UNIVERZITET METROPOLITAN BEOGRAD

SE101 - RAZVOJ SOFTVERA I INŽENJERA SOFTVERA

Agilno softversko inženjerstvo proizvoda

Lekcija 05

PRIRUČNIK ZA STUDENTE

SE101 - RAZVOJ SOFTVERA I INŽENJERA SOFTVERA

Lekcija 05

AGILNO SOFTVERSKO INŽENJERSTVO PROIZVODA

UVOD

Sadržaj lekcije

Brzo iznošenje softverskog proizvoda na tržište je od ključne važnosti. Ovo važi za sve vrste proizvoda — od jednostavnih mobilnih aplikacija do velikih poslovnih proizvoda. Ako je proizvod objavljen kasnije nego što je planirano, konkurent je možda već zauzeo tržište ili ste možda propustili tržišni prozor, kao što je početak sezone praznika. Kada se korisnici posvete proizvodu, obično nerado menjaju, čak i ne prelaze na tehnički superiorniji proizvod.

Agilno softversko inženjerstvo *se fokusira na brzu isporuku funkcionalnosti, reagovanje na promene specifikacija proizvoda i minimiziranje troškova razvoja. „Režijski troškovi“ su svaka aktivnost koja ne doprinosi direktno brzoj isporuci proizvoda. Brz razvoj i isporuka i fleksibilnost brzih promena su osnovni zahtevi za razvoj proizvoda.*
Razvijen je veliki broj „agilnih metoda“. Svaki ima svoje pristalice, koji su često evangelistički o prednostima metode. U praksi, kompanije i pojedinačni razvojni timovi biraju i biraju agilne tehnike koje su pogodne za nihov način rada i koje su najprikladnije njihovoj veličini i vrsti proizvoda koji razvijaju. Ne postoji najbolja agilna metoda ili tehnika. Zavisi od toga ko koristi tehniku, od razvojnog tima i vrste proizvoda koji se razvija.

Poglavlje 1

Agilni metodi

NEDOSTACI PLANSKI VOĐENOG RAZVOJA SOFTVERA

Nemoguće brzo isporučiti softver i brzo odgovoriti na zahteve za izmene isporučenog softvera u slučaju razvoja vođenim planom.

Tokom 1980-ih i ranih 1990-ih, bilo je široko rasprostranjeno gledište da je najbolji način za stvaranje dobrog softvera korišćenje kontrolisanih i rigoroznih procesa razvoja softvera. Procesi su uključivali detaljno planiranje projekta, specifikaciju i analizu zahteva, korišćenje metoda analize i projektovanja podržanih softverskim alatima i formalno osiguranje kvaliteta. Ovaj stav je došao iz zajednice softverskog inženjerstva koja je bila odgovorna za razvoj velikih, dugovečnih softverskih sistema kao što su vazduhoplovni i državni sistemi. To su bili „jednokratni“ sistemi, tj. softverski sistemi razvijeni u skladu sa zahtevima određenog kupca.

Ovaj pristup se ponekad nazivarazvojem vođenim planom. Evoluirao je da podrži softversko inženjerstvo gde su veliki timovi razvijali složene sisteme sa dugim životom. Timovi su često bili geografski raspoređeni i dugo su radili na softveru. Primer ovog tipa softvera je sistem upravljanja modernim avionom. Razvoj avionskog sistema može trajati pet do deset godina od početne specifikacije do postavljanja u avion.

Planski vođen razvoj uključuje značajne troškove planiranja, projektovanje i dokumentovanje sistema. Ovi troškovi su opravdani za kritične sisteme gde rad nekoliko razvojnih timova mora biti koordinisan i različiti ljudi mogu održavati i ažurirati softver tokom njegovog životnog veka. Detaljni dokumenti koji opisuju softverske zahteve i projektno rešenje su važni kada je neformalna komunikacija tima nemoguća.

Međutim, ako se planski vođen razvoj koristi za male i srednje softverske proizvode, troškovi su toliko veliki da dominiraju procesom razvoja softvera. Previše vremena se troši na pisanje dokumenata koji možda nikada neće biti pročitani, a ne na pisanje koda. Sistem je detaljno preciziran pre početka implementacije. Greške u specifikacijama, propusti i nesporazumi se često otkrivaju tek nakon što je značajan deo sistema implementiran.
Da bi rešili ove probleme, programeri moraju da ponove posao za koji su mislili da je završen. Kao posledica toga, praktično je nemoguće brzo isporučiti softver i brzo odgovoriti na zahteve za izmene isporučenog softvera.

AGILNI MANIFEST

Filozofija koja stoji iza agilnih metoda ogleda se u manifestu agilnosti oko kojeg su se složili vodeći programeri ovih metoda

Nezadovoljstvo razvojem softvera vođenim planom dovelo je do stvaranja agilnih metoda 1990-ih. Ove metode su omogućile razvojnom timu da se fokusira na sam softver, a ne na njegov dizajn i dokumentaciju. Agilne metode brzo isporučuju softver klijentima koji radi, a oni onda mogu da predlože nove ili drugačije zahteve za uključivanje u kasnije verzije sistema. Primena agilnih metoda razvoja softvera smanjiuje birokratiju softverskog procesa izbegavanjem poslova koji imaju sumnjivu dugoročnu vrednost i eliminisanjem dokumentacije koja verovatno nikada neće biti korišćena.
Filozofija koja stoji iza agilnih metoda ogleda se u manifestu agilnosti oko kojeg su se složili vodeći programeri ovih metoda. Tabela 1 prikazuje ključnu poruku u agilnom manifestu.

Tabela 1: Agilni manifesto

Otkrivamo bolje načine za razvoj softvera radeći to i pomažući drugima da to urade. Kroz ovaj rad, došli smo do vrednosti:

- **pojedinci i interakcije su važniji** od procesa i alata;
- **radni softver je važniji** od sveobuhvatne dokumentacije;
- **saradnja sa klijentima je važnija** od pregovaranja o ugovoru;
- **reagovanje na promenu je važnije** od plana.

Iako postoji vrednost na stavkama sa desne strane, mi ipak više cenimo stavke sa leve strane.

Slika 1.1 Agilni manifesto (skraćeno) [Sommerville 2021, Tabela 2.1]

Svi agilni metodi su zasnovane na inkrementalnom razvoju i isporuci. Najbolji način da se razume inkrementalni razvoj je razmišljanje o softverskom proizvodu kao o skupu svojstava. Svaka funkcija čini nešto za korisnika softvera. Možda postoji funkcija koja omogućava unos podataka, funkcija za pretragu unetih podataka i funkcija za formatiranje i prikaz podataka. Svaki korak softvera treba da implementira mali broj karakteristika proizvoda.

AKTIVNOSTI RAZVOJA INKREMENTA

Sa inkrementalnim razvojem, odlažete odluke dok ih zaista ne budete morali da donesete

Sa inkrementalnim razvojem, odlažete odluke dok ih zaista ne budete morali da donesete. Počinjete određivanjem prioriteta funkcija tako da se najvažnije funkcije prvo implementiraju. Ne brinete o detaljima svih funkcija – definišete samo detalje funkcije koju planirate da uključite u inkrement. Ta funkcija se zatim implementira i isporučuje. Korisnici mogu da ga isprobaju i da daju povratne informacije razvojnom timu. Posle toga, nastavljate da definišete i implementirate sledeću funkciju sistema.
Ovaj proces prikazan na slici 1, a inkrementalne razvojne aktivnosti su opisuane u tabeli 2.

Slika 1.2 Razvoj inkrementa [Sommerville 2021, Fig. 2.1]

Tabela T2: Aktivnosti razvoja inkremenata

Aktivnost	Opis
Izaberi svojstva u inkrementu	Upotrebom liste svojstava planiranog proizvoda, izaberi ona svojstva koja se mogu primeniti u sledećem inkrementu proizvoda.
Poboljšaj opise svojstva	Dodaj detalje u opise svojstva tako da članovi tima imaju zajedničko razumevanje svakog svojstva i da postoje dovoljno detalja da moće da se počne sa primenom (tj. programiranjem).
Primeni i testiraj svojstvo	Primeni svojstvo i razvij automatske testove za to svojstvo tako da pokažu da je njegovo ponašanje usaglašeno sa svojim opisom.
Integriši svojstvo u sistem i testiraj sistem	Integriši razvojeno svojstvo u postojeći sistem i testiraj ga radi provere da li radi usaglašeno sa drugim svojstvima.
Isporuči inkrement sistema	Isporuči inkrement sistema kupcu ili menadžeru proizvoda radi provere i komentara. Ako je dovoljno svojstava ovako primenjeno, objavi novu verziju sistema da bi je kupci počeli da koriste.

Slika 1.3 Aktivnosti rayvoja inkrementa [Sommerville 2021, Table 2.1]

PRINCIPI AGILNOG RAZVOJA

Sve agilni metodi dele skup principa zasnovanih na manifestu agilnosti, tako da imaju mnogo zajedničkog.

Stvarnost ne odgovara uvek ovom jednostavnom modelu razvoj novih svojstava softverskog proizvoda. . Ponekad se povećanje mora posvetiti razvoju infrastrukturne usluge, kao što je usluga baze podataka, koju koristi nekoliko funkcija; ponekad je potrebno da isplanirate korisnički interfejs tako da dobijete konzistentan interfejs za sve funkcije; a ponekad je potreban novi inkrement da bi se rešili neki problemi, kao što su problemi sa performansama,

koji su otkriveni tokom testiranja sistema.
Sve agilni metodi dele skup principa zasnovanih na manifestu agilnosti, tako da imaju mnogo zajedničkog. U tabeli 3. Su sumarno prikazani ovi agilni principi.

Skoro svi softverski proizvodi se sada razvijaju agilnim pristupom. Agilni metodi rade za inženjerstvo proizvoda jer su softverski proizvodi obično samostalni sistemi, a ne sistemi sastavljeni od nezavisnih podsistema. Razvijaju ih zajednički timovi koji mogu neformalno komunicirati. Menadžer proizvoda može lako da komunicira sa razvojnim timom. Shodno tome, nema potrebe za formalnim dokumentima, sastancima i međutimska komunikacija.

Tabela T3: Principi agilnog razvoja

Princip	Opis
Uključenje kupca	Uključiti kupce za blisku saradnju sa razvojnim timom softvera. Njihova uloga je da obezbede zahteve sistema i da im odrede prioritet, kao i da ocene svaki inkrement.
Podrška promenama	Očekuju se promene svojstava proizvoda i njihovih detalja, kao i promene u razvojnom timu, a i zato što menadžer proizvoda uči i saznaje više o proizvodu. Softver treba prilagođavati u saglasnoti sa ovim promenama.
Inkrementalni razvoj i isporuka	Uvek se softver razvija u inkrementima. Testira se i ocenjuje svaki inkrement po njegovom razvoju, a razvojni tim se obaveštava u zahtevanim promenama.
Održavanje jednostavnosti	Fokus je ka jednostavnošću, kako kod softverskog proizvoda, tako i u slučaju procesa razvoja. Uvek kada je to moguće, treba eliminisati složenost iz sistema.
Fokus je ka ljudima, a ne u procesu razvoja	Verujte razvojnom timu i ne očekujte da svako uvek radi stvari na isti način. Članove tima treba pustiti da razviiju svoj sopstveni način rada, bez ograničavanja definisanjem softverskih procesa.

Slika 1.4 Principi agilnog razvoj [Sommerville 2021, Table 2.2]

AN OVERVIEW OF AGILE DEVELOPMENT (VIDEO)

Pregled aginog razvoja

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Poglavlje 2

Ekstremno programiranje

PRAKSE EKSTREMNOG PROGRAMIRANJA

10 osnovnih praksi su predložili programeri ekstremnog programiranja, a koje karakterišu XP.

Ideje koje leže u osnovi agilnih metoda razvili su brojni različiti ljudi tokom 1990-ih. Međutim, najuticajniji rad koji je promenio kulturu razvoja softvera bio je razvoj ekstremnog programiranja (Extreme Programming – XP). Naziv je skovao Kent Bek 1998. jer je u pristup gurnuo priznatu dobru praksu, kao što je iterativni razvoj, do „ekstremnih“ nivoa. Na primer, redovna integracija, u kojoj rade svi programeri u timu rade na integrisanju softvera i testiranju, dobra je praksa softverskog inženjeringrstva. XP zagovara da promenjeni softver treba integrisati nekoliko puta dnevno, čim se promene testiraju.
XP se fokusirao na nove tehnike razvoja koje su bile usmerene na brz, postepeni razvoj softvera, promene i isporuku. Slika 1 prikazuje 10 osnovnih praksi, koje su predložili programeri ekstremnog programiranja, a koje karakterišu XP.

Slika 2.1 Prakse ekstremnog programiranja [Sommerville 2021, Fig. 2.2]

RASPOSTRANJENA PRAKSA XP PROGRAMIRANJA

Razvojni timovi biraju tehnike koje smatraju korisnima s obzirom na njihovu organizacionu kulturu i tip softvera koji pišu.

Programeri XP-a tvrde da je to holistički pristup. Sve ove prakse su neophodne. U stvarnosti, međutim, razvojni timovi biraju tehnike koje smatraju korisnima s obzirom na njihovu organizacionu kulturu i tip softvera koji pišu. Tabela 4 opisuje X prakse koje su postale deo glavnog softverskog inženjerstva, posebno za softver razvoj proizvoda. Ostale XP prakse prikazane na slici 2 su manje široko prihvaćene, ali se koriste u nekim kompanijama.

U okviru predmeta SE330 Agilne metode razvoja softvera, koji je planiran u trećoj godini vaših studija, biće razmatrane i ove druge agilne metode i tehnike.

Tabela T4: Široko raspostranjena XP praksa

Praksa	Opis
Inkrementalno planiranje/ korisničke priče	Ne postoji „veliki plan“ za sistem. Umesto toga, šta treba da se primeni (zahtevi) u svakom koraku se utvrđuje u razgovorima sa predstavnikom kupca. Zahtevi su napisani kao korisničke priče. Priče koje će biti uključene u izdanje određene su raspoloživim vremenom i njihovim relativnim prioritetom.
Mala izdanja	Najpre se razvija minimalni korisni skup funkcionalnosti koji pruža poslovnu vrednost. Izdanja sistema su česta i postepeno dodaju funkcionalnost prethodnom izdanju.
Razvoj vođen testom	Umesto da pišu kod, a zatim testove za taj kod, programeri prvo pišu testove. Ovo pomaže da se razjasni šta kod zapravo treba da radi i da je uvek dostupna „testirana“ verzija koda. Automatizovani okvir za testiranje jedinica se koristi za pokretanje testova nakon svake promene. Novi kod ne bi trebalo da „razbije“ kod koji je već bio urađen.
Kontinuirano integracija	Čim se završi rad na zadatku, on se integriše u ceo sistem i kreira se nova verzija sistema. Svi jedinični testovi svih programera se pokreću automatski i moraju biti uspešni pre nego što se prihvati nova verzija sistema.
Refaktoring	Refaktorisanje znači poboljšanje strukture, čitljivosti, efikasnosti i bezbednosti programa. Od svih programera se očekuje da refaktorišu kod čim se pronađu potencijalna poboljšanja koda. Ovo čini kod jednostavnim i održavanim

Slika 2.2 Široko raspostranjena praksa XP programiranja [Sommerville 2021, Table 2.4]

PROGRAMIRANJE U PARU

U programiranju u paru dva programera kreiraju svaku jedinicu koda.

Možda ćete biti iznenađeni što „Jednostavano projektno rešenje“ nije na listi popularnih XP praksi. Programeri XP-a su predložili da se pri projektovanju softvera primenjuje princip „YAGNI“ (“You Ain’t Gonna Need It”, tj. “Neće vam trebati»). To znači da bi trebalo bi da uključite samo funkcionalnost koja se zahteva i ne bi trebalo da dodajete dodatni kod da

biste se nosili sa situacijama koje su programeri predvideli. Ovo zvuči kao odlična ideja. Nažalost, ovaj pristup zanemaruje činjenicu da kupci retko razumeju probleme u okviru svog sistema, kao što su bezbednost i pouzdanost. Morate da projektujete i implementirate softver da biste uzeli u obzir ove probleme. To obično znači uključivanje koda za rešavanje situacija koje kupci verovatno neće predvideti i opisati u korisničkim pričama.

Prakse kao što je kupac na licu mesta i kolektivno vlasništvo nad kodom su dobre ideje. Klijent na licu mesta radi sa timom, predlaže priče i testove i uči o proizvodu. Međutim, *realnost je da kupci i zamjenski kupci, kao što su menadžeri proizvoda*, imaju mnogo drugih stvari da rade. Teško im je da nađu vremena da se u potpunosti uključe u razvojni tim.

Kolektivno vlasništvo obeshrabruje individualno vlasništvo nad kodom, ali se pokazalo nepraktičnim u mnogim kompanijama. Za neke vrste koda su potrebni stručnjaci. Neki ljudi mogu raditi sa skraćenim radnim vremenom na projektu i tako ne mogu učestvovati u njegovom „vlasništvu“. Neki članovi tima možda nisu psihološki neprikladni za ovaj način rada i ne žele da „poseduju“ tuđi kod.
U programiranju u paru dva programera kreiraju svaku jedinicu koda. Predložili su ga pronalazači XP-a jer su verovali da par može učiti jedan od drugog i uhvatiti greške jedni drugih. Oni su sugerisali da su dvoje ljudi koji rade zajedno produktivniji od dvoje ljudi koji rade kao pojedinci. Međutim, nema čvrstih dokaza da je programiranje u paru produktivnije od individualnog rada. Mnogi menadžeri smatraju da je programiranje u paru neproduktivno jer se čini da dvoje ljudi rade jedan posao.

UPRAVLJANJE TIMOM PRI EKSTREMNOM PROGRAMIRANJU

Mora postojati eksplicitno upravljanje u kojem menadžer može uzeti u obzir poslovne potrebe i prioritete, kao i tehnička pitanja.

Rad održivim tempom, bez prekovremenog rada, u principu je privlačan. Članovi tima bi trebalo da budu produktivniji ako nisu umorni i pod stresom. Međutim, teško je ubediti menadžere da će ovaj održivi rad pomoći da se ispoštuju kratki rokovi isporuke.
Ekstremno programiranje smatra menadžment kolektivnom timskom aktivnošću; obično ne postoji određeni menadžer projekta koji je odgovoran za komunikaciju sa menadžmentom i planiranje rada tima. U stvari, razvoj softvera je poslovna aktivnost i stoga mora da se uklopi sa širim poslovnim pitanjima finansiranja, troškova, rasporeda, zapošljavanja i upravljanja osobljem i održavanja dobrih odnosa sa klijentima. To znači da se pitanja upravljanja ne mogu jednostavno prepustiti razvojnom timu. Zbog toga, mora postojati eksplicitno upravljanje u kojem menadžer može uzeti u obzir poslovne potrebe i prioritete, kao i tehnička pitanja.

EXTREME PROGRAMMING (XP) - GEORGIA TECH - SOFTWARE DEVELOPMENT PROCESS (VIDEO)

Ekstremno programiranje

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Poglavlje 3

Scrum

ŠTA JE SCRUM?

Na početku svakog razvojnog ciklusa donose se odluke o tome kojim svojstvima softvera treba dati prioritet, kako ih treba razvijati i šta svaki član tima treba da radi.

U svakom softverskom poslu, menadžeri moraju da znaju šta se dešava i da li će projekat razvoja softvera verovatno isporučiti softver na vreme i u okviru svog budžeta. Tradicionalno, ovo uključuje izradu plana projekta koji pokazuje skup prekretnica (šta će se postići), rezultate (šta će tim isporučiti) i rokove (kada će prekretnica biti dostignuta). „Veliki plan“ za projekat pokazuje sve od početka do kraja. Napredak se ocenjuje upoređivanjem tog plana sa onim što je postignuto.

Problem sa prethodnim planiranjem projekta je što ono uključuje izradu detaljne odluke o softveru mnogo pre početka implementacije. Neizbežno se stvari menjaju. Pojavljuju se novi zahtevi, članovi tima dolaze i odlaze, poslovni prioriteti se razvijaju itd. Gotovo od dana kada se formulišu, projektni planovi moraju da se promene. Ponekad to znači da se „gotovi“ posao mora ponoviti. Ovo je neefikasno i često odlaže konačnu isporuku softvera.

Na osnovu toga, programeri agilnih metoda su tvrdili da je upravljanje zasnovano na planu rasipno i nepotrebno. Bolje je planirati postepeno kako bi se plan mogao promeniti kao odgovor na promenljive okolnosti. Na početku svakog razvojnog ciklusa donose se odluke o tome kojim svojstvima softvera treba dati prioritet, kako ih treba razvijati i šta svaki član tima treba da radi. Planiranje treba da bude neformalno sa minimalnom dokumentacijom i bez određenog menadžera projekta.
Nažalost, ovaj neformalni pristup menadžmentu ne zadovoljava širu poslovnu potrebu praćenja i procene napretka. Viši menadžeri nemaju vremena da se uključe u detaljne razgovore sa članovima tima. Menadžeri žele nekoga ko može da izveštava o napretku i da svoje brige i prioritete vrati razvojnom timu. Oni moraju da znaju da li će softver biti spreman do planiranog datuma završetka, i potrebne su im informacije da ažuriraju svoj poslovni plan za proizvod.Ovaj zahtev za proaktivnijim pristupom agilnom upravljanju projektima doprineo je razvoju Scrum-a. *Za razliku od XP-a, Scrum nije zasnovan na tehničkim praksama.* Umesto toga, on je razvijen da obezbedi okvir za stabilnu projektnu organizaciju sa određenim pojedincima (ScrumMaster i vlasnik proizvoda) koji deluju kao interfejs između razvojnog tima i organizacije. Programeri Scrum-a su želeli da naglase da ovi pojedinci nisu bili „tradicionalni“ menadžeri projekata koji imaju ovlašćenja da usmeravaju tim. Tako su izmislili novu Scrum terminologiju za pojedince i za timske aktivnostic (Tabela 5, slika 1). Morate da znate ovaj Scrum žargon da biste razumeli Scrum metod.

TERMINOLOGIJA SCRUM-A

Dve ključne uloge u Scrum-u su vlssnik proizvoda i ScrumMaster

Tabela T5: Terminologija Scrum-a

Scrum termin	Objašnjenje
Proizvod	Softverski proizvod koji razvija Scrum tim.
Vlasnik proizvoda	Član tima koji je odgovoran za identifikaciju proizvoda karakteristike i atributi. Vlasnik proizvoda rdi pregled urađenog posla i pomaže pri testiranju proizvoda.
Zaostatak proizvoda	Lista obaveza kao što su greške, funkcije i poboljšanja proizvoda koje Scrum tim još nije završio.
Razvojni tim	Mali samoorganizovani tim od pet do osam ljudi koji su odgovorni za razvoj proizvoda.
Sprint	Kratak period, obično dve do četiri nedelje, kada se proizvod razvija kao inkremet
Scrum	Dnevni sastanak tima na kojem se analizira napredak i posao koji da se uradi tog dana, a radi dogovora šta treba dalje raditi
ScrumMaster	Timski trener koji vodi tim u efikasnom korišćenju Scrum-a.
Pootencijalni inkrement za isporuku	Rezultat sprinta koji je dovoljno visokog kvaliteta da bude Isporučen korisniku
Brzina	Procenjena koičine posla koje radni tim može da uradi u toku jednog sprinta.

Slika 3.1 Terminologija Scrum-a [Sommerville 2021, Table 2.5]

Dve ključne uloge u Scrum-u nisu deo drugih metoda:

1. Vlasnik proizvoda je odgovoran da osigura da se razvojni tim uvek fokusira na proizvod koji gradi, a ne na tehnički interesantan, ali manje relevantan posao. U razvoju proizvoda, menadžer proizvoda obično treba da preuzme ulogu vlasnika proizvoda.
2. ScrumMaster je Scrum ekspert čiji je posao da vodi tim u efikasnom korišćenju Scrum metode. Programeri Scrum-a naglašavaju da ScrumMaster nije konvencionalni menadžer projekta, već je trener tima. ScrumMaster ima autoritet unutar tima o tome kako se Scrum koristi. Međutim, u mnogim kompanijama koje koriste Scrum, Scrum-Master takođe ima neke odgovornosti za upravljanje projektima.

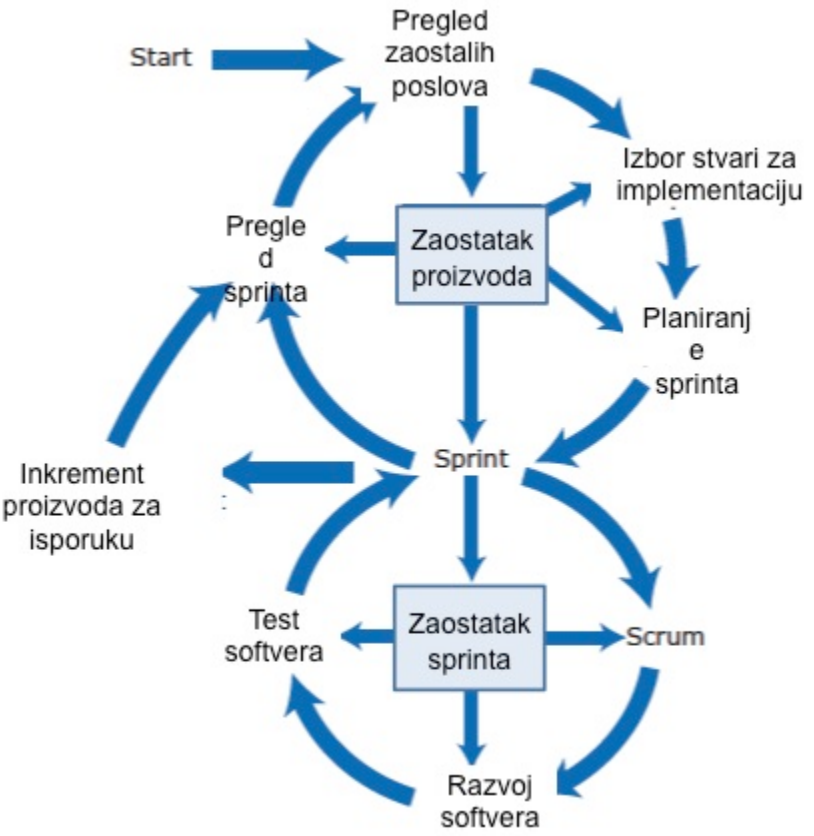
Drugi Scrum termin za koji bi možda trebalo objašnjenje je „potencijalni inkrement proizvoda koji se može isporučiti“. To znači da bi ishod svakog sprinta trebao biti kvalitetan kod poizvoda. Treba ga u potpunosti testirati, dokumentovati i, ako je potrebno, pregledati. Testove treba dostaviti sa kodom. Uvek treba da postoji visokokvalitetan sistem koji se može demonstrirati menadžmentu ili potencijalnim kupcima.

SPRINT CIKLUS

Softver treba da se razvija u nizu „sprintova“. Sprint je aktivnost fiksne dužine (vremenski okvir), pri čemu svaki sprint obično traje dve do četiri nedelje.

Scrum proces ili sprint ciklus je prikazan na slici 2. Osnovna ideja koja leži u osnovi Scrum procesa je da softver treba da se razvija u nizu „sprintova“. Sprint je aktivnost fiksne dužine (vremenski okvir), pri čemu svaki sprint obično traje dve do četiri nedelje. Tokom sprinta, tim ima dnevne sastanke (Scrums) kako bi pregledao dosadašnji rad i dogovorio se o aktivnostima tog dana. „Zaostatak sprinta“ (sprint backlog) se koristi za praćenje posla koji treba da se obavi tokom tog sprinta.

Planiranje sprinta se zasniva na zaostatku proizvoda (product backlog), koji predstavlja listu svih aktivnosti koje treba da se završe da bi se završio proizvod koji se razvija. Pre nego što počne novi sprint, pregleda se zaostali proizvod. Stavke najvišeg prioriteta se biraju za primenu u sledećem sprintu. Članovi tima rade zajedno na planiranju sprinta analizirajući odabrane stavke kako bi kreirali zaostatak sprinta. Ovo je lista aktivnosti koje treba obaviti tokom sprinta.
Tokom implementacije, tim implementira onoliko zaostalih stavki sprinta koliko može u fiksnom vremenskom periodu dozvoljenom za sprint. Nepotpune stavke se vraćaju u zaostatak proizvoda. Sprintovi se nikada ne produžavaju da bi se završio nekompletan predmet.



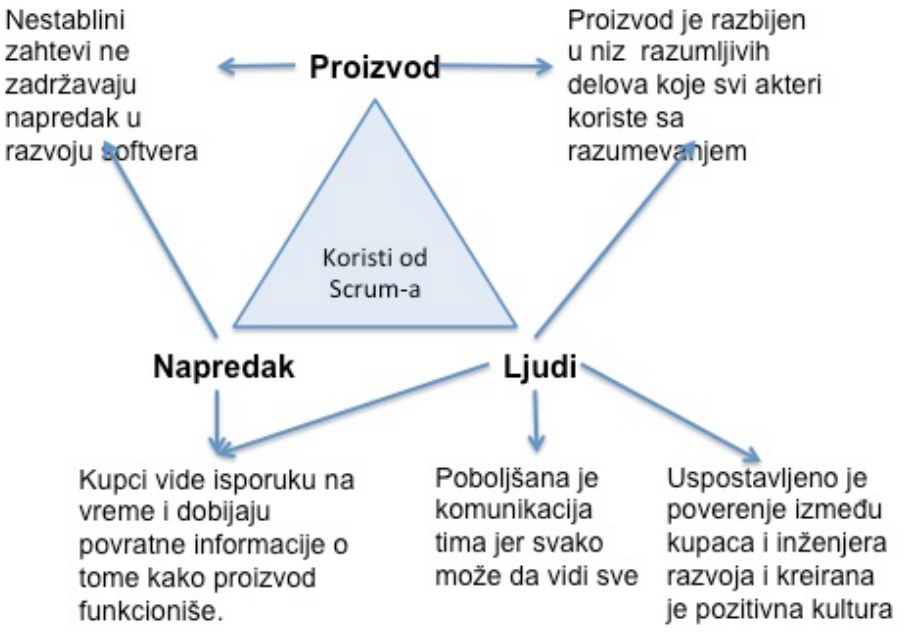
Slika 3.2 Ciklus Scrum-a [Sommerville 2021, Fig. 2.3]

KORISTI OD PRIMENE SCRUM-A

Ključne prednosti koje proizilaze iz upotrebe Scrum-a odnose se na proizvod koji se razvija, napredak projekta i uključene ljude

Sprint proizvodi ili inkrement proizvoda koji se može isporučiti kupcima ili interni proizvod. Interni rezultati, kao što su prototip proizvoda ili arhitektonsko projektno rešenje, pružaju informacije za buduće sprintove. Ako je rezultat sprinta deo konačnog proizvoda, trebalo bi da bude potpun. Osim ako tim ne mora da promeni funkcionalnost softvera, ne bi trebalo više da radi na tom povećanju softvera u budućim sprintovima.

Po završetku sprinta, održava sesastanak za pregled koji uključuje sve članove tima. Tim razgovara o tome šta je dobro prošlo tokom sprinta, koji su problemi nastali i kako su problemi rešeni. Članovi tima takođe razmišljaju o efikasnosti korišćenih alata i metoda.Cilj ovog sastanka je da timovi uče jedni od drugih kako bi izbegli probleme i poboljšali produktivnost u kasnijim sprintevima.
Ključne prednosti koje proizilaze iz upotrebe Scrum-a odnose se na proizvod koji se razvija, napredak projekta i uključene ljude (Slika 3).



Slika 3.3 Pet najznačajnijih prednosti od upotrebe Scrum-a [Sommerville 2021, Fig. 2.4]

MODIFIKACIJE SCRUM METODA

Scrum pruža okvir za „rad “ softverskog inženjerstva bez propisivanja inženjerskih tehnika koje treba koristiti.

Scrum je veoma uticajan u razvoju agilnog softverskog inženjeringstva. On pruža okvir za „rad “ softverskog inženjerstva bez propisivanja inženjerskih tehnika koje treba koristiti. Međutim,

Scrum je preskriptivan u definisanju uloga i Scrum procesa. U Scrum vodiču, „čuvari“ Scrum metode navode:
„Scrumove uloge, artefakti, događaji i pravila su nepromenljivi i iako je primena samo delova Scrum-a moguća, rezultat nije Scrum. Scrum postoji samo u celini i funkcioniše dobro kao kontejner za druge tehnike, metodologije i prakse.“

Oni veruju da ne bi trebalo da birate podskup Scrum praksi. Umesto toga, **trebalo bi da uzmete u obzir celu metodu**. Ipak, ova nefleksibilnost u suprotnosti sa osnovnim agilnim principom da pojedinci i interakcije treba da imaju prednost u odnosu na procese i alate. Ovaj princip sugeriše da pojedinci treba da budu u stanju da prilagode i modifikuju Scrum kako bi odgovarali svojim okolnostima.

U nekim okolnostima, ima smisla koristiti neke od ideja iz Scrum-a bez striktno pridržavanja metode ili definisanja uloga kako su tačno predviđene u Scrum-u. Generalno, „čisti Scrum“ sa svojim različitim ulogama ne mogu da koriste timovi sa manje od pet ljudi. Dakle, ako radite sa manjim razvojnim timom, morate da izmenite metod.

Mali timovi za razvoj softvera su norma u startapima, gde cela kompanija može biti razvojni tim. Oni su takođe uobičajeni u obrazovnim i istraživačkim okruženjima, gde timovi razvijaju softver kao deo svog učenja, i u većim proizvodnim kompanijama, gde je razvoj softvera deo šireg procesa razvoja proizvoda.
Motivisani timovi treba da donose sopstvene odluke o tome kako da koriste Scrum ili proces sličan Scrum-u. Međutim, preporučuje se da tri važna svojstva Scrum-a budu deo bilo kog procesa razvoja proizvoda: **zaostaci proizvoda**, **vremenski ograničeni sprintovi** i **samoorganizovani timovi**.

ZAOSTACI PROIZVODA

Zaostatak u razvoju proizvoda, ili kraće, zaostatak proizvoda (product backlog) je lista onoga što treba da se uradi da bi se završio razvoj proizvoda.

Zaostatak u razvju proizvoda, ili kraće, zaostatak proizvoda (product backlog) je lista onoga što treba da se uradi da bi se završio razvoj proizvoda. Stavke na ovoj listi nazivaju se stavke zaostalih proizvoda (Product Backlog Items- PBI). Zaostatak proizvoda može uključivati niz različitih stavki kao što su karakteristike proizvoda koje treba implementirati, korisnički zahtevi, osnovne razvojne aktivnosti i poželjna inženjerska poboljšanja. *Zaostatak proizvoda uvek treba da bude prioritet*, tako da stavke koje će se prvo primeniti budu na vrhu liste.

Stavke zaostalih proizvoda u početku su opisane u širem smislu bez mnogo detalja. Na primer, stavke prikazane u tabeli T6 (slika 4) mogu biti uključene u zaostatak proizvoda za verziju sistema iLearn, koji smo opisali u poglavlju 1. U poglavlju 3 objašnjeno je kako se karakteristike sistema mogu identifikovati iz vizije proizvoda. Oni tada postaju PBI. Takođe objašnjavam kako se korisničke priče mogu koristiti za identifikaciju PBI.

Tabela T6: Primeri stavki u zaostatku proizvoda (program backlog)

1. Kao nastavnik, želim da mogu da konfigurišem grupu alata koji su rasploživi za pojedine lekcije. (svojstvo)
2. Kao roditelj, želim da mogu da vidim rad mog deteta i dobijene ocene od strane nastavnika (svojstvo)
3. Kao nastavnik malih učenika, želim slikoviti koristnički ineterfejs sa oganičenim delom za čitanje.
4. Postaviti kriterijume za ocenjivanje softvera otvorenog koda koji se može da upotrebi kao osnov za razvoj delova sistema. (aktivnost razvoja).
5. Refaktorisati korisniči interfejs radi poboljšavanja njegove razumljivosti i performansi. (inženjerso poboljšanje)
6. Primeniti šifrirane personalne podatke korisnika. (inženjersko poboljšanje)

Slika 3.4 Primeri stavki u zaostatku proizvoda [Sommerville 2021, Table 2.6]

Tabela T6 prikazuje različite vrste zaostalih stavki proizvoda. Prve tri stavke su korisničke priče koje se odnose na karakteristike proizvoda koje treba implementirati. Četvrta stavka je timska aktivnost. Tim mora potrošiti vreme na odlučivanje kako da izabere softver otvorenog koda koji bi se mogao koristiti u kasnijim koracima. Ovu vrstu aktivnosti treba posebno uzeti u obzir kao PBI, a ne kao implicitnu aktivnost koja oduzima vreme članovima tima. Poslednje dve stavke se odnose na inženjerska poboljšanja softvera. Ovo ne dovodi do novih funkcija softvera.

STANJE STAVKI ZAOSTATKA PROIZVODA

Zaostatak proizvoda se stalno menja i proširuje tokom projekta kako se dodaju nove stavke i stavke u jednom stanju se analiziraju i pomeraju u prefinjenije stanje

PBI se mogu specificirati na visokom nivou i tim odlučuje kako da implementira ove stavke. Na primer, razvojni tim je u najboljoj poziciji da odluči kako da refaktoriše kod radi efikasnosti i razumljivosti. Nema smisla detaljnije precizirati ovu stavku na početku sprinta. Međutim, definicije karakteristika na visokom nivou obično zahtevaju preciziranje kako bi članovi tima imali jasnu predstavu o tome šta je potrebno i mogli da procene uključeni posao.
Smatra se da su stavke u zaostatku proizvoda u jednom od tri stanja, kao što je prikazano u tabeli T7 (slika 5). Zaostatak proizvoda se stalno menja i proširuje tokom projekta kako se dodaju nove stavke i stavke u jednom stanju se analiziraju i pomeraju u prefinjenije stanje.

Tabela T7: Stanja stavki zaostatka proizvoda (product backlog)

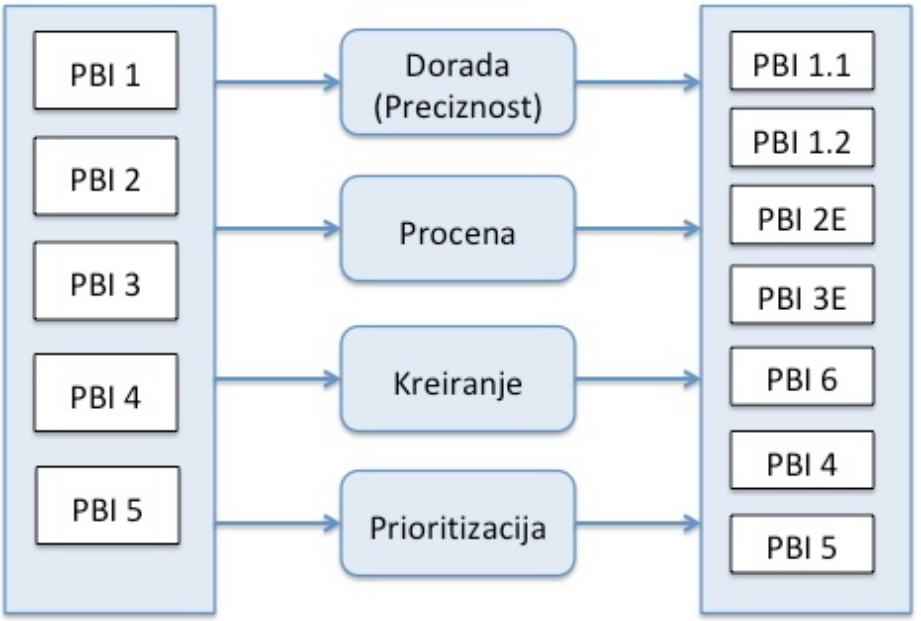
Status stavke	Opis
Spreman za razmatranje	Ovo su ideje na visokom nivou i opisi funkcija koji će biti uzeti u obzir za uključivanje u proizvod. Oni su okvirni tako da se mogu radikalno promeniti ili možda neće biti uključeni u konačni proizvod.
Spreman za razmatranje	Ovo su ideje na visokom nivou i opisi funkcija koji će biti uzeti u obzir za uključivanje u proizvod. Oni su okvirni tako da se mogu radikalno promeniti ili možda neće biti uključeni u konačni proizvod.
Spremni za doradu	Tim se složio da je ovo važna stavka koju treba implementirati kao deo trenutnog razvoja. Postoji razumno jasna definicija šta je potrebno. Međutim, potreban je rad na razumevanju i usavršavanju stavke. Spremni za implementaciju

Slika 3.5 Stanje stavki zapostatka proizvoda [Sommerville 2021, Table 2.7]

AKTIVNOSTI ZAOSTATKA PROIZVODA

Kritični deo Scrum agilnog procesa je pregled zaostatka proizvoda, koji bi trebalo da bude prva stavka u procesu planiranja sprinta.

Kritični deo Scrum agilnog procesa je pregled zaostatka proizvoda, koji bi trebalo da bude prva stavka u procesu planiranja sprinta. U ovom pregledu se analizira zaostatak proizvoda i daju se prioriteti i prečišćavaju zaostale stavke. Pregledi zaostatka se takođe mogu odvijati tokom sprinta dok tim sazna više o sistemu. Članovi tima mogu da modifikuju ili preciziraju postojeće zaostale stavke ili da dodaju nove stavke koje će biti implementirane u kasnijem sprintu. Tokom prego zaostatka proizvoda, stavke se mogu premeštati iz jednog stanja u drugo.
Slika 6 prikazuje četiri operacije koje mogu izmeniti zaostatak proizvoda. U ovom primeru, stavka 1 zaostalih predmeta je podeljena na dve stavke, stavke 2 i 3 su procenjene, stavke 4 i 5 su ponovo prioritetne, a stavka 6 je dodata. Obratite pažnju da nova stavka 6 ima veći prioritet od postojećih stavki 4 i 5.



Slika 3.6 Aktivnosti zaostatka proizvoda [Sommerville 2021, Fig. 2.5]

UREĐIVANJE ZAOSTALIH PREDMETA U ZAOSTATKU PROIZVODA

Zaostali poslovi u listi zaostatka proizvoda se mogu preuređovati pomoću 4 aktivnosti: prećišćavanje, procena, kreiranje i prioritizacija

Scrum zajednica ponekad koristi izraz „uređivanje zaostalih predmeta“ da pokrije ove četiri aktivnosti:

1. **Prečišćavanje:** Postojeći PBI se analiziraju i prerađuju kako bi se kreirali detaljniji PBI. Ovo takođe može dovesti do stvaranja novih zaostalih stavki.
2. **Procena:** Tim procenjuje količinu posla potrebnog za implementaciju PBI i dodaje ovu procenu svakom analiziranom PBI
3. **Kreiranje:** Nove stavke se dodaju u zaostatak. To mogu biti nove funkcije koje je predložio menadžer proizvoda, potrebne promene karakteristika, inženjerska poboljšanja ili procesne aktivnosti kao što je procena razvojnih alata koji bi se mogli koristiti.
4. **Prioritizacija:** Određivanje prioriteta PBI-ovi su preuređeni da primaju nove informacije i promenjene okolnosti.

Procene PBI daju indikaciju napora koji je potreban da bi se završila svaka stavka. Obično se koriste dve metrike:

1. **Potreban napor**: Količina truda može biti izražena u satima ili u danima osoba—to jest, u broju sati ili dana koji bi jednoj osobi bio potreban da implementira taj PBI. Ovo nije isto što i kalendarsko vreme. Nekoliko ljudi može raditi na stavci, što može skratiti potrebno kalendarsko vreme. Alternativno, programer može imati druge odgovornosti

koje sprečavaju puno radno vreme na projektu. Tada je potrebno kalendarsko vreme duže od procene napora.

2. **Tačke priče:** Bodovi priče su proizvoljna procena napora uključenih u implementaciju PBI, uzimajući u obzir veličinu zadatka, njegovu složenost, tehnologiju koja može biti potrebna i „nepoznate“ karakteristike posla. Tačke priče su prvobitno izvedene poređenjem korisničkih priča, ali se mogu koristiti za procenu bilo koje vrste PBI. Tačke priče se procenjuju relativno. Tim se slaže oko tačaka priče za osnovni zadatak. Ostali zadaci se zatim procenjuju upoređivanjem sa ovom osnovnom linijom—na primer, manje ili više složeni, veći ili manji, itd. Prednost tačaka priče je u tome što su apstraktnije nego što je potreban napor jer bi sve tačke priče trebalo da budu iste, bez obzira na individualne sposobnosti

PROCENA NAPORA ZA REALIZACIJU STAVKI ZAOSTATKA PROIZVODA

Procene su zasnovane na subjektivnoj proceni članova tima, a početne procene su neizbežno pogrešne. Međutim, procene se obično poboljšavaju kako tim stiče iskustvo sa proizvodom.

Procena napora je teška, posebno na početku projekta kada tim ima malo ili nimalo prethodnog iskustva sa ovom vrstom posla ili kada se koriste tehnologije nove za tim. Procene su zasnovane na subjektivnoj proceni članova tima, a početne procene su neizbežno pogrešne. Međutim, procene se obično poboljšavaju kako tim stiče iskustvo sa proizvodom i procesom njegovog razvoja.

Scrum metod preporučuje timski baziran pristup proceni pod nazivom „Planiranje pokera“, u koji ovde ne ulazimo. Obrazloženje je da timovi treba da budu u stanju da naprave bolje procene od pojedinaca. Međutim, ne postoje ubedljivi empirijski dokazi koji pokazuju da je kolektivna procena bolja od procena iskusnih, individualnih programera.
Nakon što je nekoliko sprinteva završeno, postaje moguće da tim proceni njegovu „brzinu“. Pojednostavljeno, **brzina tima je zbir procena veličine predmeta koji su završeni tokom sprinta sa određenim vremenom.** Na primer, pretpostavite da se PBI procenjuju u poenima priče i da, u uzastopnim sprintevima, tim primenjuje 17, 14, 16 i 19 poena priče. Brzina tima je stoga između 16 i 17 poena priče po sprintu. Brzina se koristi za odlučivanje na koliko PBI-ova tim može realno da se posveti u svakom sprintu. U gornjem primeru, tim bi trebalo da se posveti oko 17 tačaka priče. Brzina se takođe može koristiti kao mera produktivnosti. Timovi treba da pokušaju da preciziraju način na koji rade tako da se njihova brzina poboljšava tokom projekta.
Zaostatak proizvoda je zajednički, „živi“ dokument koji je redovno ažuriran tokom razvoja proizvoda. Obično je prevelik da bi stao na tablu, pa je logično da ga *održavate kao zajednički digitalni dokument*. Nekoliko specijalizovanih alata koji podržavaju Scrum uključuju mogućnosti za deljenje i reviziju zaostalih proizvoda. Neke kompanije mogu odlučiti da kupe ove alate za svoje programere softvera.

Mala preduzeća ili grupe sa ograničenim resursima mogu da koriste zajednički sistem dokumenata kao što su Office 365 ili Google dokumenti. Ovi jeftini sistemi ne zahtevaju kupovinu i instaliranje novog softvera. Ako počinjete da koristite zaostale proizvode u procesu

razvoja, preporučujemo ovaj opšti pristup da biste stekli iskustvo pre nego što odlučite da li vam je potrebna specijalizovana alati za upravljanje zaostalim predmetima.

VREMENSKI OKGRANIČENI SPRINOVI

Na kraju vremenskog okvira, rad na aktivnosti prestaje bez obzira da li je planirani posao završen ili ne.

Scrum koncept koji je koristan u bilo kom agilnom procesu razvoja su sprintovi sa vremenskim okvirom. Vremenski okvir (timeboxing) znači da je određeno vreme dodeljeno za završetak aktivnosti. Na kraju vremenskog okvira, rad na aktivnosti prestaje bez obzira da li je planirani posao završen ili ne. Sprintovi su kratke aktivnosti (jedna do četiri nedelje) i odvijaju se između definisanih datuma početka i završetka. Tokom sprinta, tim radi na stavkama iz zaostatka proizvoda. **Proizvod je dakle razvijen u seriji sprintova**, od kojih svaki donosi povećanje proizvoda ili softvera za podršku.
Na slici 6, pokazujemo tri važne prednosti koje proizilaze iz upotrebe sprinteva sa vremenskim okvirima.

Slika 3.7 Koristi od vremenski ograničenih sprinova [Sommerville 2021, Fig.2.6]

TRI OSNOVNE AKTIVNOSTI SPRINTA

Zaostatak sprinta se kreira tokom procesa planiranja i pokreće razvojne aktivnosti kada se sprint izvrši

Svaki sprint uključuje tri osnovne aktivnosti:

1. **Planiranje sprinta:** Radne stavke koje treba da se završe tokom tog sprinta se biraju i, ako je potrebno, dorađuju kako bi se stvorio zaostali sprint. Ovo ne bi trebalo da traje više od jednog dana na početku sprinta.
2. **Izvođenje sprinta** :Tim radi na implementaciji zaostalih stavki sprinta koje su odabrane za taj sprint. Ako je nemoguće ispuniti sve stavke zaostalog sprinta, vreme za sprint se ne produžava. Umesto toga, nedovršene stavke se vraćaju u zaostatak proizvoda i stavljaju u red za budući sprint
3. **Pregled sprinta**: Rad obavljen tokom sprinta pregledaju tim i (moguće) spoljne zainteresovane strane. Tim razmišlja o tome šta je bilo dobro, a šta loše tokom sprinta, u cilju poboljšanja procesa rada

Slika 8 prikazuje ciklus ovih aktivnosti i detaljniji pregled izvođenja sprinta. Zaostatak sprinta se kreira tokom procesa planiranja i pokreće razvojne aktivnosti kada se sprint izvrši.

Slika 3.8 Aktivnosti sprinta [Sommerville 2021, Fig.2.7]

CILJEVI SPRINTA

Cilj sprinta je sažeta izjava o tome šta tim planira da postigne tokom sprinta

Svaki sprint treba da počne sastankom za planiranje na kojem članovi tima zajednički odlučuju o PBI-ovima koji će se primeniti tokom sprinta. Ulazi u ovu aktivnost su stavke zaostataka proizvoda koje su spremne za implementaciju i informacije od vlasnika proizvoda o tome koji od ovih PBI-ova ima najveći prioritet.

Kada planirate sprint, tim radi tri stvari:

- dogovoriti cilj sprinta;
- odlučiti o listi stavki iz zaostatka proizvoda koje treba implementirati;
- kreirajte zaostatak sprinta, detaljniju verziju zaostatka proizvoda koja beleži posao koji treba obaviti tokom sprinta

Cilj sprinta je sažeta izjava o tome šta tim planira da postigne tokom sprinta. To može biti implementacija neke karakteristike proizvoda, razvoj neke suštinske infrastrukture proizvoda ili poboljšanje nekog atributa proizvoda, kao što je njegov učinak. Trebalo bi biti moguće da se objektivno odluči da li je cilj postignut do kraja sprinta. Slika 8 prikazuje tri tipa ciljeva sprinta i daje primer svakog tipa.

Slika 3.9 Ciljevi sprinta [Sommerville 2021, Fig. 2.8]

ZOSTATAK SPRINTA

Zaostatak sprinta je lista radnih stavki koje treba završiti tokom sprinta.

Funkcionalni ciljevi sprinta odnose se na implementaciju sistemskih karakteristika za krajnje korisnike. Ciljevi performansi i pouzdanosti odnose se na poboljšanja performansi, efikasnosti, pouzdanosti i sigurnosti sistema. Ciljevi podrške pokrivaju pomoćne aktivnosti kao što je razvoj infrastrukturnog softvera ili projektovanje arhitekture sistema.
Uvek treba da uzmete u obzir stavke najvišeg prioriteta u zaostatku proizvoda kada odlučujete o cilju sprinta. Tim bira ove stavke za implementaciju u isto vreme kada se postavlja cilj sprinta. Ima smisla odabrati koherentan skup stavki visokog prioriteta koje su u skladu sa ciljem sprinta. Ponekad se biraju stavke nižeg prioriteta u zaostatku proizvoda jer su usko povezane sa drugim stavkama koje su deo ukupnog cilja sprinta.
Kao opšte pravilo, cilj sprinta ne bi trebalo da se menja tokom sprinta. Ponekad, međutim, cilj sprinta mora da se promeni ako se otkriju neočekivani problemi ili ako tim pronađe način

da primeni funkciju brže nego što je prvobitno procenjeno. U ovim slučajevima, obim cilja se može smanjiti ili proširiti.

Kada je cilj sprinta uspostavljen, tim treba da razgovara i odluči o planu sprinta. Kao što sam objasnio, PBI bi trebalo da imaju pridruženu procenu napora, što je kritičan doprinos procesu planiranja sprinta. Važno je da tim ne pokušava da primeni previše stvari tokom sprinta. Prekomerna posvećenost može onemogućiti postizanje cilja sprinta

Brzina tima je još jedan važan doprinos procesu planiranja sprinta. Brzina odražava koliko posla tim može normalno da pokrije u sprintu. Možete proceniti bodove priče kao što sam objasnio, pri čemu je brzina tima broj poena priče koje normalno može da primeni u dvonedeljnom ili četvoronedeljnom sprintu. Ovaj pristup očigledno ima smisla za tim koji ima stabilnu brzinu. Međutim, brzina tima može biti nestabilna, što znači da je broj završenih PBI-ova varira od jednog sprinta do drugog. Brzina može biti nestabilna ako se promeni članstvo u timu, ako se lakšim stavkama dodeli veći prioritet od predmeta koje je teže implementirati, ili ako tim uključuje neiskusne članove koji se poboljšavaju kako projekat napreduje. Ako je brzina tima nestabilna ili nepoznata, onda morate da preduzmete intuitivniji pristup odabiru broja PBI koji će se primeniti tokom sprinta. **Zaostatak sprinta je lista radnih stavki koje treba završiti tokom sprinta**. Ponekad se PBI može preneti direktno u zaostatak sprinta. Međutim, tim obično razlaže svaki PBI na manje zadatke koji dodaju se zaostatku sprinta. Svi članovi tima zatim razgovaraju o tome kako će ti zadaci biti raspoređeni. Svaki zadatak treba da ima relativno kratko trajanje — najviše jedan ili dva dana — tako da tim može da proceni svoj napredak tokom dnevnog sprint sastanka. Zaostatak sprinta treba da bude mnogo kraći od zaostatka proizvoda, tako da se može održavati na zajedničkoj beloj tabli. Ceo tim može da vidi koje stavke treba da se implementiraju i koje stavke su završen

FOKUS SPRINTA

Fokus sprinta je razvoj karakteristika proizvoda ili infrastrukture i tim radi na kreiranju planiranog povećanja softvera

Fokus sprinta je razvoj karakteristika proizvoda ili infrastrukture i tim radi na kreiranju planiranog povećanja softvera. Da bi olakšali saradnju, članovi tima koordiniraju svoj rad svaki dan na kratkom sastanku koji se naziva scrum (tabela T8). Scrum metod je dobio ime po ovim sastancima, koji su suštinski deo metode. Oni su način na koji timovi komuniciraju - ništa kao svađa u igri ragbija.

Scrum metod ne uključuje specifične prakse tehničkog razvoja; tim može koristiti sve agilne prakse za koje smatra da su prikladne. Neki timovi vole programiranje u paru; drugi više vole da članovi rade individualno. Međutim, preporučujem da se u razvoju koda uvek koriste dve prakse sprintovi:

Tabela T8: Scrum sastanci

Scrum je kratak, dnevni sastanak koji se obično održava na početku dana. Tokom scrum-a, svi članovi tima dele informacije, opisuju svoj napredak od prethodnog dana i iznose probleme koji su se pojavili i planove za naredni dan. Ovo znači da svi u timu znaju šta se dešava i, ako se pojave problemi, mogu ponovo da planiraju kratkoročni rad kako bi se nosili sa njima.
Scrum sastanci treba da budu kratki i fokusirani. Da bi se članovi tima odvratili od uključivanja u duge diskusije, scrums se ponekad organizuju kao „stand-up“ sastanci gde nema stolica u sali za sastanke.
Tokom trke, pregleda se zaostatak sprinta. Dovršene stavke se uklanjaju iz njega. Nove stavke se mogu dodati u zaostatak kako se pojave nove informacije. Tim tada odlučuje ko treba da radi na stavkama zaostalih u sprintu tog dana.

Slika 3.10 Srcum sastanci [Sommerville 2021, Table 2.8]

1. **Automatizacija testiranja** Koliko god je to moguće, testiranje proizvoda treba da bude automatizovano. Trebalo bi da razvijete paket izvršnih testova koji se mogu pokrenuti u bilo **kom trenutku. Objašnjavam kako se to radi u 9. poglavlju.**
2. **Kontinuirana integracija** Kad god neko izvrši promene u softverskim komponentama koje razvija, ove komponente treba odmah integrisati sa drugim komponentama kako bi se stvorio sistem. Ovaj sistem zatim treba testirati da bi se proverili neočekivani problemi u interakciji komponenti.

KOMPLETNOST KODA

Važno da tim uspostavi „definiciju gotovog“, koja precizira šta treba da se završi za kod koji se razvija tokom sprinta.

Cilj sprinta je da se razvije „potencijalno povećanje proizvoda za isporuku“. Naravno, softver neće nužno biti pušten kupcima, ali ne bi trebalo da zahteva dalji rad pre nego što bude pušten. To znači različite stvari za različite tipove softvera, tako da je važno da tim uspostavi „definiciju gotovog“, koja precizira šta treba da se završi za kod koji se razvija tokom sprinta. Na primer, za softverski proizvod koji se razvija za spoljne kupce, tim može da napravi kontrolnu listu koja se primenjuje na sav softver koji se razvija. Tabela T9 je primer kontrolne liste koja se može koristiti za procenu kompletnosti implementirane funkcije.

Tabela T9: Lista za proveru (čeklista) kompletnosti koda

Status stavke	Opis
Pregledano	Kod je pregledao drugi član tima koji je proverio da li ispunjava dogovorene standarde kodiranja, da li je razumljiv, uključuje odgovarajuće komentare i refaktorisan je ako neophodno.
Jedinica testirana	Svi testovi jedinica su pokrenuti automatski i svi testovi su uspešno izvršeni.
Integrisano	Kod je integrisan sa projektnom kodnom bazom i nisu prijavljene greške u integraciji.
Integracija testirana	Svi integracijski testovi su pokrenuti automatski i svi testovi su uspešno izvršeni.
Prihvaćeno	Testovi prihvatanja su sprovedeni ako je bilo potrebno i vlasnik proizvoda ili razvojni tim su potvrdili da je stavka zaostalog zaostatka proizvoda završena.

Slika 3.11 Lista za proveru kompletnosti koda [Sommerville 2021, Table 2.9]

PREGLED SPRINTA

Pregleda se da li je sprint ispunio svoj cilj. Postavlja sve nove probleme i probleme koji su se pojavili tokom sprinta. To je način da tim razmišlja o tome kako može da poboljša način rada.

Ako nije moguće završiti sve stavke na ovoj kontrolnoj listi tokom sprinta, nedovršene stavke treba dodati u za ostatak proizvoda za buduću primenu. Sprint nikada ne bi trebalo da se produžava na potpune nedovršene stavke.

Na kraju svakog sprinta, postoji **sastanak za pregled** koji uključuje ceo tim. Ovaj sastanak ima tri svrhe. Prvo, pregleda se da li je sprint ispunio svoj cilj. Drugo, postavlja sve nove probleme i probleme koji su se pojavili tokom sprinta. Konačno, to je način da tim razmišlja o tome kako može da poboljša način na koji rade. Članovi raspravljaju o tome šta je prošlo dobro, šta loše i koja poboljšanja bi se mogla napraviti.

Pregled može uključiti spoljne zainteresovane strane kao i razvojni tim. Tim treba da bude iskren u pogledu onoga što je postignuto, a šta nije tokom sprinta, tako da rezultat pregleda bude konačna procena stanja proizvoda koji se razvija. Ako su stavke nedovršene ili ako su identifikovane nove stavke, treba ih dodati u zaostatak proizvoda. Vlasnik proizvoda ima krajnje ovlašćenje da odluči da li je cilj sprinta postignut ili ne. Oni treba da potvrde da je implementacija izabranih zaostalih stavki proizvoda završena.

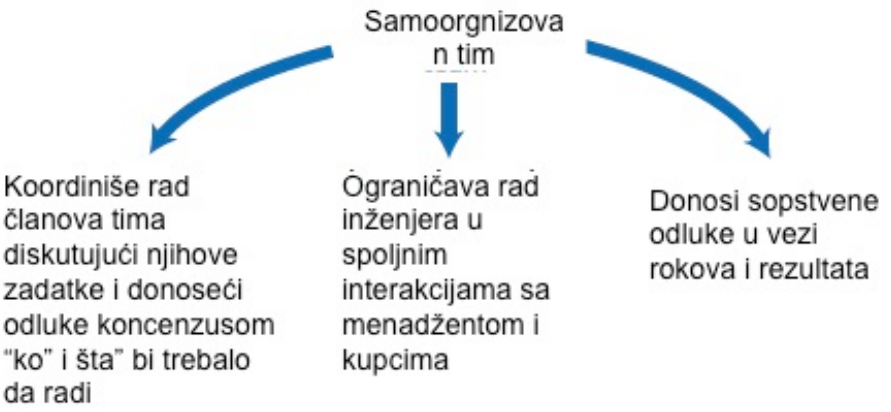
Važan deo pregleda sprinta je pregled procesa, u kome tim razmišlja o sopstvenom načinu rada i načinu na koji je Scrum korišćen. Cilj pregleda procesa je da se identifikuju načini za poboljšanje i da se razgovara o tome kako da se Scrum koristi produktivnije. Tokom razvojnog procesa, Scrum tim treba da pokušava da stalno poboljšava svoju efikasnost.

Tokom pregleda, tim može razgovarati o kvarovima u komunikaciji, dobrim i lošim iskustvima sa alatima i razvojnim okruženjem, tehničkim praksama koje su usvojene, softveru i bibliotekama za višekratnu upotrebu koji su otkriveni i drugim pitanjima. Ako su problemi identifikovani, tim bi trebalo da razgovara o tome kako ih treba rešiti u budućim sprintovima. Na primer, može se doneti odluka da se istraže alternativni alati za one koje koristi tim. Ako su aspekti rada bili uspešni, tim može eksplicitno rasporedite vreme tako da se iskustvo može podeliti i dobra praksa usvojiti u celom timu.

SAMOORGANIZOVANI TIMOVI

Samoorganizovani timovi nemaju menadžera projekta koji dodeljuje zadatke i donosi odluke za tim

Osnovni princip svih metoda agilnog razvoja je da tim za razvoj softvera treba da bude samoorganizovan. Samoorganizovani timovi nemaju menadžera projekta koji dodeljuje zadatke i donosi odluke za tim. Umesto toga, kao što je prikazano na slici 12, oni sami donose odluke. Samoorganizovani timovi rade tako što raspravljaju o pitanjima i donose odluke konsenzusom.



Slika 3.12 Samoorganiyovani tim [Sommerville 2021, Fig.2.9]

Idealna veličina Scrum tima je između pet i osam ljudi — dovoljno velika da bude raznolika, ali dovoljno mala da komunicira neformalno i efikasno i da se dogovori oko prioriteta tima. Pošto timovi moraju da se pozabave različitim zadacima, važno je imati niz stručnosti u Scrum timu, kao što su umrežavanje, korisničko iskustvo, dizajn baze podataka i tako dalje.

U stvarnosti, možda nije moguće formirati idealne timove. U nekomercijalnom okruženju kao što je univerzitet, timovi su manji i sastavljeni od ljudi koji imaju uglavnom isti skup veština. Širom sveta postoji nedostatak softverskih inženjera, tako da je ponekad nemoguće pronaći ljude sa pravom mešavinom veština i iskustva. Tim se može promeniti tokom projekta kako ljudi odlaze i zapošljavaju se novi članovi. Neki članovi tima mogu raditi sa skraćenim radnim vremenom ili od kuće.

Prednost efikasnog samoorganizovanog tima je u tome što može da bude kohezivan i da se prilagođava promenama. Pošto tim, a ne pojedinci, preuzima odgovornost za rad, tim može da se nosi sa ljudima koji odlaze i pridružuju se grupi. Dobra timska komunikacija znači

da članovi tima neizbežno nauče nešto o oblastima jedni drugih. Oni stoga mogu donekle nadoknaditi kada ljudi napuste tim.
U upravljanom timu, rukovodilac projekta koordinira rad. Menadžeri gledaju na posao koji treba obaviti i dodeljuju zadatke članovima tima. Menadžeri projekta moraju da urede stvari tako da se posao ne odlaže jer jedan član tima čeka da drugi završe posao. Moraju da kažu svim članovima tima o problemima i drugim faktorima koji mogu odložiti rad. Članovi tima se ne podstiču da preuzmu odgovornost za koordinaciju i komunikaciju.

KOORDINACIJA TIMA

Članovi tima objašnjavaju svoj rad i svesni su napretka tima i mogućih rizika koji mogu uticati na tim

U timu koji se samoorganizuje, sam tim mora da uspostavi načine za koordinaciju rada i saopštavanje problema svim članovima tima. Programeri Scrum-a su pretpostavili da su članovi tima ko-locirani. Oni rade u istoj kancelariji i mogu neformalno da komuniciraju. Ako jedan član tima treba da zna nešto o tome šta je drugi uradio, oni jednostavno razgovaraju jedni sa drugima da bi saznali. Nema potrebe da ljudi dokumentuju svoj rad da bi ih drugi pročitali. Svakodnevni scrum znače da članovi tima znaju šta je urađeno i šta drugi rade.

Scrum pristup utjelovljuje osnove za koordinaciju u timu kojim se samostalno upravlja — naime, dobru neformalnu komunikaciju i redovne sastanke kako bi se osiguralo da su svi u toku. Članovi tima objašnjavaju svoj rad i svesni su napretka tima i mogućih rizika koji mogu uticati na tim. Međutim, postoje praktični razlozi zašto neformalna verbalna komunikacija možda ne funkcioniše uvek:

1. Scrum pretpostavlja da je tim sastavljen od radnika sa punim radnim vremenom koji dele radni prostor. U stvarnosti, članovi tima mogu biti honorarni i mogu raditi na različitim mestima. Za studentski projekat, članovi tima mogu pohađati različite časove u različito vreme, tako da može biti teško pronaći vremenski termin gde se svi članovi tima mogu sastati.
2. Scrum pretpostavlja da svi članovi tima mogu prisustvovati jutarnjem sastanku kako bi koordinirali rad tokom dana. Ovo ne uzima u obzir da članovi tima mogu da rade fleksibilno radno vreme (na primer, zbog obaveza za brigu o deci) ili mogu da rade skraćeno radno vreme na nekoliko projekata. Oni, dakle, nisu dostupni svakog jutra.

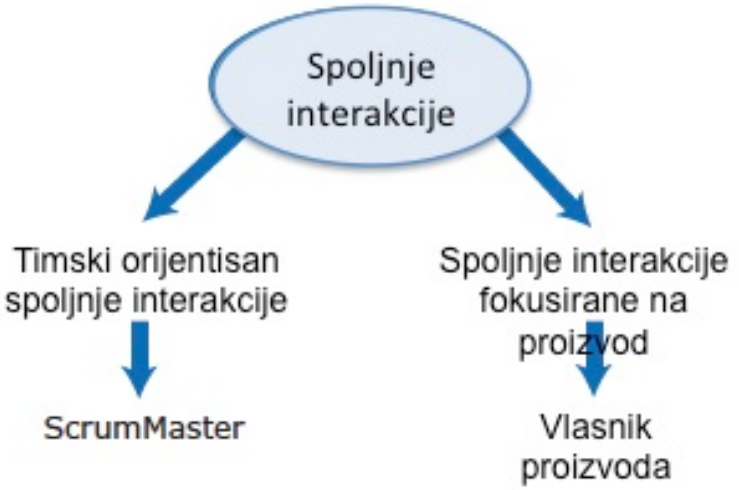
Ako je zajednički rad sa dnevnim sastancima nepraktičan, onda tim mora da osmisli druge načine komunikacije. Sistemi za razmenu poruka, kao što je Slack, mogu biti efikasni za neformalnu komunikaciju. Prednost razmene poruka je što se sve poruke snimaju tako da ljudi mogu da nadoknade razgovore koje su propustili. Razmena poruka nema neposrednost komunikacije licem u lice, ali je bolja od e-pošte ili deljenih dokumenata za koordinaciju.

UPRAVLJANJE SPOLJNIM INTERAKCIJAMA

ScrumMaster i vlasnik proizvoda treba da budu zajednički odgovorni za upravljanje interakcijama sa ljudima izvan tima

Razgovor jedni s drugima je najbolji način da članovi tima koordiniraju rad i da komuniciraju šta je dobro prošlo i koji su problemi nastali. Dnevni sastanci mogu biti nemogući, ali agilni timovi zaista moraju redovno da zakazuju sastanke o napretku čak i ako svi članovi ne mogu da prisustvuju ili moraju da prisustvuju virtuelno koristeći telekonferencije. Članovi koji ne mogu da prisustvuju treba da podnesu kratak rezime svog napretka kako bi tim mogao da proceni koliko dobro ide posao.Svi razvojni timovi, čak i oni koji rade u malim startapima ili nekomercijalnim razvojima, imaju neke spoljne interakcije. Neke interakcije će pomoći timu da razume šta klijenti zahtevaju od softvera proizvoda koji se razvija. Drugi će biti sa menadžmentom kompanije i drugim delovima kompanije, kao što su ljudski resursi i marketing.

U Scrum projektu, ScrumMaster i vlasnik proizvoda treba da budu zajednički odgovorni za upravljanje interakcijama sa ljudima izvan tima (slika 10)Vlasnici proizvoda su odgovorni za interakciju sa trenutnim i potencijalnim kupcima, kao i za prodajno i marketinško osoblje kompanije. Njihov posao je da razumeju šta kupci traže u softverskom proizvodu i da identifikuju moguće prepreke za usvajanje i upotrebu proizvoda koji se razvija. Trebalo bi da razumeju inovativne karakteristike proizvoda kako bi utvrdili kako kupci mogu imati koristi od njih. Vlasnici proizvoda koriste ovo znanje da pomognu u razvoju zaostalih proizvoda i da daju prioritet zaostalim stavkama za implementaciju.



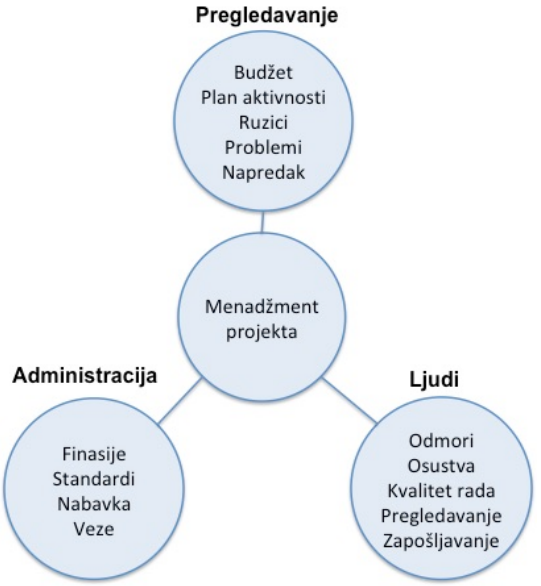
Slika 3.13 Upravljanje spoljnim interakcijama [Sommerville 2021, Fig.2.10]

Uloga ScrumMaster-a ima dvostruku funkciju. Deo uloge je i bliski rad sa timom, obučavajući ih u korišćenju Scrum-a i radeći na razvoju zaostatka proizvoda. Scrum vodič navodi da ScrumMaster takođe treba da radi sa ljudima van tima kako bi „uklonio prepreke“; odnosno treba da se bave eksternim problemima i upitima i predstavljaju tim široj organizaciji. Namera je da tim bude u stanju da radi na razvoju softvera bez spoljnih smetnji ili ometanja.

ODGOVORNOST ZA UPRAVLJANJE PROJEKTOM

Posao ScrumMaster-a je da pomogne članovima tima da efikasno koriste Scrum metod.

Uloga ScrumMaster-a ima dvostruku funkciju. Deo uloge je i bliski rad sa timom, obučavajući ih u korišćenju Scrum-a i radeći na razvoju zaostatka proizvoda. Scrum vodič navodi da ScrumMaster takođe treba da radi sa ljudima van tima kako bi „uklonio prepreke“; odnosno treba da se bave eksternim problemima i upitima i predstavljaju tim široj organizaciji. Namera je da tim bude u stanju da radi na razvoju softvera bez spoljnih smetnji ili ometanja.
Bez obzira da li tim koristi Scrum ili neki drugi agilan pristup, morate obratiti pažnju na ova pitanja. U malim timovima, možda je nemoguće da različiti ljudi vode računa o interakciji sa kupcima i interakcijama sa menadžerima. Najbolji pristup može biti da jedna osoba preuzme obe ove uloge i da honorarno radi na razvoju softvera. Ključni zahtev za „spoljne komunikatore“ je dobra komunikacija i veštine ljudi kako bi mogli da pričaju o radu tima na način na koji ljudi izvan tima može da razume i da se odnosi na. ScrumMaster nije konvencionalni menadžer projekta. Posao je da se pomogne članovima tima da efikasno koriste Scrum metod i da se osigura da ih ne ometaju spoljni obziri. Međutim, u svim komercijalnim projektima neko mora da preuzme suštinske odgovornosti za upravljanje projektima (slika 13).
Bez obzira da li tim koristi Scrum ili neki drugi agilan pristup, morate obratiti pažnju na ova pitanja. U malim timovima, možda je nemoguće da različiti ljudi vode računa o interakciji sa kupcima i interakcijama sa menadžerimhonorarno radi na razvoju softvera. Ključni zahtev za „spoljne komunikatore“ a. Najbolji pristup može biti da jedna osoba preuzme obe ove uloge i da honorarno radi na rayvoju softvera.
Ključni zahtev za „spoljne komunikatore“ je dobra komunikacija i veštine ljudi kako bi mogli da pričaju o radu tima na način na koji ljudi izvan tima može da razume i da se odnosi na.
ScrumMaster nije konvencionalni menadžer projekta. Posao je da se pomogne članovima tima da efikasno koriste Scrum metod i da se osigura da ih ne ometaju spoljni obziri. Međutim, u svim komercijalnim projektima neko mora da preuzme suštinske odgovornosti za upravljanje projektima (slika 14).



Slika 3.14 Odgovornost i upravljanje projektim [Sommerville 2021, Fig.2.11]

SCRUMMASTER U RUKOVOĐENJU PROJEKTIMA

Nije realno da uloga ScrumMastera isključi odgovornosti za upravljanje projektima.

Scrum vodič i mnoge Scrum knjige jednostavno ignorišu ova pitanja. Ali oni su realnost rada u svim kompanijama osim u najmanjim. Tim koji se samoorganizuje mora da imenuje člana tima koji će preuzeti zadatke upravljanja. Zbog potrebe da se održi kontinuitet komunikacije sa ljudima van grupe, podela zadataka upravljanja među članovima tima nije održiv pristup.
Kao odgovor na ovo pitanje, Rubin sugeriše da bi ponekad moglo biti prikladno da menadžer projekta van tima radi za nekoliko Scrum timova. Međutim ova ideja neizvodljiva iz tri razloga:

1. Mala preduzeća možda nemaju resurse za podršku posvećenim menadžerima projekata.
2. Mnogi zadaci upravljanja projektima zahtevaju detaljno poznavanje tima rad. Ako menadžer projekta radi u nekoliko timova, možda i jeste nemoguće je detaljno upoznati rad svakog tima.
3. Timovi koji se samoorganizuju su kohezivni i imaju tendenciju da negoduju kada im ljudi izvan tima govore šta da rade. Članovi su skloni da ometaju, a ne podržavaju, eksternog menadžera projekta.

Nije realno da uloga ScrumMastera isključi odgovornosti za upravljanje projektima. ScrumMasteri znaju posao koji se odvija i u daleko su najboljoj poziciji da pruže tačne informacije i planove projekta i napredak.

HIGH LEVEL SCRUM PROCESS - GEORGIA TECH - SOFTWARE DEVELOPMENT PROCESS (VIDEO)

Scrum proces sa visokim nivoom apstrakcije

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Poglavlje 4

Vežba

ZADACI

Zadaci za rad na oba časa vežbi

U okviru časova vežbi, a i u okviru samostalnog studiranja kod kuće, ovde se navode zadaci na koje bi student trebalo da daju svoje odgovoredaci se rade na oba časa vežbanja

1. Objasnite zašto je važno da se softverski proizvodi brzo razvijaju i isporučuju. Zašto je ponekad razumno isporučiti nedovršeni proizvod, a zatim izdati nove verzije tog proizvoda nakon isporuke?
2. Objasnite zašto su osnovni ciljevi agilnog softverskog inženjeringa u skladu sa ubrzanim razvojem i isporukom softverskih proizvoda.
3. Navedite tri razloga zašto se ekstremno programiranje, kako ga predviđaju njegovi programeri, ne koristi široko.
4. Razvijate softverski proizvod koji će vam pomoći da upravljate prijemom studenata na univerzitet. Vaš agilni razvojni tim predlaže da kreiraju nekoliko malih izdanja koja potencijalni kupci mogu da isprobaju, a zatim daju povratne informacije. Prokomentarišite ovu ideju i predložite zašto možda nije prihvatljiva za korisnike sistema.
5. Objasnite zašto Vlasnik proizvoda igra ključnu ulogu u Scrum razvojnom timu. Kako razvojni tim koji radi u okruženju u kojem nema spoljnih kupaca (npr. tim studentskog projekta) može da reprodukuje ovu ulogu vlasnika proizvoda?
6. Zašto je važno da svaki sprint normalno proizvodi potencijalno povećanje proizvoda? Kada bi tim mogao da ublaži ovo pravilo i proizvede nešto što nije spremno za isporuku?
7. Objasnite zašto procena napora potrebnog da se završi stavka zaostalih radova na proizvodu korišćenjem radnih sati ili radnih dana može dovesti do značajnih varijacija između procenjenog i stvarnog napora.
8. Zašto će svakodnevne svađe verovatno smanjiti vreme koje je normalno potrebno da novi članovi tima postanu produktivni?
9. Jedan od problema sa samoorganizovanim timovima je taj što iskusniji članovi tima imaju tendenciju da dominiraju diskusijama i stoga utiču na način rada tima. Predložite načine da se suprotstavite ovom problemu.
10. Scrum jerazvijen za upotrebu od strane tima od pet do osam ljudi koji zajedno rade na razvoju softverskog proizvoda. Koji problemi mogu nastati ako pokušate da koristite Scrum za projekte studentskog tima u kojima članovi rade zajedno na razvoju programa? Koji delovi Scrum-a može se koristiti u ovoj situaciji?

Poglavlje 5

Zaključak

ZAKLJUČAK

Glavne poruke ove lekcije

1. Najbolji način za razvoj softverskih proizvoda je korišćenje agilnih metoda softverskog inženjeringa koje su usmerene na brz razvoj i isporuku proizvoda.
2. Agilne metode se zasnivaju na iterativnom razvoju i minimiziranju troškova tokom procesa razvoja.
3. Ekstremno programiranje (KSP) je uticajna agilna metoda koja je uvela prakse agilnog razvoja kao što su korisničke priče, razvoj prvi test i kontinuirana integracija. Ovo su sada glavne aktivnosti razvoja softvera.
4. Scrum je agilan metod koji se fokusira na agilno planiranje i upravljanje. Za razliku od KSP-a, on ne definiše inženjerske prakse koje će se koristiti. Razvojni tim može koristiti bilo koju tehničku praksu koju smatra prikladnom za proizvod koji se razvija.
5. U Scrum-u, posao koji treba obaviti se održava u zaostatku proizvoda, listi radnih stavki koje treba završiti. Svaki inkrement softvera implementira neke od radnih stavki iz zaostatka proizvoda.
6. Sprintovi su aktivnosti sa fiksnim vremenom (obično dve do četiri nedelje) u kojima se razvija povećanje proizvoda. Povećanja bi trebalo potencijalno da se isporučuju; odnosno da im ne treba dalji rad pre nego što budu isporučeni.
7. Tim koji se samoorganizuje je razvojni tim koji organizuje posao koji treba obaviti diskusijom i dogovorom među članovima tima.
8. Scrum prakse, kao što su zaostaci proizvoda, sprintovi i samoorganizovani timovi, mogu se koristiti u bilo kom agilnom razvojnom procesu, čak i ako se drugi aspekti Scrum-a ne koriste

LITERATURA

Reference korišćene u ovoj lekciji

Ova lekcija je pripremljena korišćenjem sledećeg udžbenika:

- [Sommerville 2021] Ian Sommerville, Engineering Software Products - An Introduction to Modern Software Engineering, Global Edition,Chapter 2, Pearson (2021)

Dodane, opcione reference

- The agile Mainset, https://blog.scottlogic.com/2014/09/18/the-agile-mindset.html

- The Advantages and Disadvantages of Agile SCRUM Software Development, http://www.my-project-management-expert.com/the-advantages-and-disadvantages-of-agile-scrum-software-development.html
- Citisism of Scrum, https://www.aaron-gray.com/a-criticism-of-scrum/
- Manifesto for Agile Software Development , http://agilemanifesto.org/