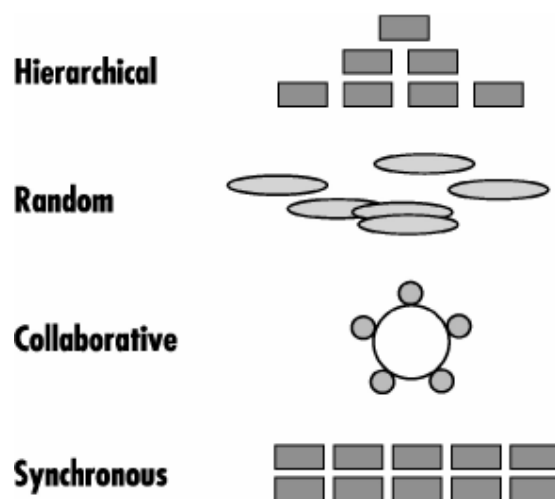


Slika Uloga menadžera projekta u agilnim SW projektima

Sponzor projekta (Project Sponsor) To je osoba koja predstavlja “spiritus movens” projekta. Projekat je “živ” sve dok su članovima tima na raspolaganju resursi (novac, ljudi i oprema) potrebni za njegovu realizaciju. Sponzor predstavlja sponu projektnog tima i finansijera projekta.

Timovi

Softverski timovi mogu biti organizovani na više načina. Sledeća slika prikazuje tipične organizacije.



Slika Četiri principa organizacije

Kod hijerhijske organizacije uočava se jasna struktura tima sa “šefom” na vrhu i jednim ili više podređenih nivoa. (Primer: Vojna jedinica)

Random (razasuta) organizacija je organizacija u kojoj “svako radi svoj posao” bez subordinacije i definisane kolaboracije. (Primer: Fudbalska reprezentacija Srbije)

Kolaborativna organizacija zahteva povećanu komunikaciju između članova tima, nema hijerarhiju, ali deluje povezano. (Primer: Vaterpolo reprezentacija Srbije)

Sinhrona organizacija pokušava da obezbedi sinhronizovano delovanje svojih delova. Nema izraženu hijerarhiju. (Primer: Sinhrono plivanje).

U softverskih firmama prisutne su sva četiri tipa organizacije, jer imaju međusobne prednosti i nedostatke. Bez obzira o kojoj se vrsti organizacije radi za timove važi pravilo da je potrebno vreme za njihovo uspostavljanje. Proces “uigravanja” se odvija kroz više faza, od formiranja tima (forming), kroz burnu fazu upoznavanja (storming), do uspostavljanja normalnog režima funkcionisanja (norming) i skladnog izvršavanja projektnih zadataka (performing).

Jedan primer moderne organizacije je takozvano “programiranje u parovima” (pair programming), gde dva programera rade simultano, kako prikazuje sledeća slika.



Slika. Programiranje u parovima

Na kraju, pominjemo i najnoviju inicijativu za metodologiju izrade softvera poznatu kao “Agilno programiranje” (Agile initiative). Ova metodologija se zasniva na principima datim u manifestu koji proklamuje ovu ideju (Agile Manifesto).

Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it.

Through this work we have come to value:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Pitanja

1. Navedite osnovne elemente razvoja softvera za koje je potrebna metodologija?
2. Zašto je važno korišćenje metodologija?
3. Šta je životni ciklus razvoja softvera (SDLC)?
4. Koje su glavne faze SDLC-a? Objasnite svaku fazu.
5. Navedite nekoliko poznatih modela za razvoj softvera?
6. Koje su četiri poznata načina organizacije?
7. Navedite neke aktivnosti pri razvoju softvera.
8. Opišite ulogu dizajnera-programera.
9. Opišite ulogu SW testera?
10. Šta su glavne odgovornosti softverskog menadžera projekta?
11. Kakva je uloga sponzora SW projekta?
12. Navedite nekoliko uloga koje mogu imati članovi softverskog tima.
13. Šta je agilni softver?
14. Koje su prednosti (nedostaci) programiranja u parovima?