UNIVERZITET METROPOLITAN BEOGRAD

SE322 - INŽENJERSTVO ZAHTEVA

Izrada prototipova i postavljanje prioriteta zahteva

Lekcija 12

PRIRUČNIK ZA STUDENTE

SE322 - INŽENJERSTVO ZAHTEVA

Lekcija 12

IZRADA PROTOTIPOVA I POSTAVLJANJE PRIORITETA ZAHTEVA

UVOD

Uvodne napomene

Korišćenje prototipova softvera je probni korak u razvoju softvera. To zahteve čini stvarnijima, oživljava slučajeve upotrebe i zatvara praznine u vašem razumevanju zahteva. Korišćenje prototipova postavlja pred korisnike nacrt ili početni deo novog sistema da bi podstakao njihovo razmišljanje i katalizirao dijalog sa zahtevima. Rane povratne informacije o prototipovima pomažu akterima da postignu zajedničko razumevanje zahteva sistema, što smanjuje rizik od nezadovoljstva kupaca.

Reč prototip ima višestruka značenja, a učesnici u aktivnosti izrade prototipa mogu imati vrlo različita očekivanja. Prototip aviona zapravo leti - to je prva instanca nove vrste aviona. Suprotno tome, prototip softvera je samo deo ili model stvarnog sistema - možda uopšte ne donosi ništa korisno. Prototipi softvera mogu biti statički dizajni interfejsa ili radni modeli; brze skice ili vrlo detaljni ekrani; vizuelni displeji ili puni delovi funkcionalnosti; ili simulacije
Ova lekcija opisuje kako prototipiranje daje vrednost projektu i različite vrste prototipova koje možete kreirati u različite svrhe. Takođe nudi smernice o tome kako ih koristiti tokom razvoja zahteva, kao i načine na koji će prototipiranje biti efikasan deo vašeg softverskog inženjerskog procesa.

UVODNI VIDEO

Trajanje video snimka: 1min 12sek

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

Poglavlje 1

Zašto su nam prototipovi potrebni?

VIDEO PREDAVANJE ZA OBJEKAT "ZAŠTO SU NAM PROTOTIPOVI POTREBNI?"

Trajanje video snimka: 27min 55sek

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

KOJA JE SVRHA PROTOTIPA SOFTVERA?

Prototip softvera je delimična, moguća ili preliminarna primena predloženog novog proizvoda.

Prototip softvera je delimična, moguća ili preliminarna primena predloženog novog proizvoda. Prototipovi mogu poslužiti u tri glavne svrhe i ta svrha mora biti jasna od samog početka:

- **Razjašnjenje, kompletiranje i potvrđivanje zahteva:** Korišteni kao alat za zahteve, prototip pomaže u postizanju sporazuma, pronalaženju grešaka i propusta i proceni tačnosti i kvaliteta zahteva. Korisnička evaluacija prototipa ukazuje na probleme sa zahtevima i otkriva previđene zahteve, koje možete ispraviti po niskoj ceni pre nego što konstruišete stvarni proizvod. Ovo je posebno korisno za delove sistema koji nisu dobro razumeti ili su posebno rizični ili složeni.
- **Istraživanje alternativnog dizajna korisničkog interfejsa:** Korišćen kao alat za dizajn korisničkog interfejsa, prototip omogućava zainteresovanim stranama da istražuju različite tehnike interakcije korisnika, zamisle konačni proizvod, optimiziraju upotrebljivost sistema i procene potencijalne tehničke pristupe. Prototipovi mogu demonstrirati izvodljivost zahteva kroz radne dizajne korisničkog interfejsa. Oni su korisni za potvrđivanje razumevanja zahteva programera pre nego što konstruišu stvarno rešenje.

- **Kreiranje podskupa koji će prerasti u krajnji proizvod:** Korišćen kao alat za izradu, prototip je funkcionalna primena podskupa proizvoda, koji se može razviti u kompletan proizvod, kroz niz ciklusa malih ciklusa razvoja. Ovo je siguran pristup samo ako je prototip pažljivo osmišljen sa eventualnim izdanjem koje je planirano od početka.

Primarni razlog za kreiranje prototipa je rešavanje nesigurnosti u ranom procesu razvoja. Ne trebate izraditi prototip celog proizvoda. Usredsredite se na područja visokog rizika ili poznate nejasnoće da biste odlučili koje delove sistema ćete uključiti u prototip i šta želite da naučite

iz procena prototipa. Prototip je koristan za otkrivanje i rešavanje nejasnoća i nepotpunosti u zahtevima. Korisnici, menadžeri i drugi netehnički interesni akteri otkrivaju da prototipovi daju nešto konkretno za razmišljanje dok se proizvod specificira i projektuje. Obavezno za svaki prototip koji napravite znajte i komunicirajte - zašto ga kreirate, šta očekujete da naučite iz njega i šta ćete raditi sa prototipom nakon što su ga ljudi procenili.

TRI KLASE ATIBUTA PROTOTIPA

Svaki prototip koji napravite imaće određenu kombinaciju ovih atributa: obuhvata, buduća upotreba i oblik.

Zbog rizika od zbrke, važno je staviti neke deskriptore ispred reči „prototip“ kako bi učesnici projekta razumeli zašto i kada možete da napravite jednu ili drugu vrstu prototipa. Ovo poglavlje opisuje tri klase atributa prototipa, od kojih svaka ima dve alternative:

- **Obuhvat**: Model prototipa fokusiran je na korisničko iskustvo; prototip dokaza o konceptu istražuje tehničku ispravnost predloženog pristupa.
- **Buduća upotreba** Prototip izbacivanja se odbacuje nakon što je korišćen za generisanje povratnih informacija, dok evolutivni prototip izrasta u konačni proizvod kroz niz iteracija.
- **Oblik:** Papirni prototip je jednostavna skica nacrtana na papiru, beloj tabli ili u alatu za crtanje. Elektronski prototip se sastoji od radnog softvera za samo deo rešenja.

Svaki prototip koji napravite imaće određenu kombinaciju ovih atributa. Na primer, mogli biste osmisliti iscrtavanje papirnog ispisa s jednostavnim crtežima mogućih ekrana. Ili možete izraditi evolutivni elektronski dokaz.

Poglavlje 2

Prototipovi za bacanje i evolucijski prototipovi

MAKETA SOFTVERA

Maketa (mock-up) podrazumeva ponašanje bez postojanja softvera. Prikazuje fasade ekrana korisničkog interfejsa i omogućava neku navigaciju između njih.

Kad ljudi kažu „prototip softvera“, obično razmišljaju o izgledu mogućeg korisničkog interfejsa. Izgled se naziva i horizontalnim prototipom. Takav se prototip fokusira na deo korisničkog interfejsa; ne ulazi se duboko u sve arhitektonske slojeve ili u detaljne funkcionalnosti. Ova vrsta prototipa omogućava vam da istražite neka određena ponašanja željenog sistema, sa ciljem da se preciziraju zahtevi. Modeli pomažu korisnicima da presude da li će im sistem zasnovan na prototipu dozvoliti da svoj posao obavljaju na razuman način.

Maketa (mock-up) podrazumeva ponašanje bez postojanja softvera. Prikazuje fasade ekrana korisničkog interfejsa i omogućava neku navigaciju između njih, ali sadrži malo ili nimalo stvarne funkcionalnosti. Zamislite set za zapadnjački film: kauboj ulazi u salon i potom izlazi potpuno stabilno, a ipak ne pije piće i ne vidi konja jer iza lažnih kulisa zgrada nema ništa . Modeli mogu demonstrirati funkcionalne opcije koje će korisnik imati na raspolaganju, izgled korisničkog interfejsa (boje, izgled, grafiku, kontrole) i navigacionu strukturu. Navigacije mogu da funkcionišu, ali u određenim bi trenucima korisnik mogao videti samo poruku koja opisuje šta bi se stvarno prikazalo ili će otkriti da neke kontrole ne čine ništa.

Informacije koje se pojavljuju kao odgovor na upit baze podataka mogu biti lažne ili konstantne, a sadržaji izveštaja tvrdo su kodirani. Ako kreirate model, pokušajte da koristite stvarne podatke u oglednim prikazima i izlazima. Ovo povećava validnost prototipa kao modela stvarnog sistema, ali će procenjivačima prototipa biti jasno da su prikazani i izlazi simulirani, a ne aktivni.

Maketa ne radi ništa korisno, mada izgleda kao da bi trebalo. Simulacija je često dovoljno dobra da korisnici mogu na osnovu toga da odluče da li je bilo koja funkcionalnost nestala, pogrešna ili nepotrebna. Neki prototipovi predstavljaju koncept programera kako se može primeniti određeni slučaj upotrebe.

Korisničke evaluacije prototipa mogu ukazati na alternativne tokove za slučaj upotrebe, nedostajuće korake interakcije, dodatne izuzetke, predviđene post-uslove i relevantna pravila poslovanja.

PROTOTIP ZA BACANJE I PROTOTIP ZA DOKAZIVANJE KONCEPTA

Pri radu sa prototipom za bacanje korisnik treba da se usredsredi na široke zahteve i probleme sa radnim tokom. Prototipovi za izbacivanje, za evoluciju i za dokazivanje koncepta.

Pri radu sa **prototipom za bacanje**, korisnik treba da se usredsredi na široke zahteve i probleme sa radnim tokom, a da ne bude ometan preciznim izgledom elemenata ekrana. U ovoj fazi se ne brinite o tome gde će tačno biti pozicionirani elementi ekrana, fontovi, boje ili grafike. Vreme za istraživanje specifičnosti dizajna korisničkog interfejsa je posle razjašnjavanja zahteva i određivanja opšte strukture interfejsa. **Sa evolucijskim prototipom**, postepena ugradnja detalja približava korisnički interfejs svom izgledu planiranom za određeno izdanje

Prototip za dokazivanje koncepta, poznat i kao vertikalni prototip, implementira jedan isečak funkcionalnosti aplikacije iz korisničkog interfejsa kroz sve slojeve tehničkih usluga. Prototip dokazanog koncepta deluje kako da bi pravi sistem trebao da funkcioniše jer se dotiče svih nivoa implementacije sistema. Razvijte prototip za dokazivanje koncepta kada niste sigurni da li je predloženi arhitektonski pristup izvodljiv i zdrav, ili kada želite da optimizirate algoritme, procenite predloženu šemu baze podataka, potvrdite ispravnost rešenja u oblaku (cloud) ili testirate kritične zahteve za definisan vremenski interval.

Da bi rezultati bili smisleni, takvi prototipovi su konstruisani korišćenjem proizvodnih alata u radnom okruženju sličnom proizvodnji. Dokazivanje koncepta je takođe korisno za prikupljanje informacija kako bi se poboljšala sposobnost tima da proceni trud koji je uključen u implementaciju određene korisničke priče ili bloka funkcionalnosti. Agilni razvojni projekti ponekad prototip dokaza o konceptu nazivaju „špicom“.

PROTOTIP ZA BACANJE

Prototip koji se baca podržava brzu implementaciju i modifikaciju, na račun robusnosti, pouzdanosti, performansi i dugoročne održivosti.

Pre konstruiranja prototipa, treba doneti eksplicitnu i jasno saopštenu odluku da li će prototip biti istraživački ili će postati deo isporučenog proizvoda. U prvom slučaju, razvijate **prototip koji se kasnije baca** da biste pronašli odgovorili na pitanja, rešili nesigurnosti i poboljšali kvalitet zahteva. Budući da ćete odbaciti prototip nakon što je ispunio svoju svrhu, napravite ga što brže i jeftinije. Što više napora uložite u prototip, to više učesnici projekta nije voljno da ga odbaci i imate manje vremena na raspolaganju za izgradnju pravog proizvoda.

Ne morate bacati prototip ako vidite korist od njegovog korišćenja u budućnosti. Međutim, on neće biti ugrađen u isporučeni proizvod. Iz tog razloga, često se naziva i prototipom koji se ne izdaje.

Kada programeri izrade prototip za bacanje (throughaway prototype) , zanemaruju tehnike konstrukcije softvera. Prototip koji se izbacuje podržava brzu implementaciju i modifikaciju, na račun robusnosti, pouzdanosti, performansi i dugoročne održivosti. Iz tog razloga, on ne sme da se prevodi u proizvod. Ako to učinite, korisnici i održavači trpeće posledice po život proizvoda.

Prototip izbacivanja je najprikladniji kada se tim suoči sa nesigurnošću, dvosmislenošću, nekompletnošću ili nejasnoćom u zahtevima ili kada imaju poteškoća da sistem predstave samo iz zahteva.

Rešavanjem ovih pitanja smanjuje se rizik od nastavka razvoja proizvoda. Prototip koji pomaže korisnicima i programerima da vizualizuju kako se zahtevi mogu implementirati može otkriti nedostatke u zahtevima. Takođe omogućava korisnicima da procene da li će zahtevi omogućiti potrebne poslovne procese.

Žičani okvir (wireframe) je poseban pristup izradi prototipa koji se obično koristi za dizajn korisničkog interfejsa i dizajn veb lokacija. Možete da koristite žičane okvire da biste bolje razumeli tri aspekta veb stranice:

- Konceptualni zahtevi
- Informaciona arhitektura ili dizajn navigacije
- Detaljni dizajn stranica u visokoj rezoluciji

Skicirani izgledi ekrana, na ovaj način, ne moraju da liče na završne ekrane. Ovaj žični okvir je koristan za rad sa korisnicima da biste razumeli vrste aktivnosti koje bi mogli da izvrše na ekranu. Druga vrsta žičanog okvira ne mora uopšte da uključuje dizajniranje stranica. Dijaloška karta je odličan alat za istraživanje navigacije po veb stranama. Treća vrsta žičara unosi se u detalje kako bi izgledale završne stranice.

EVOLUCIJSKI PROTOTIP

Evolucijski prototip mora biti projektovan za lak rast i učestala unapređenja, tako da programeri moraju voditi računa o arhitekturi softvera i principima projektovanja.

Za razliku od prototipa koji je za bacanje, evolucijski prototip (evolutionary prototype) pruža solidnu arhitektonsku osnovu za postepeno građenje proizvoda kako zahtevi vremenom postanu jasni. Agilni razvoj daje primer evolucionog prototipiranja. Agilni timovi konstruišu proizvod kroz niz iteracija, koristeći povratne informacije o ranim iteracijama za prilagođavanje pravca budućih razvojnih ciklusa. To je suština evolucionog prototipiranja.

Za razliku od prototipa za odbacivanje, evolutivni prototip mora se od samog početka graditi sa robusnim kvalitetom. Zbog toga je za izradu evolucijskog prototipa potrebno duže vreme nego za prototipiranje koji simulira iste mogućnosti sistema. Evolucijski prototip mora biti projektovan za lak rast i učestala unapređenja, tako da programeri moraju voditi računa o arhitekturi softvera i primeniti čvrste principe projektovanja. Nema mesta za prečice u kvalitetu evolucijskog prototipa.

Zamislite prvu iteraciju evolucijskog prototipa kao pilot izdanje koje implementira početni deo zahteva. Lekcije naučene tokom testiranja prihvatanja od strane korisnika i početne upotrebe vode do modifikacija u sledećoj iteraciji. Potpuni proizvod je vrhunac niza ciklusa evolucionog prototipiranja. Takvi prototipovi brzo dobijaju korisnu funkcionalnost u ruke korisnika.

Evolucijski prototipovi dobro rade za aplikacije za koje znate da će vremenom rasti, ali to mogu biti korisne za korisnike bez primene svih planiranih funkcionalnosti. Agilni projekti se često planiraju tako da na kraju iteracije mogu zaustaviti razvoj i dalje imati proizvod koji je koristan kupcima, iako je nepotpun.

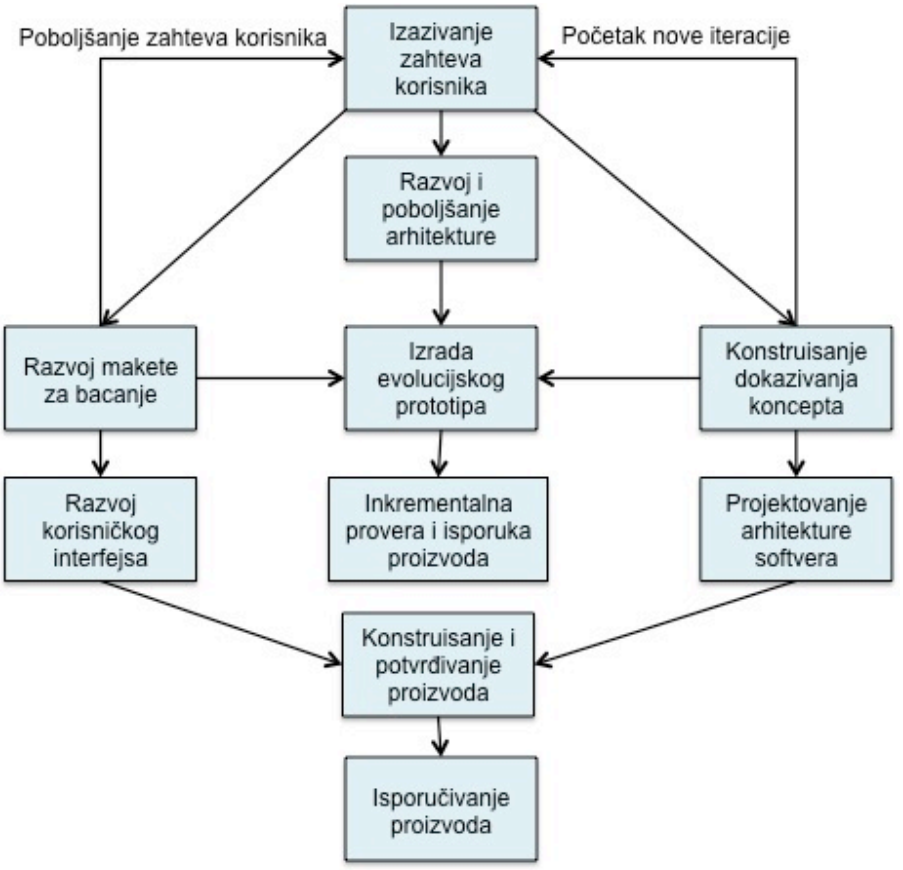
PRIMENENA PROTOTIPOVA ZA IZBACIVANJE I EVOLUCIJSKIH PROTOTIPOVA

Mogu se kombinovati primene prototipova za izbacivanje i evolucijskih prototipova.

Slika 2 ilustruje nekoliko mogućih načina kombinovanja različitih prototipova. Na primer, možete da upotrebite znanje stečeno iz niza prototipova koji se bacaju da biste precizirali zahteve, a koje biste kasnije mogli da postepeno primenite kroz evolucijski niz prototipova. Alternativni put kroz sliku 1 koristi model izbacivanja da bi se razjasnili zahtevi pre finalizacije dizajna korisničkog interfejsa, dok istovremeno se radi izrada prototipa za dokazivanje koncepta koji potvrđuje arhitekturu i osnovne algoritme. Ono što ne možete uspešno učiniti je da pretvorite namerno nizak kvalitet odbačenog prototipa u održivu robusnost koju zahteva proizvodni sistem. Pored toga, prototipovi za koje se čini da posao obavljaju za nekolicinu istovremenih korisnika verovatno neće dostići obim koji može da obradi hiljade korisnika bez većih arhitektonskih promena. Tabela na slici 1 rezimira neke tipične primene izbacivanja, evolucije, izvlačenja i prototipa dokaza o konceptu.

Svrha prototipa	Prototip za izbacivanje	Evolucijski prototip
Maketa	Razjašnjava i poboljšava korisničke i funkcionalne zahteve	Primena ključnih korisničkih zahteva
	Utvrđivanje nedostajućih zahteva	Primena dodatnih korisničkih zahteva u skladu sa prioritetima
	Istraživanje pristupa korisničkom interfejsu	Primeni u poboljšaj veb sajtove
		Radi sa sistemom koji je usklađen za brzim promenama poslovnih potreba
Dokazivanje koncepta	Demonstriranje tehičke prihvatljivosti	Primena i rast ključne višeslojne funkcionalnosti i komunikacijskih nivoa
	Ocenjivanje performansi	Primena i optimizacija ključnih algoritama
	Prikupljanje znanja za popravku pretpostavki za konstrukciju softvera	Testiranje i doterivanje performansi

Slika 2.1 Tipične primene prototipova u razvoju softvera [Wiegers]



Izvor: Karl Wiegers, Joy Beaty, Software Requirements, 3rd ed., Microsoft, 2013

Slika 2.2 Nekoliko slučajeva korišćenja prototipova u razvoju softvera

Poglavlje 3

Papirni i elektronski prototipovi

PROTOTIP NA PAPIRU

Papirni prototip je jeftin, brz i nisko-tehnološki način da se istraži kako deo jednog implementiranog sistema može da izgleda.

Nije vam uvek potreban izvršni prototip da biste rešili nesigurnosti zahteva. Papirni prototip (koji se ponekad naziva i prototip niske vernosti) je jeftin, brz i nisko-tehnološki način da se istraži kako deo jednog implementiranog sistema može izgledati. Prototipovi na papiru pomažu vam da testirate da li korisnici i programeri dele zajedničko razumevanje zahteva. Omogućavaju vam preduzimanje probnog i nisko rizičnog koraka u mogući prostor rešenja pre razvijanja proizvodnog koda. Koristite prototipove niske vernosti da biste istražili funkcionalnost i protok, a prototipovima visoke preciznosti odredite precizan izgled i osećaj.

Prototipovi na papiru (paper prototypes) uključuju alate koji nisu sofisticiraniji od papira, indeks kartica, nalepnice i bele table. Dizajner skicira ideje mogućih ekrana ne brinući se tačno gde se kontrole pojavljuju i kako izgledaju. Korisnici rado pružaju povratne informacije o dizajnu nacrtanom na komadu papira, mada su ponekad manje željni kritike prelepog računarskog prototipa u koji se čini da je programer uložio mnogo posla. I programeri bi se mogli odupreti značajnim promenama u pažljivo kreiranom elektronskom prototipu.

Kada se ocenjuje prototip niske vernosti, neko igra ulogu računara, dok korisnik prolazi kroz scenario ocenjivanja. Korisnik inicira akcije govoreći naglas što bi želio da radi na određenom ekranu: „Idem da izaberem Preview Print u meniju File (Datoteka). Osoba koja simulira računar zatim prikazuje komad papira ili indeksnu karticu koja predstavlja prikaz koji će se pojaviti kada korisnik preduzme tu akciju. Korisnik može proceni da li je to zaista očekivani odgovor i da li prikazana stavka sadrži ispravne elemente. Ako nije u redu, jednostavno uzmete praznu stranicu ili indeksnu karticu i pokušajte ponovo.

Bez obzira na to koliko su efikasni vaši alati za izradu prototipa, skice prikaza na papiru ili beloj tabli su brže. Prototipiranje papira omogućava brzu iteraciju, a iteracija je ključni faktor uspeha u razvoju zahteva. Prototipiranje papira je odlična tehnika za preciziranje zahteva pre dizajniranja detaljnih korisničkih interfejsa, konstrukcije evolucionog prototipa ili preduzimanja tradicionalnih dizajnerskih i građevinskih aktivnosti. Takođe pomaže razvojnom timu da upravlja očekivanjima kupaca.

ELEKTRONSKI PROTOTIPOVI

Elektronski prototipovi će vam omogućiti da lako implementirate i menjate komponente korisničkog interfejsa, bez obzira koliko je neefikasan privremeni kod koji stoji iza interfejsa.

Dostupno je mnogo alata ako se odlučite za izradu prototipa za elektronski prenos. Oni se kreću od jednostavnih alata za crtanje kao što su Microsoft Visio i Microsoft PowerPoint do komercijalnih alata za izradu prototipa i stvaranja grafičkih korisničkih interfejsa. Alatke su takođe dostupne posebno za izradu žičnih okvira veb stranica. Takvi alati će vam omogućiti da lako implementirate i menjate komponente korisničkog interfejsa, bez obzira koliko je neefikasan privremeni kod koji stoji iza interfejsa. Naravno, ako pravite evolucijski prototip, od početka morate koristiti alate za razvoj proizvodnog softvera.

Pošto se alati i njihovi dobavljači menjaju tako brzo, ovde nećemo predlagati konkretne. Komercijalni dostupni su različiti alati koji vam omogućavaju da simulirate aplikaciju pre nego što je napravite. Aplikaciona simulacija omogućava brzo sastavljanje izgleda ekrana, kontrola korisničkog interfejsa, protoka navigacije i funkcionalnosti u nešto što podseća na proizvod za koji mislite da treba da napravite. Sposobnost ponavljanja simulacije pruža vredan mehanizam za interakciju sa predstavnicima korisnika kako bi se razjasnili zahtevi i revidiralo vaše razmišljanje o rešenju.

Sa bilo kojom vrstom izrade prototipa - papirnim prototipovima, žičanim okvirima, elektronskim prototipima ili simulacijama - poslovni analitičar mora biti oprezan da se ne prerano uvuče u dizajn preciznog korisničkog interfejsa. Procenjivači prototipa često nude povratne informacije poput „Može li ovaj tekst biti malo tamnije crvene boje?“, „Pomerimo ovo polje samo malo“ ili „Ne sviđa mi se taj font.“ Osim ako svrha prototipa nije detaljna Dizajn ekrana ili veb stranica, takvi komentari su samo distrakcija. Boja, font i pozicioniranje okvira nisu bitni ako aplikacija ne podržava pravilno poslovne zadatke korisnika. Dok ne budete sigurni da ste dobro razumeli neophodnu funkcionalnost, usredsredite napore prototipiranja na preciziranje zahteva, a ne na vizuelni dizajn.

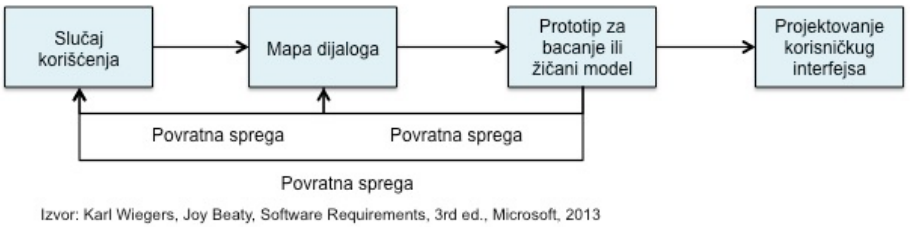
Poglavlje 4

Rad sa prototipovima

PRIMENA PROTOTIPOVA PRI RAZVOJU VEB APLIKACIJE SA KORISNIČKIM INTERFEJSOM

Ovaj progresivni način usavršavanja jeftiniji je od skoka direktno iz opisa slučajeva korišćenja do kompletne implementacije korisničkog interfejsa.

Na slici 1 prikazan je jedan mogući niz razvojnih aktivnosti kojim se kreće od slučajeva korišćenja do detaljnog dizajna korisničkog interfejsa uz pomoć prototipa koji odbacuje. Svaki opis **slučaja korišćenja** uključuje niz akcija aktera i odgovora sistema, koji možete da modelirate pomoću **dijaloške mape** da biste prikazali moguću arhitekturu korisničkog interfejsa. Prototip ili **žičani okvir** razrađuje elemente dijaloga u posebne ekrane, menije i dijaloške okvire. Kada korisnici procene prototip, njihove povratne informacije mogu dovesti do promena u opisima slučajeva korišćenja (ako je recimo, otkriven alternativni tok) ili do promena u dijaloškoj mapi. Nakon što se zahtevi usavrše i skiciraju ekrani, svaki element korisničkog interfejsa može biti optimizovan za upotrebljivost.



Izvor: Karl Wiegers, Joy Beaty, Software Requirements, 3rd ed., Microsoft, 2013

Slika 4.1 Redosled aktivnosti razvoja softvera primenom prtotipova za bacanje

Ove aktivnosti nije potrebno izvoditi strogo sekvencijalno. Iteracija na slučaj korišćenja, dijaloški mapu i žični okvir je najbolji način da se brzo postigne prihvatljiv i dogovoren pristup dizajniranju korisničkog interfejsa. Ovaj progresivni način usavršavanja jeftiniji je od skoka direktno iz opisa slučajeva korišćenja do kompletne implementacije korisničkog interfejsa i zatim otkrivanja glavnih problema koji zahtevaju obimno preispitivanje. Potrebno je da samo izvršite onoliko koraka u ovom nizu koji su potrebni da biste prihvatljivo smanjili rizik od greške u dizajnu korisničkog interfejsa. Ako je vaš tim uveren da razume zahteve, da su zahtevi dovoljno kompletni i da imaju dobro rešenje na pravom korisničkom interfejsu za izgradnju, tada nema puno smisla u prototipovanju. Takođe, izradu prototipova možete usredsrediti na zahteve korisnika koji imaju veliki rizik od greške ili veliki uticaj ako postoji problem. Jedan je projekat izvršio redizajn veb stranica za e-trgovinu za veliku korporaciju koju bi koristili milioni korisnika. Tim je izradio prototipove osnovnih elemenata veb lokacije, uključujući mrežni katalog, korpu za kupovinu i postupak odjave, kako bi bili sigurni da su ih

dobili ispravno prvi put. Oni su trošili manje vremena na istraživanju staza izuzetka i manje uobičajenih scenarija.

PRIMER ZA PEARLSFROMSAND.COM

Planiranje stranica na kojima bi se obezbedile veb stranice i zamisliti navigacione staze među njima

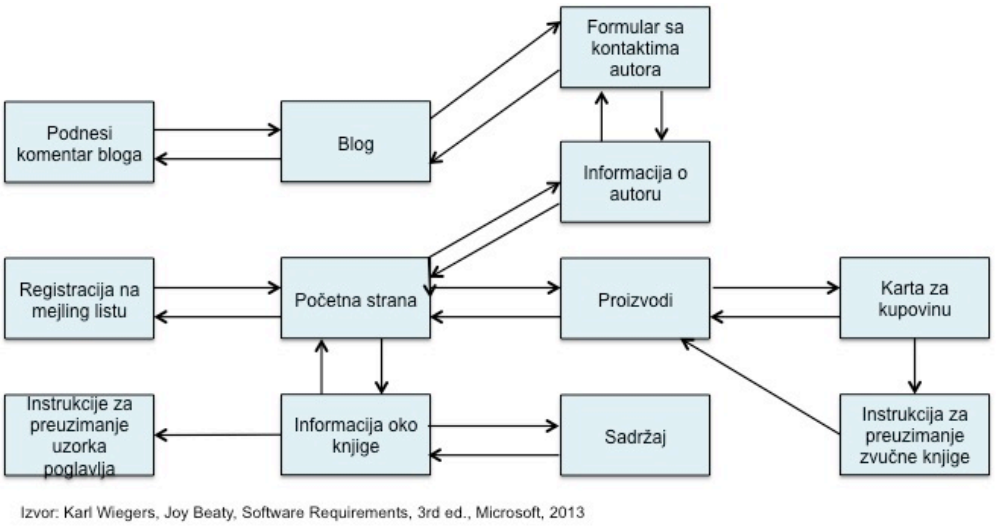
Da bismo ceo ovaj proces učinili opipljivijim, pogledajmo stvarni primer, malu veb lokaciju za promociju knjige, memoara životnih lekcija pod nazivom "Biseri iz peska" ("Pears from sand"). Autor knjige (Karl, u stvari) razmišljao je o nekoliko stvari koje bi posetioci trebali da urade na veb lokaciji, od kojih je svaka korisna. Postoje dodatni slučajevi korišćenja (ili upotrebe) za druge korisničke klase (slika 2).

Klase korisnika	Slučajevi korišćenja
Posetilac (Visitor)	Dobij informaciju o knjizi Dobij informaciju o autoru Čitanje uzorke poglavlja Čitanje bloga Kontakt sa autorom
Korisnik (Consumer)	Naručivanje proizvoda Preuzmi elektronski proizvod Zahtevanje pomoći u slučaju problema
Administrator	Upravljanje listom proizvoda Pitanja vraćanje novca kupcima Upravljanje liste meil adresa

Izvor: Karl Wiegers, Joy Beaty, Software Requirements, 3rd ed., Microsoft, 2013

Slika 4.2 Neki slučajevi korišćenja u veb aplikaciji PearlsFromSand.com

Sledeći korak je bio planiranje stranica na kojima bi se obezbedile veb stranice i zamisliti navigacione staze među njima. Konačna veb lokacija možda ne može implementirati sve ove stranice odvojeno. Neke se stranice mogu sažeti zajedno; drugi mogu funkcionisati kao skočni prozori ili druge modifikacije jedne stranice. Slika 3 ilustruje deo dijaloške mape koja ilustruje konceptualnu arhitekturu stranice. Svaki okvir predstavlja stranicu koja bi doprinela pružanju usluga identifikovanih u slučajevima korišćenja. Strelice predstavljaju veze koje omogućavaju navigaciju sa jedne stranice na drugu. Dok crtate dijalošku mapu, mogli biste otkriti nove radnje koje bi korisnik želeo da izvede. Dok radite kroz slučaj upotrebe, možda ćete pronaći načine da pojednostavite i pojednostavite korisničko iskustvo.



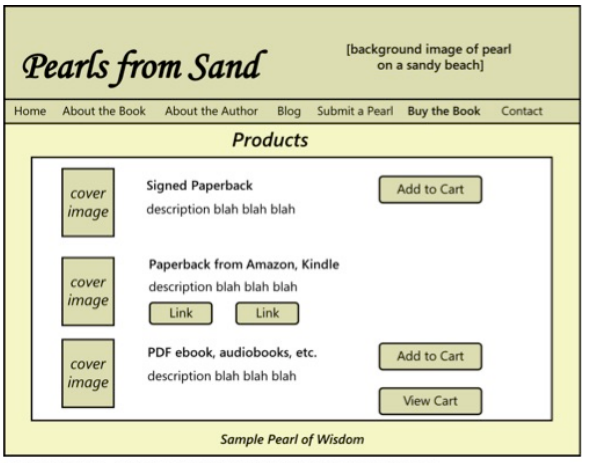
Izvor: Karl Wiegers, Joy Beaty, Software Requirements, 3rd ed., Microsoft, 2013

Slika 4.3 Mapa dijaloga za PearlsFromSand.com

DOBIJENI GRAFIČKI KORISNIČKI INTERFEJS

Ovaj iterativni pristup dizajniranju korisničkog interfejsa vodi do boljih rezultata od uranjanja odmah u dizajn stranice visoke rezolucije

Sledeći korak je bio konstruisanje prototipa ili žičanog okvira odabranih stranica kako bi se razvio pristup vizuelnog dizajna. Svaka od njih može biti ručno nacrtana skica na papiru, jednostavan crtež na liniji ili crtež kreiran sa namenskim prototipom ili kreiran alatom za vizuelno dizajniranje. Žičani okvir prikazan na slici 4 nacrtan je korišćenjem PowerPoint-a za samo nekoliko minuta. Takav jednostavan dijagram je alat za rad sa predstavnicima korisnika da bi se razumeli široki potezi kakav izgled stranice i kozmetičke karakteristike bi olakšale stranice za razumevanje i upotrebu.

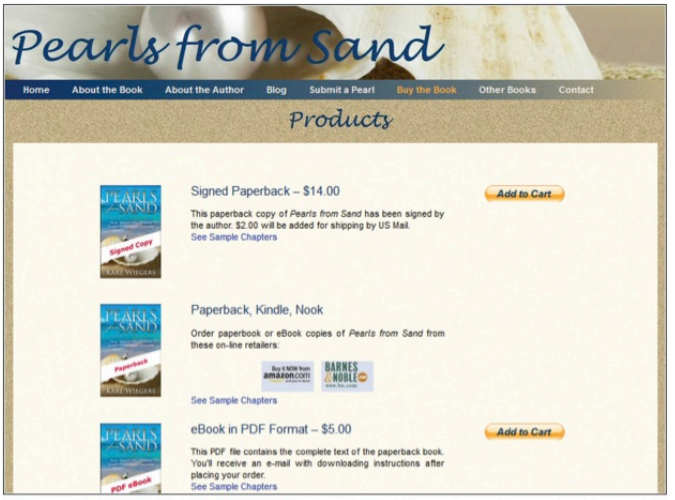


Izvor: Karl Wiegers, Joy Beaty, Software Requirements, 3rd ed., Microsoft, 2013

Slika 4.4 Žičani model jedne veb strane u PearlsFromSand.com

Konačno, četvrti korak prikazan na slici 1 je kreiranje detaljnog dizajna zaslona korisničkog interfejsa. Na slici 5 prikazana je jedna poslednja stranica sa veb stranice PearlsFromSand.com, koja je kulminacija aktivnosti analize i primene prototipova koji su se

koristili ranije. Ovaj iterativni pristup dizajniranju korisničkog interfejsa vodi do boljih rezultata od uranjanja odmah u dizajn stranice visoke rezolucije, bez jasnog razumevanja šta će pripadnici različitih klasa korisnika želeti da rade kada posete veb lokaciju.



Izvor: Karl Wiegers, Joy Beaty, Software Requirements, 3rd ed., Microsoft, 2013

Slika 4.5 Konačno urađena jedna veb strana za PearlsFromSand.com

Poglavlje 5

Vrednovanje protptipova

VIDEO PREDAVANJE ZA OBJEKAT "VREDNOVANJE PROTPTIPOVA"

Trajanje video snimka: 33min 41sek

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

PROCENA VREDNOSTI PROTOTIPOVA

Uključite članove više korisničkih klasa, iskusne i neiskusne. Naglasite da se on odnosi samo na deo funkcionalnosti.

Procena prototipa povezana je sa testiranjem upotrebljivosti. Naučićete više gledajući korisnike kako rade sa prototipom, nego samo tako što ćete tražiti da vam kažu šta misle o njemu. Pogledajte gde korisnikov prst ili pokazivač miša pokušavaju da idu instinktivno. Tačka mesta gde se prototip sukobljava sa ponašanjem drugih aplikacija koje evaluatori koriste. Procenjivač može pokušati neispravne prečice na tastaturi ili mora potražiti „miš okolo“ za ispravnu mogućnost menija. Potražite namrštene obrve koje označavaju zbunjenog korisnika koji ne može odrediti šta dalje raditi, kako se kretati do željenog odredišta ili kako krenuti u neki drugi deo aplikacije. Pogledajte da li prototip ima mrtve tačke, kao što se to ponekad dešava kada korisnik pošalje obrazac na veb lokaciji.

Neka pravi ljudi procene prototip iz odgovarajuće perspektive. Uključite članove više korisničkih klasa, i iskusne i neiskusne. Kada predstavite prototip evaluatorima, naglasite da se on odnosi samo na deo funkcionalnosti; ostalo će se implementirati kada se razvije stvarni sistem.

Da biste poboljšali procenu prototipa korisničkog interfejsa, napravite skripte koje će korisnike voditi kroz niz operacija i postaviti specifična pitanja kako biste dobili informacije koje tražite. Ovo dopunjuje opšti poziv za „recite mi šta mislite o ovom prototipu.“ Izvlačite skripte za procenu iz slučajeva upotrebe, korisničkih priča ili funkcija koje prototip adresira. Skripta traži od evaluatora da izvršavaju određene zadatke, radeći kroz delove prototipa koji imaju najviše neizvesnosti.

Na kraju svakog zadatka, a moguće i u srednjim tačkama, skripta prikazuje specifična pitanja vezana za zadatak. Možete postaviti i opšta pitanja poput sledećeg:

- Da li prototip implementira funkcionalnost onako kako ste očekivali?

- Koja funkcionalnost nedostaje prototipu?
- Možete li da pomislite na moguće uslove greške na koje se prototip ne odnosi?
- Postoje li neke nepotrebne funkcije?
- Koliko vam se čini logična i potpuna navigacija?
- Postoje li načini za pojednostavljenje bilo kojeg od zadataka koji zahtevaju previše koraka interakcije?
- Da li ste ikada bili sigurni šta dalje?

Zatražite od procenjivača da podele svoje misli naglas dok rade sa prototipom kako biste razumeli o čemu razmišljaju i mogli da otkriju sve probleme sa kojima prototip loše postupa. Stvorite okruženje u kome se ne osećaju različitost okruženja u kojem će evaluatori slobodno izraziti svoje misli, ideje i zabrinutosti. Izbegavajte da trenirate korisnike na „pravom“ putu kako bi obavljali neku funkciju sa prototipom. Dokumentirajte šta ste naučili iz evaluacije prototipa. Koristite informacije sa prototipa za obradu da biste precizirali zahteve.

Poglavlje 6

Rizici pri primeni prototipova

RIZICI OD NEUSPEHA

Rizici: direktna primena prototipa, smetnje sa detaljima, nerealno očekivanje performansi, ulaganje preteranog rada na prototipovima

Izrada čak i jednostavnog prototipa košta vreme i novac. Iako prototipiranje smanjuje rizik od neuspeha softverskog projekta, to predstavlja sopstvene rizike, od kojih su neki objašnjeni u ovoj sekciji.

Pritisak za oslobađanje prototipa

Najveći rizik je da će akteri i videti prototip tekućeg metala i zaključiti da je proizvod gotovo završen. Međutim, prototip za bacanje nikada nije namenjen proizvodnji, bez obzira koliko on izgledao kao prava stvar. To je samo model, simulacija, eksperiment. Ako ne postoji uverljiva poslovna motivacija za postizanje trenutnog prisustva na tržištu, oduprite se pritisku da vam se isporuči prototip. Isporuka ovog prototipa verovatno će odložiti završetak projekta jer su dizajn i kod namerno stvoreni bez obzira na kvalitet ili trajnost. Upravljanje očekivanjima je ključ za uspešno korišćenje prototipova. Svako ko vidi prototip mora razumeti njegovu svrhu i ograničenja. Budite jasni zašto pravite posebne vrste prototipa, odlučite kakva će biti njihova konačna sudbina i jasno o tome obavestite one zainteresovane strane koji su sa njima uključeni.

Smetanje detaljima

Još jedan rizik od prototipiranja je da se korisnici fiksiraju na pojedinosti o tome kako će izgledati i raditi korisnički interfejs. Kada radite sa prototipima stvarnog izgleda, korisnici lako zaboravljaju da bi se oni u fazi zahteva trebali baviti prevashodno konceptualnim pitanjima.

Nerealna očekivanja od performansi

Treći rizik je da će korisnici zaključiti očekivane performanse krajnjeg proizvoda prema performansama prototipa. Ipak nećete ocenjivati makete u predviđenom proizvodnom okruženju. Prototip dokaza o konceptu možda ne koristi podešene algoritme ili će mu nedostajati zaštitni slojevi koji će umanjiti krajnje performanse. Ako evaluatori vide da prototip trenutno reaguje na simulirani upit baze podataka koristeći teško kodirane rezultate uzoraka upita, mogu očekivati iste fenomenalne performanse u proizvodnom softveru sa ogromnom distribuiranom bazom podataka. Razmislite o izgradnji vremenskih kašnjenja kako biste realnije simulirali očekivano ponašanje krajnjeg proizvoda - i možda kako bi prototip izgledao još manje spreman za trenutnu isporuku..

U agilnom razvoju i drugim situacijama za razvoj prototipa, budite sigurni da od početka dizajnirate robusnu i proširivu arhitekturu i visokokvalitetni kod. Pravite proizvodni softver, samo mali deo odjednom. Dizajn možete prilagoditi refaktoringom u kasnijim ponavljanjima, ali nemojte zameniti refaktoring u budućnosti razmišljanjem o dizajnu danas.

Ulaganje preteranog napora u prototipove

Konačno, pripazite na aktivnosti izrade prototipa koje zahtevaju toliko truda da razvojnom

timu nedostaje vremena i prisiljeni su da isporuče prototip kao proizvod ili da projure kroz nesretnu implementaciju proizvoda.

Poglavlje 7

Faktori uspeha pri primeni prototipova

SMERNICE ZA USPEŠNO KORIŠĆENJE PROTOTIPIVA

Faktori uspeha: planirajte zadatke, dajte im svrhu, planirajte više prototipova, ne proveravajte ulazne podatke, bez prototopiva za zahteve koje razumete, koristite verodostojne podatke.

Prototipiranje softvera pruža snažan skup tehnika koji mogu umanjiti vremenski plan razvoja, osigurati zadovoljstvo kupaca i proizvesti proizvode visokog kvaliteta. Da biste prototipizaciju učinili efikasnim delom vašeg zahteva, sledite ove smernice:

- Uključite zadatke za izradu prototipa u svoj projektni plan. Zakažite vreme i resurse za razvoj, procenu i modifikaciju prototipa.
- Navedite svrhu svakog prototipa pre nego što ga napravite i objasnite šta će se dogoditi sa rezultatom: ili odbacite (ili arhivirajte) prototip, zadržavajući znanje koje je pružio, ili nadogradite na njemu da biste ga pretvorili u konačno rešenje.
- Uverite se da oni koji grade prototipove i oni koji ih procenjuju razumeju ove namere.
- Planirajte razvoj više prototipa. Retko ćete ih dobiti odmah pri prvom pokušaju, što je smisao izrade prototipa!
  Kreirajte prototipe za odbacivanje što brže i jeftinije. Uložite minimalni iznos napora koji će odgovoriti na pitanja ili rešiti neizvesnosti zahteva. Ne pokušavajte da usavršite prototip izbacivanja.
- Ne uključuju proveru ulaznih podataka, tehnike defanzivnog kodiranja, kod rukovanja greškama ili opsežnu dokumentaciju o kodu u prototip protokola. To je nepotrebno ulaganje napora koji ćete upravo odbaciti.

- Ne postavljajte zahteve za prototipove koje već razumete, osim da istražite alternative dizajna.
- Koristite verodostojne podatke u prototipskim ekranima i izveštajima. Procenjivači se mogu odvratiti od nerealnih podataka i ne mogu se usredsrediti na prototip kao model kako stvarni sistem može izgledati i ponašati se.
- Ne očekujte da će prototip zameniti pismene zahteve. Mnogo zakulisnih funkcionalnosti podrazumeva samo prototip i trebalo bi da ih dokumentuje u SRS-u da bi bio kompletan, specifičan i sledljiv. Slike na ekranu ne daju detalje definicija polja podataka i kriterijuma validacije, odnosa između polja (poput UI kontrola koje se pojavljuju samo ako korisnik izvrši određene odabire u drugim kontrolama), rukovanje izuzetkom, poslovna pravila i druge bitne bitove informacija .

Zamišljeno primenjeni i vešto izvedeni, prototipovi služe kao dragoceno sredstvo za pomoć u zahtevima, validaciji zahteva i škakljivom prevođenju iz potreba u rešenja

Poglavlje 8

Postavljanje prioriteta zahteva

ZAŠTO DEFINISATI PRIORITETE ZAHTEVA?

Prioritizacija je dinamičan i trajan proces. Uspostavite prioritete na početku projekta.

Svaki projekat s ograničenjima resursa mora definisati relativne prioritete traženih mogućnosti proizvoda. Prioritizacija, koja se takođe naziva trijaža potreba, pomaže u otkrivanju konkurentskih ciljeva, rešavanju sukoba, planiranju postupnih ili inkrementalnih isporuka, kontroliranju kretanja obima i donošenju neophodnih kompromisnih odluka.
Kada su očekivanja kupaca velika, a vremenski rokovi kratki, morate da se uverite da proizvod pruža najkritičniju ili najvredniju funkcionalnost što je ranije moguće. Prioritizacija je način da se reše konkurentski zahtevi za ograničenim resursima. Uspostavljanje relativnog prioriteta svake mogućnosti proizvoda omogućava vam da planirate izgradnju tako da obezbedite najveću vrednost uz najnižu cenu. Budući da je prioritizacija relativna, možete da započnete prioritizaciju čim otkrijete svoj drugi zahtev.
Ponekad kupci ne vole da daju prioritet zahtevima, misleći da nikada neće dobiti one koji su niskog prioriteta. Pa, ako nećete dobiti sve što želite, kao što je to često slučaj, trebali biste osigurati da steknete mogućnosti koje su najvažnije za postizanje poslovnih ciljeva. Ponekad programeri ne vole da daju prioritet zahtevima, jer se stvara utisak da ne mogu sve. Realnost je da ne mogu, barem ne odjednom. Prioritizacija pomaže projektu da postigne maksimalnu poslovnu vrednost što je brže moguće u okviru ograničenja projekta. Prioritizacija je kritična strategija za agilne ili druge projekte koji razvijaju proizvode kroz niz vremenskih okvira sa fiksnim rasporedom. Projektni timovi mogu popuniti zaostatke proizvoda korisničkim pričama, funkcijama, poslovnim procesima i pričama o defektima (greške koje čekaju ispravljanje).

Kupci daju prednost pričama u zaostatku i biraju koje žele da primene u svakoj razvojnoj iteraciji. Programeri procenjuju trud koji je uključen u implementaciju svake priče i procenjuju koliko ovih priča mogu da se uklope u svaku iteraciju, na osnovu njihovog empirijski pokazanog kapaciteta isporuke, mereno brzinom ekipe. Kako se predlažu nove priče, kupci procenjuju svoje prioritete u odnosu na sadržaj zaostatka i na taj način dinamički prilagođavaju domet predstojećim iteracijama. Svi projekti bi trebali to učiniti kako bi se obezbedilo da tim uvek radi na onim mogućnostima koji će što pre dobiti korisni softver u ruke korisnika.

Na svakom projektu, menadžer projekta mora da uravnoteži željeni obim projekta sa ograničenjima rasporeda, budžeta, osoblja i ciljeva kvaliteta. Jedan od načina da se to postigne je odbacivanje ili odlaganje naknadnog izdanja sa zahtevima malog prioriteta kada se prihvate novi, bitniji zahtevi ili kada se promene drugi uslovi projekta. Odnosno, prioritizacija je dinamičan i trajan proces. Ako kupci ne razlikuju svoje zahteve po važnosti i hitnosti, menadžeri projekata moraju donositi ove odluke sami. Nije iznenađujuće što se

kupci možda ne slažu sa prioritetima projektnog menadžera; stoga kupci moraju navesti koji su zahtevi u početku potrebni i koji mogu da sačekaju. Uspostavite prioritete na početku projekta, kada imate više fleksibilnosti za postizanje uspešnog rezultata projekta, i revidirajte ih povremeno.

NEKE PRAGMATIKE PRIORITIZACIJE

Razmatranja vrednosti kupca, poslovne vrednosti, poslovnog ili tehničkog rizika, troškova, teškoća implementacije, vremena do tržišta, propisa ili politike firme.

Čak i srednji projekat može imati na desetine korisničkih zahteva i stotine funkcionalnih zahteva, previše za u analitičku i doslednu klasifikaciju . Da biste ga održali, izaberite odgovarajući nivo apstrakcije za određivanje prioriteta - funkcije, slučajeve upotrebe, korisničke priče ili funkcionalne zahteve. U slučaju upotrebe, neki alternativni tokovi mogli bi imati veći prioritet od drugih. Možda ćete se odlučiti za početno određivanje prioriteta na nivou funkcija, a zatim zasebno davanje prioriteta funkcionalnim zahtevima unutar određenih funkcija. Ovo će vam pomoći da razlikujete jezgru funkcionalnosti od preciziranja koja se mogu odložiti ili u potpunosti smanjiti. Prioritetna funkcija se uključuje direktno u planiranje obima i izdanja. Ne gubite iz vida zahteve sa niskim prioritetom, mada nema smisla da ih dalje analiziramo. Njihov prioritet će se možda promeniti kasnije, a saznanje o njima sada će pomoći programerima da planiraju buduća unapređenja.

U prioritizaciji trebaju učestvovati razni akteri, koji predstavljaju kupce, sponzore projekata, upravljanje projektima, razvoj i možda druge perspektive. Zaista vam je potreban jedan konačan donositelj odluka kada se akteri ne mogu složiti. Dobra polazišna tačka je da se učesnici u određivanju prioriteta dogovore o skupu kriterijuma koji će koristiti za procenu da li jedan zahtev ima veći prioritet od drugog. Prioritizacija može uključivati razmatranja vrednosti kupca, poslovne vrednosti, poslovnog ili tehničkog rizika, troškova, teškoća implementacije, vremena do tržišta,

poštovanje propisa ili politike, konkurentske prednosti na tržištu i ugovornih obaveza. Za uspešno određivanje prioriteta potrebno razumevanje šest pitanja:

- Potrebe kupaca
- Relativni značaj zahteva za kupce
- Vreme kada su potrebne mogućnosti
- Zahtevi koji služe kao prethodnici za ostale zahteve i druge odnose između zahteva
- Koji zahtevi se moraju sprovesti kao grupa
- Trošak za ispunjavanje svakog zahteva

Kupci stavljaju veliki prioritet onim funkcijama koje pružaju najveću korist od poslovanja ili upotrebljivosti. Međutim, nakon što programer ukaže na troškove, poteškoće, tehnički rizik ili kompromise povezane sa određenim zahtevom, kupci bi mogli zaključiti da to nije toliko bitno kao što su prvo pomislili. Programer se takođe može odlučiti da ranije primenjuje određene funkcije sa nižim prioritetom zbog njihovog uticaja na arhitekturu sistema, i tako postavlja temelje za efikasnu primenu buduće funkcionalnosti bez većeg prestrukturiranja.

Neke funkcionalnosti moraju imati visoki prioritet jer su potrebne za ispunjavanje regulatornih zahteva za aplikaciju.

TEHNIKE PRIORITIZACIJE (1)

Zahtevi sa najvišim prioritetom su oni koji obezbeđuju najveći deo ukupne vrednosti proizvoda uz najmanji deo ukupnih troškova.

Na malom projektu, akteri bi trebalo da budu u mogućnosti da se neformalno dogovore o prioritetima potreba. Veliki ili sporni projekti sa mnogim akterima zahtevaju strukturiraniji pristup koji uklanja neke emocije, politiku i nagađanja iz procesa. Predloženo je nekoliko analitičkih i matematičkih tehnika koje će pomoći u postavljanju zahteva. Ove metode uključuju procenu relativne vrednosti i relativne cene svakog zahteva. Zahtevi sa najvišim prioritetom su oni koji obezbeđuju najveći deo ukupne vrednosti proizvoda uz najmanji deo ukupnih troškova.

Ulazi ili izlazi

Najjednostavnija od svih metoda prioritizacije je da grupa aktera sastavi listu zahteva i donese binarnu odluku: da li je u prioritetu ili nije? Nastavite se pozivati na poslovne ciljeve projekta da biste doneli ovu presudu, parirajući listu na najmanji minimum potreban za prvo izdanje. Zatim, kada je u toku implementacija tog izdanja, možete se vratiti na prethodno „out“ zahteve i ponovo proći kroz postupak za sledeće izdanje.

Poređenje, upoređivanje i redosled redosleda
Ljudi ponekad pokušavaju da dodele jedinstveni redni broj prioriteta svakom zahtevu. Poredak liste zahteva za poredak podrazumeva parno poređenje svih njih tako da možete proceniti koji član svakog para ima veći prioritet. Može se koristiti tabele za izvođenje poređenja atributa kvaliteta u paru; ista strategija može se primeniti na skup funkcija, korisničkih priča ili bilo koji drugi skup zahteva iste vrste. Izvođenje takvih poređenja postaje neugodno za više od nekoliko desetina zahteva. Moglo bi da radi na nivou preciznosti funkcija, ali ne i za sve funkcionalne zahteve za sistem u celini.
U stvarnosti,ređanje svih zahteva po prioritetima je nepotrebno. Nećete sve ovo implementirati u pojedinačnim izdanjima; umesto toga, grupišite ih u grupe po izdanju ili vremenskom okviru za razvoj. Dovoljno je grupisanje zahteva u karakteristike ili u male skupove zahteva koji imaju sličan prioritet ili koji se u suprotnom moraju sprovesti zajedno.

TEHNIKE PRIORITIZACIJE (2)

Kada vršite analizu prioriteta sa skalom na tri nivoa, morate biti svesni zavisnosti zahteva.

Trostepena skala
Uobičajeni pristup prioritizaciji grupiše zahteve u tri kategorije. Bez obzira kako ih označite, ako koristite tri kategorije, oni se svode na visok, srednji i nizak prioritet. Takve skale prioriteta su subjektivne i neprecizne. Da bi skala bila korisna, akteri se moraju da slože šta svaki nivo

znači u skali koju koriste.
Jedan od načina za procenu prioriteta jeste razmatranje **dve dimenzije važnosti i hitnosti.** Svaki zahtev se može smatrati ili važnim za postizanje poslovnih ciljeva ili ne tako važan, i kao hitan ili ne tako hitan. Ovo je relativna procena između niza zahteva, a ne apsolutno binarno razlikovanje. Kao što prikazuje slika 1, ove alternative daju četiri moguće kombinacije koje možete koristiti za definisanje skale prioriteta:

1. Zahtevi visokog prioriteta su i važni (kupcima je potrebna sposobnost), i hitni (kupcima je to potrebno u sledećem izdanju). Alternativno, ugovorne obaveze ili obaveze polaganja zakona mogu nalagati da se mora uključiti određeni zahtev, ili možda postoje ubedljivi poslovni razlozi za njegovo brzo sprovođenje. Ako možete da sačekate da implementirate zahtev u kasnijem izdanju bez štetnih posledica, onda nije visok prioritet po ovoj definiciji.

2. Zahtevi sa srednjim prioritetom su važni (kupcima je potrebna sposobnost), ali nisu hitni (mogu da sačekaju kasnije izdanje).

3. Zahtevi sa niskim prioritetom nisu ni važni (kupci mogu da žive bez mogućnosti ako je potrebno) niti su hitni (kupci mogu da čekaju, možda zauvek).

Izgleda da su zahtevi u četvrtom kvadrantu hitni za nekog zainteresovanog, možda iz političkih razloga, ali oni zaista **nisu važni** za postizanje poslovnih ciljeva. Ne gubite vreme na radu na tome, jer ne dodaju dovoljnu vrednost proizvodu. Ako nisu bitni, ili ih postavite na nizak prioritet ili ih u celosti očistite

Uključite prioritet svakog zahteva kao atribut zahteva u dokumentaciju o zahtevima korisnika, SRS ili bazu podataka sa zahtevima. Uspostavite konvenciju tako da čitalac zna da li prioritet dodeljen zahtevu visokog nivoa nasleđuju svi njegovi podređeni zahtevi ili da li svaki pojedinačni funkcionalni zahtev ima svoj atribut prioriteta.

Izvor: Karl Wiegers, Joy Beaty, Software Requirements, 3rd ed., Microsoft, 2013

Slika 8.1 Prioritizacija zahtvea u odnosu na važnost i hitnost

TEHNIKE PRIORITIZACIJE (3)

Akteri raspoređuju 100 USD zahtevima prema svojoj oceni prioriteta, a onda se sabiraju dolari.

MoSCoW

Četiri velika slova u shemi prioriteta MoSCoV predstavljaju četiri moguće klasifikacije prioriteta za zahteve iz skupa:

- **Mora** (Must): Uslov mora biti ispunjen da bi se rešenje smatralo uspešnim.

- **Treba** (Shoud): Zahtev je važan i treba ga uključiti u rešenje ako je moguće, ali nije obavezno da biste uspeli.
- **Može (**Could**)**: To je poželjna sposobnost, ali ona koja se može odložiti ili eliminisati. Primenite ga samo ako vreme i resursi to dozvoljavaju.
- **Neće** (Won't): Ovo ukazuje na zahtev koji se trenutno neće implementirati, ali koji bi mogao biti uključen u buduće izdanje.

MoSCoW šema menja trostepenu skalu visokog, srednjeg i niskog u skalu od četiri nivoa. Ne nudi nikakvo obrazloženje za donošenje odluke o tome kako oceniti prioritet određenog zahteva u odnosu na druge. MoSCoW je dvosmislen u pogledu vremena, posebno kada je u pitanju rejting „Neće“. „Neće“ može značiti ili „ne u narednom izdanju“ ili „nikada ne.“ Takve razlike moraju biti jasne tako da svi akteri dele zajedničko razumevanje implikacija određenog rejtinga prioriteta. Prethodno opisana skala na tri nivoa je oštriji način razmišljanja o prioritetima. Ne preporučujemo MoSCoV.

100 USD

Prioritizacija je zamišljena alokacija ograničenih resursa kako bi se postigla maksimalna korist od ulaganja koja neka organizacija ulaže u projekat. Jedan od načina da se prioritizacija učini opipljivijim je da se ona baci na stvarni resurs: novac. U ovom slučaju to je samo igranje novca, ali ipak novac.

Dajte timu za određivanje prioriteta 100 imaginarnih dolara za rad. Članovi tima dodeljuju ove dolare za „kupovinu“ predmeta koje bi želeli da primene iz kompletnog niza zahteva kandidata. Oni teže pretežno postavljaju zahteve dodeljujući im više dolara. Ako je jedan zahtev tri puta bitniji od drugog zahteva, on bi prvom zahtevu dodelio možda devet dolara, a drugom tri dolara. Ali 100 dolara dobijaju svi prioriteti - kada ostanu bez novca, ništa drugo se ne može implementirati, barem ne u izdanju na koje se trenutno fokusiraju. Jedan od pristupa je da različiti učesnici u procesu prioritizacije izvršavaju sopstvene alokacije dolara, a zatim sabiraju ukupan broj dolara dodeljen svakom zahtevu da bi se videlo koji od njih zajednički ima najveći prioritet.

Ni ova šema ne uzima u obzir zabrinutost u pogledu relativne količine napora potrebnog za sprovođenje svakog od tih zahteva. Ako biste mogli dobiti tri zahteva za svaki u vrednosti od 10 USD za isti napor kao i onaj u vrednosti od 15 USD, verovatno će vam biti bolje sa tri. Shema se zasniva isključivo na uočenoj vrednosti određenih zahteva određenog skupa zainteresovanih strana, ograničenju mnogih tehnika prioritizacije.

PRIORITIZACIJA ZASNOVANA NA VREDNOSTI, TROŠKOVIMA I RIZIKU

Ova šema se zasniva na vrednosti kupca i na naknadi koja je pružena kupcu ako je prisutna posebna karakteristika proizvoda, i na gubitka ako je ne primeni.

Kada se akteri ne mogu dogovoriti o prioritetima potreba putem drugih relativno neformalnih tehnika, možda bi bilo korisno primeniti analitičniju metodu. Definitivan, strog način povezivanja vrednosti kupca sa predloženim karakteristikama proizvoda je tehnika koja se

naziva Quality Function Deployment, ili QFD. Čini se da je nekoliko softverskih organizacija spremno preuzelo strogost QFD-a, iako se pokazalo da je korisna strukturirana metoda prioritizacije prilagođena QFD-u.

Tabela na slici 2 ilustruje model proračunske tabele koji će vam pomoći da procenite relativne prioritete za skup zahteva. Ova tehnika je svrstana u najviši nivo efikasnosti u uporednoj proceni 17 metoda prioritizacije zahteva . Primer na slici 2 prikazuje nekoliko karakteristika (šta još?) projekta Chemical Tracking System. Ova šema se zasniva na QFD konceptu zasnovanog na vrednosti kupca i na naknadi koja je pružena kupcu ako je prisutna posebna karakteristika proizvoda, i na plaćanju kazne ako je ta karakteristika odsutna. Atraktivnost karakteristike direktno je proporcionalna vrednosti koju pruža i obrnuto je proporcionalna njenom trošku i tehničkom riziku povezanom sa njegovom primenom. Ako su ostale jednake, one funkcije sa najvećim rizikom prilagođenim odnosom vrednosti i troškova treba da imaju najveći prioritet. Ovaj pristup distribuira skup procenjenih prioriteta kroz kontinuitet, umesto da ih grupiše u samo nekoliko diskretnih nivoa.

Relative weights		**2**	**1**			**1**		**0.5**		
Feature		Relative benefit	Relative penalty	Total value	Value %	Relative cost	Cost %	Relative risk	Risk %	Priority
1.	Print a material safety data sheet.	2	4	8	5.2	1	2.7	1	3.0	1.22
2.	Query status of a vendor order.	5	3	13	8.4	2	5.4	1	3.0	1.21
3.	Generate a chemical stockroom inventory report.	9	7	25	16.1	5	13.5	3	9.1	0.89
4.	See history of a specific chemical container.	5	5	15	9.7	3	8.1	2	6.1	0.87
5.	Search vendor catalogs for a specific chemical.	9	8	26	16.8	3	8.1	8	24.2	0.83
6.	Maintain a list of hazardous chemicals.	3	9	15	9.7	3	8.1	4	12.1	0.68
7.	Change a pending chemical request.	4	3	11	7.1	3	8.1	2	6.1	0.64
8.	Generate a laboratory inventory report.	6	2	14	9.0	4	10.8	3	9.1	0.59
9.	Check training database for hazardous chemical training record.	3	4	10	6.5	4	10.8	2	6.1	0.47
10.	Import chemical structures from structure drawing tools.	7	4	18	11.6	9	24.3	7	21.2	0.33
	Totals	**53**	**49**	**155**	**100.0**	**37**	**100.0**	**33**	**100.0**	

Izvor: Karl Wiegers, Joy Beaty, Software Requirements, 3rd ed., Microsoft, 2013

Slika 8.2 Matrica priotizacije zahteva u slučaju Chemical Tracking System

PRIMENA MATRICE PRIORITETA U SLUČAJU CHEMICAL TRACKING SYSTEM

Koristite model da biste razmerili relativne prioritete preostalih mogućnosti, a ne glavnih prioriteta.

Primenite ovu shemu prioriteta na diskrecione zahteve, one koji očigledno nisu glavni prioritet. Na primer, u ovu analizu ne biste uključili stavke koje implementiraju osnovne poslovne funkcije proizvoda, ključne diferencijatore proizvoda ili stavke potrebne za usaglašavanje sa propisima. Nakon što identifikujete one karakteristike koje apsolutno moraju biti uključene da bi proizvod mogao da se pušta, koristite model u tabeli na slici 2 da biste razmerili relativne prioritete preostalih mogućnosti. Tipični učesnici u procesu prioritizacije uključuju:

- Menadžer projekta ili poslovni analitičar, koji vodi proces, arbitrira sukobe i prilagođava podatke o prioritetu koji su primljeni od ostalih učesnika ako je potrebno.
- Predstavnici kupaca, poput šampiona proizvoda, menadžera proizvoda ili vlasnika proizvoda, koji isporučuju bonuse i kaznene ocene.
- Predstavnici za razvoj, koji pružaju ocene troškova i rizika.

Sledite ove korake za korišćenje ovog modela prioriteta (složenije je objasniti nego koristiti):

1. U proračunskoj tablici (Excell) navedite sve funkcije, slučajeve upotrebe, upotrebu tokova predmeta, korisničke priče ili funkcionalne zahteve kojima želite da odredite prednost jedan prema drugom. Koristili smo funkcije u primeru. Sve stavke moraju biti na istom nivou apstrakcije - ne kombinujte funkcionalne zahteve sa funkcijama, slučajevima upotrebe ili korisničkim pričama. Određene funkcije mogu biti logički povezane (implementirali biste funkciju B samo ako je funkcija A uključena) ili imaju zavisnosti (funkcija A mora biti implementirana pre funkcije B). Za ove, uključite u analizu samo pokretačke funkciju. Ovaj model će raditi sa nekoliko desetina stavki pre nego što postane neugodan. Ako imate više od toga, grupirajte stavke povezane da biste kreirali listu koju možete voditi. Metod možete primeniti hijerarhijski. Nakon što izvršite početno određivanje prioriteta, na primer, funkcija, možete ga ponovo primeniti u okviru funkcije da biste odredili prioritete njegovih pojedinačnih podfunkcija ili funkcionalnih zahteva.
2. Neka predstavnici kupca procene relativnu korist koju bi svako svojstvo pružilo klijentu ili preduzeću na skali od 1 do 9. Ocena 1 govori o tome da ga niko ne bi smatrao korisnim; 9 znači da bi bio izuzetno vredan. Ove ocene koristi ukazuju na usklađivanje funkcija sa poslovnim ciljevima proizvoda.

RAČUNANJE PRIORITETA NEKE FUNKCIJE

Proračunska tablica (matrice prioriteta) izračunava ukupnu vrednost za svaku karakteristiku kao zbir njenih koristi i kaznenih bodova

3. Procenite relativnu kaznu koju bi pretrpeo kupac ili preduzeće da svaka karakteristika nije uključena. Ponovo koristite u ocene skali od 1 do 9. Ocena 1 znači da niko neće biti uznemiren ako je odsutan; 9 ukazuje na ozbiljan nedostatak. Zahtevi i sa niskom naknadom i sa niskom kaznom dodaju troškove, ali malo vrednosti. Ponekad funkcija može imati prilično nisku vrednost, ako je neće koristiti mnogo klijenata, ali visoku kaznu ako se vaš konkurentski proizvod pohvali tom funkcijom, a kupci očekuju da je tu - čak i ako lično ne planiraju da je koriste! Ljudi koji se bave marketingom ponekad nazivaju ove

„checkbox funkcije“: treba da ih imate, čak iako je malo ljudi zaista stalo do njih. Kada dodeljujete relativnu kaznu, postavite sebi sledeća pitanja:

- Da li bi vaš proizvod bio inferioran u poređenju s drugim proizvodima koji imaju tu mogućnost?
- Da li bi bilo pravnih ili ugovornih posledica?
- Da li biste prekršili neki državni ili industrijski standard?
- Da li korisnici ne bi mogli obavljati neke potrebne ili očekivane funkcije?
- Da li bi bilo mnogo teže dodati tu sposobnost kasnije kao poboljšanje?

- Da li bi se pojavili problemi jer je marketing nekim klijentima obećao funkciju?

4. Proračunska tablica izračunava ukupnu vrednost za svaku karakteristiku kao zbir njenih koristi i kaznenih bodova (ponderirano kako je opisano kasnije u ovom poglavlju). Proračunska tablica sumira vrednosti za sve funkcije i izračunava procenat ukupne vrednosti koja dolazi od svake od karakteristika (kolona Vrednost%). Imajte na umu da ovo nije procenat ukupne vrednosti za ceo proizvod, već za skup funkcija koje ovde dajete prioritetima.

5. Neka programeri procene relativne troškove implementacije svake funkcije, opet na skali od 1 (brzo i lako) do 9 (dugotrajno i skupo). Tabela će izračunati procenat ukupnih troškova koje svaka karakteristika doprinosi. Programeri procenjuju ocene troškova na osnovu složenosti funkcije, obima potrebnog korisničkog interfejsa, potencijalne mogućnosti ponovne upotrebe postojećeg koda, potrebne količine testiranja i tako dalje. Timovi agilnih projekata razvoja mogu bazirati ove ocene troškova na broju bodova priča koji su dodeljeni svakoj korisničkoj priči. (Pogledajte poglavlje 19 u [1], „Preko razvoja zahteva“, za više o proceni agilnih projekata.)

BILANS VREDNOSTI, TROŠKOVA I RIZIKA

Karakteristike na vrhu liste imaju najpovoljniji bilans vrednosti, troškova i rizika - ako bi svi drugi faktori bili jednaki - trebali bi imati najveći prioritet.

6. Slično tome, da li programeri procenjuju relativni tehnički (a ne poslovni) rizik povezan sa svakom značajkom na skali od 1 do 9. Tehnički rizik je verovatnoća da neće dobiti ispravnu funkciju. Ocena 1 znači da ga možete programirati i u snu. Broj 9 ukazuje na ozbiljnu zabrinutost zbog izvodljivosti, nedostatka potrebne ekspertize u timu, upotrebe nepoznatih alata i tehnologija ili zabrinutost zbog veličine složenosti skrivene u zahtevu. Tabela će izračunati procenat ukupnog rizika koji dolazi od svake funkcije.
7. Nakon što ste uneli sve procene u proračunsku tabelu, izračunaće vrednost prioriteta za svaku karakteristiku koristeći sledeću formulu:
prioritet = vrednost% / (cena% + rizik%)

8. Na kraju, sortirajte listu funkcija u silaznom redosledu prema izračunatom prioriteta, krajnjem desnom kolonu. Karakteristike na vrhu liste imaju najpovoljniji bilans vrednosti, troškova i rizika, pa bi svi drugi faktori bili jednaki - trebali bi imati najveći prioritet. Rasprave koje su usredsređene na te karakteristike na vrhu liste omogućiće vam da precizirate to

preliminarno rangiranje u redosled prioriteta oko kojeg se mogu složiti akteri, čak iako svi ne dobiju upravo ono što žele.

Podrazumevano, uslovi koristi, kazne, troškova i rizika važe se podjednako. Možete da promenite relativni ponder za četiri faktora u gornjem redu tabele kako biste odražavali misaoni proces kojim vaš tim donosi prioritetne odluke. U tabeli na slici 2, sve date ocene su ponderisane dvostruko veće od odgovarajućih kazna, kazne i troškova su ponderisane isto, a rizik ima polovinu težine troškova i uslova kazne. Da biste izbacili izraz iz modela, postavite njegovu težinu na nulu.

Kada koristite ovaj model proračunske tablice sa učesnicima prioriteta, možda želite da sakrijete određene kolone koji se pojavljuju u tabeli: "Ukupna vrednost, Vrednost%, Cena% i Rizik%." Oni pokazuju posredne rezultate obračuna koji bi mogli biti samo distrakcija. Skrivanjem njih kupci će se usredsrediti na četiri kategorije rejtinga i izračunate vrednosti prioriteta.

Korisnost ovog prioritetnog modela je ograničena sposobnošću tima da proceni korist, kaznu, troškove i rizik samo za svaku stavku (funkciju). Stoga koristite izračunate prioritete samo kao smernicu. Akteri treba da pregledaju popunjenu proračunsku tablicu kako bi se usaglasili ocena i rezultat sortiranog redosleda prioriteta. Ako niste sigurni da li možete da verujete rezultatima, razmislite o kalibraciji ovog modela za sopstvenu upotrebu sa setom implementiranih zahteva iz prethodnog projekta.

TUMAČENJE DOBIJENIH RAZULTATA

Ovaj model vam takođe može pomoći da donesete kompromisne odluke kada procenjujete predložene dopune zahteva.

Prilagodite faktore ponderisanja sve dok izračunati redosled prioriteta ne bude u korelaciji s vašom procenom nakon činjenice koliko su zaista bili važni zahtevi u vašem kalibracijskom setu. Ovo će vam dati malo poverenja u upotrebu alata kao prediktivnog modela načina na koji donosite prioritetne odluke o svojim projektima.

Različiti akteri često imaju sukobljene ideje o relativnoj koristi određenog zahteva ili o kazni za njegovo izostavljanje. Proračunska tablica prioriteta uključuje varijantu koja prihvata unose iz više korisničkih klasa ili drugih grupa zainteresovanih strana. Na kartici radnog lista s više udela u tabeli za preuzimanje možete kopirati kolone relativne koristi i relativne kazne tako da imate skup za svaki faktor koji doprinosi analizi. Zatim dodelite ponder za svaki faktor , dajući veću težinu klase favoriziranih korisnika nego grupama koje imaju manji uticaj na odluke projekta. Neka svaki predstavnik aktera obezbedi sopstvenu korist i kaznene ocene za svaku karakteristiku. Proračunska tablica će uključivati pondere aktera kada izračunava konačnu ocenu vrednosti.
Ovaj model vam takođe može pomoći da donesete kompromisne odluke kada procenjujete predložene dopune zahteva. Dodajte nove zahteve u proračunsku tablicu prioriteta i pogledajte kako se njihovi prioriteti poklapaju sa onima postojeće osnovne vrednosti tako da možete odabrati odgovarajući redosled primene.
Ne morate uvek da koristite ovde iznetu metodu. Neka vaš prioritet bude što je moguće jednostavan, ali ne i jednostavniji. Nastojite da premestite prioritete sa političke i

emocionalne arene u forum u kome akteri mogu da daju iskrene procene. To će vam pružiti veće šanse za izgradnju proizvoda koji pružaju maksimalnu poslovnu vrednost uz minimalne troškove.

VIDEO 16 - PRIORITIZING REQUIREMENTS - WIEGERS (VIDEO)

Trajanje: 6:44 minuta

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

VIDEO 19 - PROTOTYPING - WIEGERS (VIDEO)

Trajanje: 7:13 minuta

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

Poglavlje 9

Vežba

PITANJA ZA DISKUSIJU

Tekst pitanja za diskusiju

PITANJE 1.

Dajte svoje mišljenje na temu u kojim je sve fazama inženjeringa zahteva koristan prototip. Da li možda smatrate da je prototip nepotreban? U svakom slučaju dajte obrazloženje za svoj stav. (10 min)

PITANJE 2.

Da li je bolje kreirati prototipove koji se bacaju nakon faze utvrđivanja zahteva ili je bolji potez praviti evolucijske prototipove? Za razvoj kojih tipova sistema biste koristili prvi model prototipa, a za razvoj kojih tipova sistema drugi model? Navedite što više konkretnih primera. (10 min)

PITANJE 3.

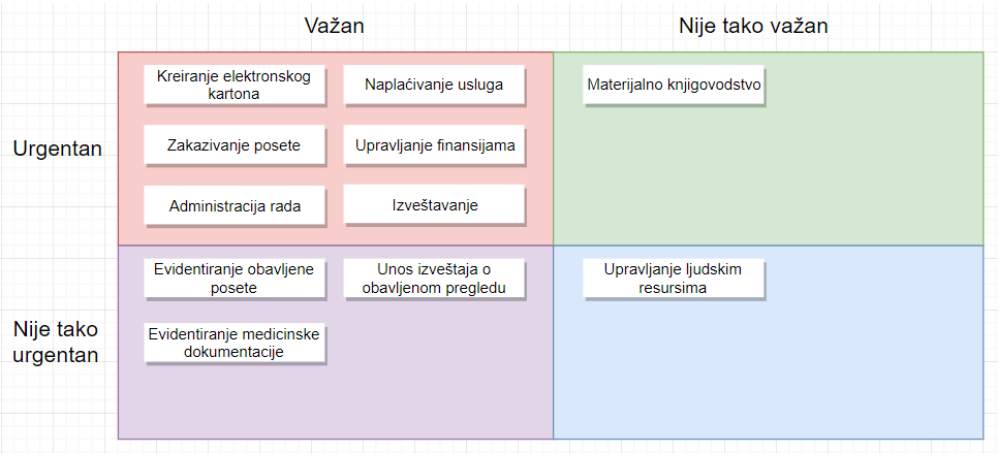
Od tehnika prioritizacije koje su objašnjene u tekstu predavanja, koju biste najradije primenjivali na svojim projektima? Uzmite za primer neki od prethodno rađenih projekata. Ukoliko ste imali priliku da radite neki softver za konkretnog klijenta, kako biste ga sada naveli da odredi prioritete zahteva? Koja tehnika bi, prema vašem mišljenju, najlakše zainteresovala klijenta da sarađuje na putu određivanja prioriteta zahteva? (10 min)

ZADATAK ZA VEŽBU 1

Tekst zadatka za vežbu 1

ZADATAK 1.

Primernom modela zasnovanog na važnosti i hitnosti, predstavljenom na slici 1 u poglavlju 8, raspoređeni su slučajevi korišćenja definisani za poslovni sistem privatne klinike (slika 1 ovog poglavlja):



Slika 9.1 Procena prioriteta zasnovana na važnosti i hitnosti [Izvor: Marina Damnjanović]

Menadžer klinike i medicinski radnici su imali najveći uticaj na proces prioritizacije zahteva. Za raspoređivanje zahteva presudnu ulogu su imale regulative medicinskih ustanova, poslovna pravila, dovođenje načina rada u red i, naravno, isplativost i zarađivanje.

Procenite da li je prioritizacija slučajeva korišćenja adekvatno urađena. Da li imate drugačiji predlog prema svom subjektivnom osećaju za prioritete? (10 min)

ZADACI ZA VEŽBU 2-3

Tekst zadataka za vežbu 2-3

ZADATAK 2.

Uzmite spisak zahteva koje ste napisali za ISUM-ov modul E-student. Koristeći definicije na slici 1 u poglavlju 8, preispitajte zahteve koje imate. Rasporedite svaki od zahteva u odgovarajuće polje, koristeći ID zahteva. Imajte na umu dve tačke gledišta: glavnog aktera – studenta i sekundarne aktere, poput predsednika i direktora Univerziteta. Da li vam to pomaže da otkrijete zahteve koji imaju veći prioritet u odnosu na one koji mogu da čekaju? Da li mislite da bi pomenute dve strane drugačije uradile prioritizaciju ovih zahteva? Kako biste pomirili dve suprotstavljene strane stejkholdera? (15 min)

ZADATAK 3.

Razvija se onlajn sistem za učenje namenjen školi stranih jezika. U razgovoru sa direktorom škole, izvedeni su sledeći funkcionalni zahtevi:
(Polaznik) Pregled kurseva koje pohađa
(Polaznik) Pregled evidentiranih poena po kursu
(Polaznik) Pregled finansija za koje je zadužen
(Predavač) Pregled kurseva na kojima predaje
(Predavač) Pregled polaznika i evidencija poena po kursu
(Finansije) Pregled svih polaznika i kurseva koje pohađaju
(Finansije) Evidencija zaduženja i unos napomena po polazniku
(Menadžment) Registrovanje korisnika i dodela privilegija

(Menadžment) Prijava korisnika na kurseve
(Menadžment) Pregled svih prijava i upravljanje prijavama

Primenom tehnike **100 dolara** uradite prioritizaciju navedenih zahteva. (20 min)

Poglavlje 10

Domaći zadatak

DOMAĆI ZADATAK 12

Tekst domaćeg zadatka

Odlučite se za jednu od tehnika određivanja prioriteta zahteva, koje su izložene u lekciji, i primenite je kako biste odredili prioritete za slučajeve korišćenja identifikovanih šestom domaćem zadatku. Ukratko obrazložite zbog čega ste rešili da primenite baš tu tehniku.

Napomene:

Zadatak se rešava opisno i šalje kao .docx fajl.
Rešenje zadatka pošaljite na mejl adresu predmetnog asistenta. Rok za izradu je definisan Plan i programom predmeta.

Poglavlje 11

Projektni zadatak

ZADATAK ZA RAD NA PROJEKTU

Tekst zadatka za rad na projektu

Proučite **Poglavlje 5 – Zahtevi za spoljni interfejs** u uzorku SRS dokumenta.

U svom SRS dokumentu, na kome radite već četiri nedelje, popunite poglavlje 5. Opišite odlike korisničkih, hardverskih, softverskih i komunikacionih interfejsa koji karakterišu vaš sistem. Ako imate vremena ili smatrate da će biti jasnije, možete kreirati prototipove i priložiti ih u delu koji se odnosi na korisnički interfejs ili kao dodatke koje ćete navesti pod **Dodatak B: Modeli analize**.

Nakon realizacije ovog zadatka, trebalo bi da imate potpun SRS dokument. Pročitajte ga pažljivo još nekoliko puta, ispravite greške ili dopunite ako je potrebno. Obratite pažnju i na suvišne delove teksta ili delove koji se ponavljaju, te njih izbacite iz dokumenta. Imajte na umu da dokumentacija treba da bude koncizna, a da istovremeno da sve neophodne informacije za dalji razvoj softvera.
Nakon što ste uradili završnu reviziju svog SRS dokumenta, pošaljite ga predmetnom asistentu na mejl, zajedno sa propratnim modelima.

Poglavlje 12

Zaključak

ZAKLJUČAK

1. Prototip softvera je delimična, moguća ili preliminarna primena predloženog novog proizvoda.
2. Svaki prototip koji napravite imaće određenu kombinaciju ovih atributa: obuhvat, buduća upotreba i oblik.
3. Maketa (mock-up) podrazumeva ponašanje bez postojanja softvera. Prikazuje fasade ekrana korisničkog interfejsa i omogućava neku navigaciju između njih, ali bez funkcionalnosti
4. Pri radu sa prototipom za izbacivanje, korisnik treba da se usredsredi na široke zahteve i probleme sa radnim tokom. Sa evolucijskim prototipom, postepeno se ide ka rešenju. Prototip koji se izbacuje podržava brzu implementaciju i modifikaciju, na račun robusnosti, pouzdanosti, performansi i dugoročne održivosti. Evolucijski prototip mora biti projektovan za lak rast i učestala unapređenja, tako da programeri moraju naglasiti arhitekturu softvera i čvrste principe projektovanja. Mogu se kombinovati primene prototipova za izbacivanje i evolucijskih prototipova.
5. Papirni prototip je jeftin, brz i nisko-tehnološki način da se istraži kako deo jednog implementiranog sistema može da izgleda. Elektronski prototipovi će vam omogućiti da lako implementirate i menjate komponente korisničkog interfejsa, bez obzira koliko je neefikasan privremeni kod koji stoji iza interfejsa.
6. Progresivni način usavršavanja jeftiniji je od skoka direktno iz opisa slučajeva korišćenja do kompletne implementacije korisničkog interfejsa. Neophodno je planiranje stranica na kojima bi se obezbedile veb stranice i zamisliti navigacione staze među njima. Ovaj iterativni pristup dizajniranju korisničkog interfejsa vodi do boljih rezultata od uranjanja odmah u dizajn stranice visoke rezolucije
7. Pri proceni vrednosti prototipova uključite članove više korisničkih klasa, iskusne i neiskusne. Naglasite da se on odnosi samo na deo funkcionalnosti.
8. Najčešći rizici od neuspeha su: direktna primena prototipa, smetnje sa detaljima, nerealno očekivanje performansi, ulaganje preteranog rada na prototipovima
9. Faktori uspeha: planirajte zadatke, dajte im svrhu, planirajte više prototipova, ne proveravajte ulazne podatke, bez prototipova za zahteve koje razumete, prototip ne zamenjuje pismene zahteve
10. Prioritizacija je dinamičan i trajan proces. Uspostavite prioritete na početku projekta. Preporuke: Razmatranja vrednosti kupca, poslovne vrednosti, poslovnog ili tehničkog rizika, troškova, teškoća implementacije, vremena do tržišta, propisa ili politike, konkurentske prednosti i ugovori.

ZAKLJUČAK (NASTAVAK)

11. Kada vršite analizu prioriteta sa skalom na tri nivoa, morate biti svesni zavisnosti zahteva. Akteri raspoređuju 100 USD zahtevima prema svojoj oceni prioriteta, a onda se sabiraju dolari.

12. Prioritizacija zasnovana na vrednosti, troškovima i riziku je najpreciznija šema za određivanje prioriteta. Zasniva se na vrednosti kupca i na naknadi koja je pružena kupcu ako je prisutna posebna karakteristika proizvoda, i na plaćanju kazne ako je ta karakteristika odsutna. Koristite model da biste razmerili relativne prioritete preostalih mogućnosti, a ne glavnih prioriteta. Proračunska tablica (matrice prioriteta) izračunava ukupnu vrednost za svaku karakteristiku kao zbir njenih koristi i kaznenih bodova. Karakteristike na vrhu liste imaju najpovoljniji bilans vrednosti, troškova i rizika - ako bi svi drugi faktori bili jednaki - trebali bi imati najveći prioritet. Ovaj model vam takođe može pomoći da donesete kompromisne odluke kada procenjujete predložene dopune zahteva.

LITERATURA

Nastavi materijal pripremljen za studente se pravi s namerom da im omogući brži i skraćeni uvid u program lekcije, a na bazi jedne ili više referentnih udžbenika i drugih izvora . Nastavni materijal nije zamena za ove udžbenike, koje treba koristiti ako student želi da se detaljnije upozna sa nastavnom materijom. Očekuje se od studenta da poseduje bar jedan od navedenih udžbenika u Planu i programu predmeta.

Ova lekcija je urađena na bazi dela teksta datom **u poglavlju 15 и 16 knjige: Karl Wiegers, Joy Beaty, Software Requirements, 3rd ed., Microsoft, 2013**. Za detaljnije proučavanje i primere, studentima se preporučuje da pročitaju ovo poglavlje. Manji uticaj na sadržaj lekcije imaju ostale reference navedene u Planu i programu predmeta,