UNIVERZITET METROPOLITAN BEOGRAD

SE101 - RAZVOJ SOFTVERA I INŽENJERA SOFTVERA

Modeli razvoja softera u uslovima stalnih promena zateva

Lekcija 03

PRIRUČNIK ZA STUDENTE

SE101 - RAZVOJ SOFTVERA I INŽENJERA SOFTVERA

Lekcija 03

MODELI RAZVOJA SOFTERA U USLOVIMA STALNIH PROMENA ZATEVA

UVOD

Saržaj lekcije

Ova lekcija se nadovezuje na prethodnu lekciju br. 2 u kojoj smo analizirali softverske procese i tri opšta modela softverskih procesa). U ovoj lekciji ćemo nastaviti sa prikazom modela softverskih procesa, te ćete se upoznati sa nekoliko novih modela procesa, ali i sa problemima razvoja softvera u uslovima stalnih promena u zatevima, što je čest slučaj u praksi: :

- Razvoj softvera u uslovim stalnih promena zahteva i uslova okruženja.
- Primena prototipova softvera
- Spiralni model razvoja softvera
- RUP (Rational Unified Process), tj. Rational ujedinjeni proces (Rational je naziv firme koja ga je razvila)
- Specijalizovani modeli softverskog procesa
- Izbor modela softverskog procesa

Poglavlje 1

Razvoj softvera u uslovima stalnih promena

KAKO RAZVIJATI SOFTVER U USLOVIMA STALNIH PROMENA?

Kako se u praksi često menjaju zahtevi u toku razvoja softvera, primenjuju se dva pristupa u razvoju: izrada prototipa softvera, radi provere zahteva.

Kod svih projekata razvoja velikih sistema, promene zahteva su neminovnost, jer se menjaju i uslovi kod naručioca sistema. Zato je vrlo važno da se, bez obzira na primenjen model softverskog procesa, model procesa menja u skladu sa promenjenim zahtevima.

Promenjeni zahtevi obično uzrokuju ponavljanje nekih aktivnosti u razvoju, što povećava troškove razvoja. Koriste se dva pristupa smanjivanja ovih troškova:

- 1. ***Izbegavanje promena***, u slučajevima kada softverski proces sadrži aktivnosti u kojima se očekuju promene, proverava se da li su promene zaista neophodne pre nego što se zahteva značajniji ponovni rad.
  - Na primer, pre nego što se pribegne promeni sistema koje zahtevaju veće troškove, razvije se poseban prototip pomoću koga se naručiocu pokazuje kako bi sistem radio, kada se izvrše njegove promene.
  - Tek kada se naručilac složi sa očekivanim rezultatom, pristupa se aktivnostima kojima se menja stvarni sistem, radi ostvarivanja usaglašenih rezultata.

- 2. **Tolerisanje promena**, neka forma inkrementalnog razvoja. Predložene promene se onda realizuju na inkrementu koji još nije razvijen. Na taj način, samo mali deo sistema (inkrement) je zahvaćen promenom, a ne i ceo sistem kada je proces tako projektovan da se promene mogu obavljati sa relativno niskim troškovima. U ovom slučaju, najčešće se primenjuje.

Imajući ovo u vidu, u praksi se primenjuju dva pristupa razvoju softvera u uslovima čestih promena zahteva:

1. ***Izrada prototipa sistema***, kojim se prototip sistema ili njegov deo brzo razvija radi provere zahteva naručioca i radi provere ostvarljivosti nekih projektnih rešenja. Promene sistema se odlažu sve dok se te promene ne verifikuju na prototipu. Na taj način se smanjuje broj zahteva za promenama posle isporuke softverskog sistema.
2. ***Inkrementalna isporuka***, kojim se naručiocu isporučuju inkrementi sistema radi komentarisanja i eksperimentisanja. To obuhvata i izbegavanje (prevremenih)

promena i toleranciju promena. Time se izbegava prerano opredeljavanje da se nešto u sistemu menja, dok se to ne proveri na inkrementu. Takođe, umesto da promene obuhvate ceo sistem, one se odnose samo na inkrement, što je znatno jeftinije.

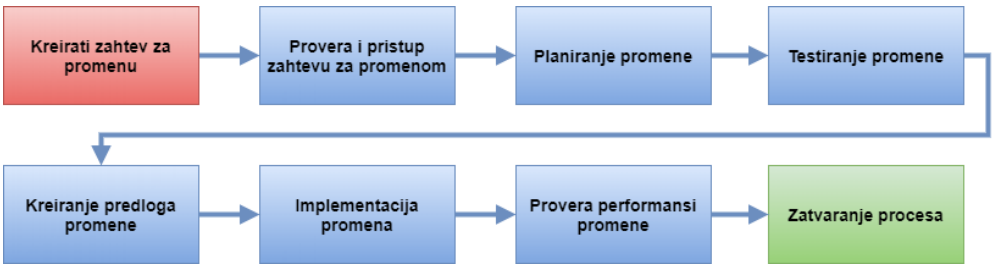
PRIMER - SISTEM ZA ONLINE GLEDANJE FILMOVA I SERIJA

Sistem za online gledanje filmova i serija.

Krajnji cilj sistema je da omogući korisniku pretraživanje i gledanje filmova i serija, ocenjivanje istih i dodavanje u listu omiljenih, kao i mogućnost korisnika da šalje zahteve administatoru sistema za postavljanje novog filma ili serije. Takođe sistem treba da obezbedi administratorima osnovne CRUD operacije nad filmovima i serijama. Sistem takođe prati najgledanije filmove/serije i postavlja ih na početnu stranicu. Pretraga se vrši po različitim kategorijama, kao što su naslov filma, godina izdavanja, žanr, rating itd.

Ako bi za ovaj sistem primenili inkrementalni model, jer nemamo precizno definisane zahteve omogućili bismo da se sistem koristi pre nego što su implementirane sve funkcionalnosti, prva faza razvoja bi trebalo da omogući CRUD funkcionalnost za filmove,serije i korisnike aplikacije, kao i pretragu i prikaz filmova i serija. Nove manje prioritetne funkcionalnosti bi se dodavale sa svakim novim inkrementom, kao što su na primer ocenjivanje filmova/serija, stavljanje u listu omiljenih, komentarisenje, praćenje popularnosti.

Ovako projektovan sistem bio bi u stanju da odgovori na promene koje nastaju usled izmene u zahtevima korisnika.



Slika 1.1 Proces upravljanja promenama [nepoznat autor]

Poglavlje 2

Primena prototipa softvera

ŠTA JE PROTOTIP SOFTVERA?

Prototip je početna verzija softverskog sistema koja se upotrebljava radi pokazivanja koncepata, probe projektnih opcija, i boljeg razumevanja problema.

Prototip je početna verzija softverskog sistema koja se upotrebljava radi pokazivanja koncepata, probe projektnih opcija, i boljeg razumevanja problema i njegovih mogućih rešenja. Brz, iterativan razvoj prototipa sa niskim troškovima, je neophodan, kako bi korisnici softverskog sistema mogli da eksperimentišu sa prototipom u ranim fazama softverskog procesa.

Softverski prototip se koristi u procesu razvoja softvera da bi se olakšala primena zahtevane promene:

1. *U procesu inženjeringa zahteva*, prototip može da pomogne da se izmame i potvrde sistemski zahtevi.
2. *U procesu projektovanja sistema*, prototip se koristi radi ispitivanja određenog softverskog rešenja i radi podrške projektovanog korisničkog interfejsa.

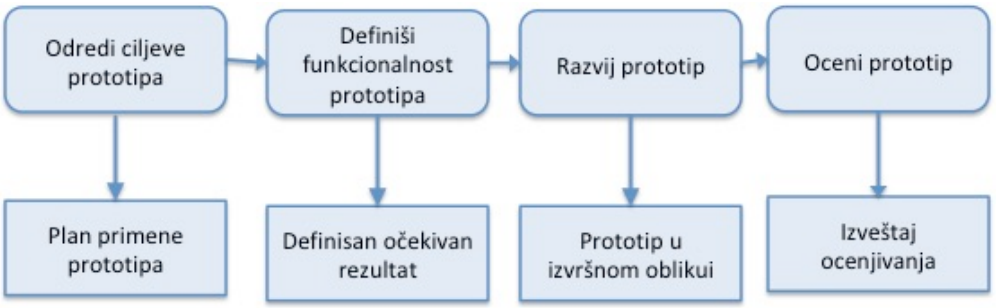
Prototipovi sistema pomažu korisnicima da vide koliko dobro sistem podržava njihov rad. Tada mogu da im se jave nove ideje za zahteve, a i da utvrde područja u kojima je softver slab ili jak. Tada oni predlažu nove zahteve. Pored toga, primena prototipova može da ukaže na greške i nedostatke u zahtevima koje su predloženi. Funkcija definisana u dokumentaciji, sama za sebe, može izgledati da je prikladna. Međutim, ista funkcija, kada se kombinuje sa drugim funkcijama, može se pokazati kao neprihvatljivom ili nekorektnom. U tom slučaju se menja specifikacija sistema kako bi bolje odražavala promene zahteva.

Prototip sistema se može koristiti u fazi projektovanja sistema radi sprovođenja eksperimenata i radi ocenjivanja ostvarljivosti predloženog projektnog rešenja. Primena brzih metoda razvoja radi provere korisničkog interfejsa, na primer, je često u upotrebi.

PROCES RAZVOJA PROTOTIPA SOFTVERA

Proces razvoja prototipa određuje aktivnosti njegovog razvoja u skladu sa postavljenim ciljem. Ključno pitanje je šta prototip treba da obuhvati, a šta ne.

Model procesa razvoja protoripa je prikazan da slici. Ciljevi razvoja prototipa bi trebalo da se jasno definišu na početku procesa. To može biti provera projektnog rešenje korisničkog interfejsa, ili validacija funkcionalnih zahteva sistema, ili razvoj sistema za proveru ostvarljivosti aplikacije. Isti prototip se može koristiti za više ciljeva. Bez jasnih ciljeva, može se postaviti pitanje namene prototipa.



Slika 2.1 Proces razvoja prototipa softvera [1.2, Fig.2.9]

Pri razvoju prototipa, ključno pitanje je šta on treba da obuhvati, a šta ne treba da bude u njemu. Radi minimazacije troškova, prototip treba osloboditi nepotrebnih funkcija. Obično se izostavljavaju i zahtevi vezani za performanse, kao što je brzina sistema i korišćenje memorije. I otklanjanje grešaka može da bude izostavljeno, ako je cilj, na primer, provera dizajna korisničkog interfejsa.

Krajnja faza procesa je evaluacija prototipa, koja se vrši na bazi postavljenih ciljeva prototipa. Vremenom, korišćenjem sistema, korisnici otkrivaju greške u zahtevima i ispuštene zahteve.

PROTOTIP NIJE SOFTVER ZA ISPORUKU

Ponekad, menadžment vrši pritisak da razvojni tim pretvori prototip u konačno rešenje, da bi se uštedelo na vremenu i novcu, što nije najčešće prihvatljivo.

Jedan od problema rada sa prototipom je što se on ne koristi isto kao i konačan sistem. Ako je prototip spor, korisnici će ga koristiti drugačije (idu skraćenicama) nego što će kasnije koristiti konačan sistem, koji je brži. To može da dovede do ispuštanja iz vida (i provere) nekih svojstava softvera u fazi ispitivanja prototipa, a koji se manifestuju pri radu konačnog sistema.

Ponekad, menadžment vrši pritisak da razvojni tim pretvori protoip u konačno rešenje, da bi se uštedelo na vremu i novcu, što nije najčešće prihvatljivo:

1. Najčešće je nemoguće podesiti prototip tako da zadovoljava nefunkcionalne zahteve, kao što su performanse, bezbednost, robusnost i pouzdanost, jer se na njih ne obraća pažnja pri razvoju prototipa.
2. Brze promene u toku razvoja, neophodno dovode do prototipa bez dokumentacije. Postoji samo kod. To nije dovoljno dobro za kasnije održavanje sistema.
3. Promene urađene na prototipu, najčešće kasnije negativno utiču na strukturu sistema. Sistem postaje težak i skup za održavanje.

4. Pri razvoju prototipa obično se ne koriste standardi kvaliteta organizacije.

Prototipovi ne moraju uvek da budu u vidu izvršnog koda. Mogu biti i samo na papiru, kao na primer, u slučaju provere dizajna korisničkog interfejsa, što znatno ubrzava proces i smanjuje cenu. I interakcija sa korisnikom, može da bude simulirana a ne stvarna, tako da odgovore korisnika prima čovek, a ne sistem.

KADA I KAKO KORISTITI PROTOTIPOVE?

Prototipovi se najčešće koriste da bi se proverili i verifikovali zahtevi ili da bise testirala tehnologija koja će se koristiti pri razvoju softvera.

Ovde ćemo rezimirano do sada izloženo. Najčešće se prototipovi koriste u dva slučaja pri razvoju softvera:

1. **Utvrđivanje zahteva korisnika:** Pokazivanjem prototipa softvera korisniku, daje mu se prilika da potvrdi da li je to što očekuje ili da dopuni svoje zahteve. Tada se često utvrđuju suprostavljeni i nejasni zahtevi.
2. **Procenjivanje tehničkih i arhitektonskih rizika:** Pri projektovanju softvera se najčešće koriste projektni šabloni. Međutim, kod složenih softvera njihova primena nije uvek moguća. Primena prototipa je neophodna da bi se testirala i postavljena arhitektura softverskog sistema. U slučaju primene novih tehnologija, na ovaj način se proverava njena upotrebljivost u slučaju softvera koji se razvija.

Pri razvoju softvera, prototipovi se koriste u dva oblika kao:

a) model procesa i kao

b) tehničko rešenje, tj. tehnologija.

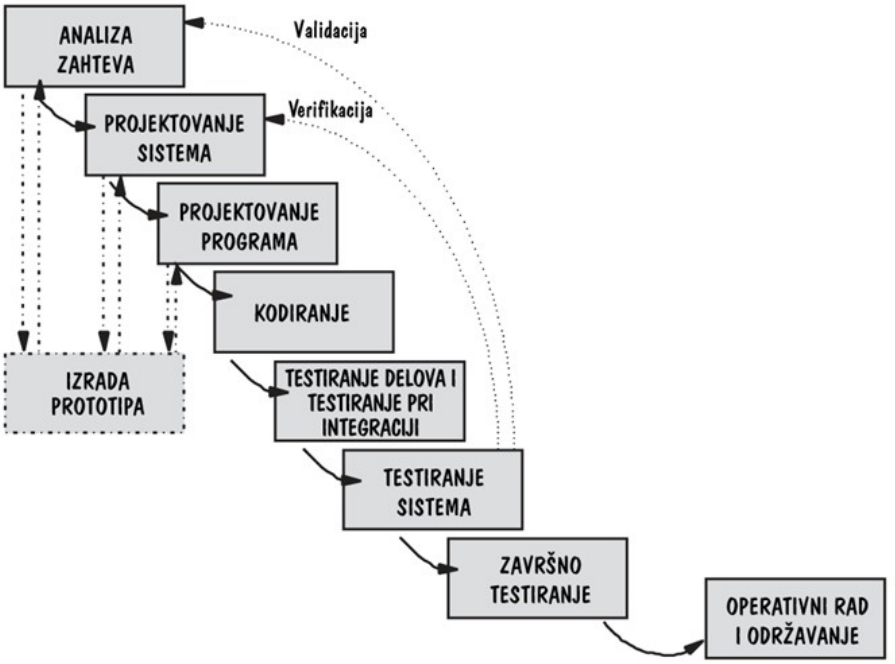
Ako se prototip koristi radi modelovanja procesa, onda se za svaku aktivnost procesa razvoja softvera koriste prototipovi, kako bi se proverilo da li se dobija ono što se želi. Kada se to dobije, prelazi se na sledeću aktivnost, itd. Na kraju, posle poslednje aktivnosti, vrši se isporuka softvera razvijenog na ovakav način.

Ako se, pak, prototip koristi da bi testirala primenjena tehnologija, onda se on koristi unutar nekog od modela procesa. RUP i spiralni model razvoja koriste prototipove unutar svog modela procesa razvoja softvera. Kada se prototipom izvrši provera tehnologije, on se odbacije, i za razvoj softvera se ne koristi više prototip, već se razvija softver za korisnika.

Prototipovi se retko koriste kao modeli procesa, jer su retki slučajevi da su zahtevi potpuno nepoznati ili da su postavljenje arhitekture toliko rizične i nepoznate da ih je potrebno proveriti. Mnogo ćešće se prototipovi koriste radi provere zahteva i tehnologija.

PRIMENA PROTOTIPOVA U FAZAMA RAZVOJA SOFTVERA

Primena prototipa softvera



Slika 2.2 Razvoj i primena prototipova softvera u raynim fazama njegovog razvoja [nepoznat autor]

Kreiranje prototipova za sistem podrazumeva:

Prototip za bacanje (Throwaway Prototyping)

- podrazumeva kreiranje prototipa koji će u dogledno vreme biti odbačen
- nakon preliminarne faze zahteva, konstruiše se jednostavniji radni model sistema radi ilustracije korisniku
- izrada radnog modela različitih delova sistema u ranoj fazi razvoja
- može da se uradi veoma brzo
- testira se i User Interface

Evoutivan prototip (Evolutionary Prototyping)

- izgradnja vrlo robusnog prototipa na strukturiran način i njegovo konstantno usavršavanje
- gradi se samo na osnovu onih zahteva koji su dobro razmotreni i prihvaćeni
- dodavanje novih funkcionalnosti, evaluacija kroz različita operativna okruženja

EVOLOUCIJA PROTOTIPOVA (VIDEO)

Primena prototipa softvera - Video klip 1

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

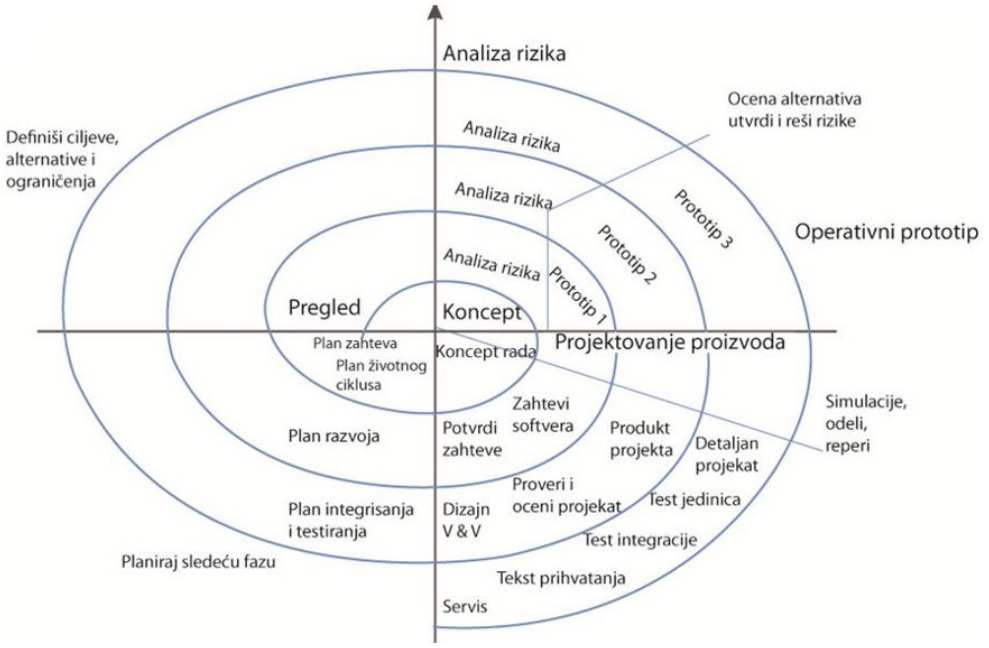
Poglavlje 3

Spiralni model razvoja

ŠTA JE SPIRALNI MODEL RAZVOJA?

Softverski proces je predstavljen kao spirala pre nego sekvenca aktivnosti. Svaka petlja u spiralnom modelu predstavlja fazu procesa.

Spiralni model softverskog procesa (prikazan na slici 1) prvobitno je bio predložen od Boehm-a (1998) i sada je naširoko poznat. Proces je predstavljen kao spirala pre nego sekvenca aktivnosti sa povratnom spregom*. Svaka petlja u spiralnom modelu predstavlja fazu procesa*. Prema tome, krajnja unutrašnja petlja može se odnositi na izvodljivost sistema, sledeća petlja na definiciju zahteva sistema, sledeća petlja na projektovanje sistema, itd



Slika 3.1 Spiralni model rarzvoja procesa 1.[3. Fig. 2.11]

ČETIRI SEKCIJE PETLJE U SPIRALI MODELI

Važna razlika između spiralnog modela i drugih modela softverskih procesa je eksplicitno razmatranje rizika u spiralnom modelu.

Svaka petlja u spirali je podeljena u četiri sekcije kao što je prikazano na slici 3.7. Sekcije spiralnog modela su:

1. **Definisanje ciljeva** – Definisani su specifični ciljevi za ovu fazu projekta. Identifikovana su ograničenja procesa i proizvoda i skiciran je detaljni plan upravljanja. Identifikovani su rizici. Mogu biti planirane alternativne strategije zavisno od rizika.
2. **Određivanje i redukcija rizika** - Za svaki od identifikovanih projekata rizika, izvedena je detaljna analiza. Da bi redukovali rizik uvedeni su koraci. Na primer, ako postoji rizik za koji su zahtevi nezadovoljavajući, može biti razvijen prototip sistema.
3. **Razvoj i validacija** – Nakon proračuna rizika, izabran je razvojni model za sistem. Na primer, ako su rizici korisničkog interfejsa dominantni, odgovarajući razvojni model može biti evolucionarni prototip. Ako su rizici sigurnosti dominantni, može biti pogodan razvoj baziran na formalnoj transformaciji, itd. Model vodopada može biti najpogodniji razvojni model ako je glavni identifikovani rizik integracija pod-sistema.
4. **Planiranje** – Projekat je razmotren i doneta je odluka da li nastaviti sa narednom petljom spirale. Ako je doneta odluka da se nastavi, crtaju se planovi za narednu fazu projekta.

Važna razlika između spiralnog modela i drugih modela softverskih procesa je eksplicitno razmatranje rizika u spiralnom modelu. Neformalno, rizik je jednostavno nešto što može da krene pogrešno. Na primer, ako je intencija da koristimo nove programske jezike, rizik je da su postojeći kompajleri nepouzdani ili da ne proizvode dovoljno efikasan objektni kod. Rezultat rizika u projektnim problemima kao što su raspored i prekoračenje troškova je vrlo važna aktivnost upravljanja projektom.

Ciklus spirale počinje sa razradom ciljeva kao što su performanse, funkcionalnost, itd. Zatim su nabrojani alternativni načini dostizanja ovih ciljeva i ograničenja zadata od strane svake alternative. Svaka alternativa je ocenjena nasuprot svakom cilju. Ovo obično rezultira identifikacijom izvora rizika projekta. Sledeći korak je proračunavanje ovih rizika preko aktivnosti kao što su detaljna analiza, simulacija, itd. Jednom kada su rizici ocenjeni, izvršen je razvoj i ovo je praćeno planiranjem aktivnosti za narednu fazu procesa.

SVOJSTVA SPIRALNOG MODELA

Spiralni model je pogodan za velike i složene projekte, ali je skup i složen za korišćenje.

Primena spiralnog modela je odgovarajuća za slučaj velikih i složenih projekata, koje prate veliki razici, jer se u spiralnom modelu posebna pažnja poklanja analizi rizika.

Dobro svojstvo spiralnog modela je što *omogućava ikrementalnu isporuku* softvera i što u sebi *uključuje i tehnike korišćenja prototipova*, kako bi se smanjili rizici pri razvoju softvera. Ustvari, spiralni model podržava korišćenje i inkrementalnog i sekvencijalnog modela razvoja, kao i upotrebu modela sa prototipovima. Ova kombinacija pristupa, čini ovaj model vrlo moćnim.

Međutim*, složenost modela mu je istovremeno i slabost*, jer je složen za upravljanje i njegovu primenu prate visoki troškovi korišćenja, pre svega zbog analiza rizika i korišćenja prototipova u svakoj petlji. Zato, nije pogodan za primenu u malim i srednjim projektima razvoja softvera ili u slučajevima gde je agilnost procesa vrlo bitna.

SPIRALNI MODEL SOFTVERSKOG PROCESA (VIDEO)

Spiralni model razvoja - Video klip 1

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

Poglavlje 4

Rational ujedinjeni proces (RUP)

RUP MODEL

RUP model se bavi analizom rizika i podržava razvoj koji primenjuje slučajeve korišćenja. Posebnu pažnju posvećuje arhitekturi softvera.

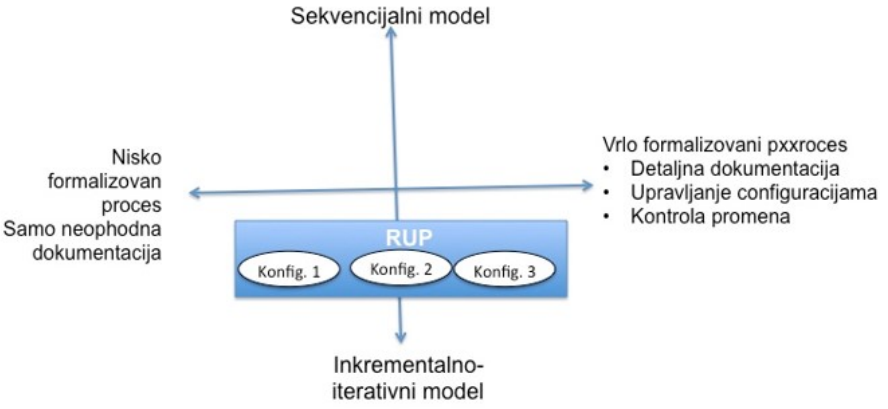
Rational Unified Process ili RUP je model procesa koji se bazira na inkrementalnom i iterativnom proces razvoja softvera. RUP model se bavi analizom rizika i podržava razvoj koji primenjuje slučajeve korišćenja. Posebnu pažnju posvećuje arhitekturi softvera i ona je u centru njegove pažnje.

RUP ie takođe i okvir za modeliranje procesa. Omogućuje prilagođavanje procesa potrebama korisnika (process customization) i omogućava definisanje procesa. Na taj način, omogućava dobijanje različitih konfiguracija procesa. Te specifične konfiguracije omogućavaju:

- Podršku razvojnim timovima različite veličine (male, srednje i velike)
- Primenu vrlo formalizovanog ili malo formalizovanog procesa razvoja softvera, tj. metoda razvoja softvera.

Da bi jasnije pozicionirali RUP u odnosu na bitne parametre koje klasifikuje različite pristupe u postavljanju procesa razoja softvera, koristićemo dve ose sa po dve ekstremne vrednosti tih parametara (slika 1).

Na vertikalnoj osi prikazan je metod isporuke razvijenog softvera. Na vrhu je ekstremni slučaj primene metoda vodopada, a na dnu je drugi ekstrem – primena inkrementalnog i iterativnog načina isporuke, tj. razvoja softvera. Na horizontalnoj osi se prikazuje mera formalizacecije procesa razvoja softvera (novo proizvedene i korišćene dokumenstacije, upravljanja konfiguracijama i kontrole promena u softveru. Na desnoj strani se daje najveći stepen korišćenja projektne i druge dokumentacije, metoda konfigurisanja softvera i kontrole promena u softveru. Na drugom, levom kraju je dat drugi ekstremni slučaj: vrlo malo dokumentacije i velika fleksibilnost u definisanju procesa razvoja.



Slika 4.1 Konfiguracije RUP-a [1.3]

FAZE SOFTVERSKOG PROCESA U RUP MODELU

RUP model ima četiri faze softverskog procesa u kome su ove faze povezane sa aktivnostima, koje su su povezane u skladu sa poslom, a ne tehnikom.

Rational ujedinjeni proces (engl. The Rational Unified Process – RUP) je primer modernog modela procesa koji primenjuje UML (engl. Unified Modelling Language) za modeliranje procesa. Kao model hibridnog procesa, on sadrži elemente sadržane u svim opštim modelima procesa, i predstavlja dobru praksu pripreme specifikacije i projektnog rešenja softvera, kao i podrške primene prototipova i inkrementalne isporuke softvera.

RUP opisuje proces iz **tri perspektive:**

1. Dinamička perspektiva – pokazuje faze model tokom vremena.
2. Statička perspektiva – prikazuje aktivnosti procesa
3. Perspektiva prakse – sugeriše dobre prakse upotrebe modela

Najčešće se primenjuje kombinovanje statičke i dinamičke perspektive u vidu jednoh jedinstvenog dijagrama.

RUP model ima **četiri faze softverskog procesa**. Za razliku od modela vodopada, u kome su ove faze povezane sa aktivnostima, kod RUP modela, ove faze su povezane u skladu sa poslom, a ne tehnikom (slika). Naredna faza počinje kada se prethodna završi, ali u okviru svake faze postije više iteracija. Te faze su:

1. **Početak**: Cilj početne faze je postavljanje poslovnog scenarija (slučaja) sistema, tj. određivanja šta sistem treba da radi. Utvrđuju se spoljni akteri (ljudi i sistemi) koji su interakciji sa sistemom i definišu se te interakcije. *Vrši se procena efekata rada sistema na poslovanje i procena rizika*. Ako su efekti mali, može da dođe i do zaustavljanja projekta.
2. **Razrađivanje** (elaboracija): Cilj ove faze je razvoj razumevanja domena problema, postavljanje arhitektonskog okvira sistema (koncepcijsko rešenje arhitekture), razvoj plana projekta, i utvrđivanje ključnih rizika projekta. Na kraju ove faze, dobija *se*

model zahteva sistema u vidu skupa UML slučajeva primene, opis arhitekture i razvijen plan projekta.

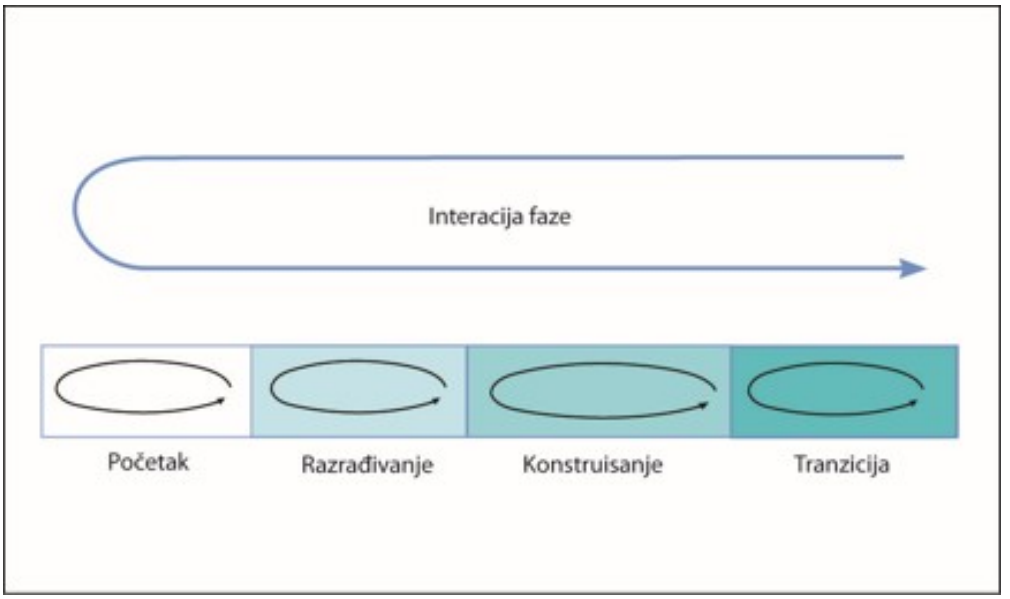
3. **Konstrukcija**: Faza konstrukcije obuhvata projektovanje sistema, programiranje i testiranje. Delovi sistema se paralelno razvijaju u ovoj fazi. Na kraju ove faze, *sistem je u radnom stanju*, zajedno sa pratećom dokumentacijom, te je *spreman za isporuku korisnicima.*

4. **Tranzicija**: Konačna faza RUP-a se bavi prenosom sistema iz razvojnog okruženja u korisničko okruženje, i stavlja ga u rad u stvarnom okruženju. Na kraju ove faze, *sistem je opremljen kompletnom softverskom dokumentacijom i ispravno radi u operativnom (radnom) okruženju*.

RADNI TOKOVI U RUP MODELU

Statički pogled na RUP se fokusira na aktivnosti razvojnog procesa. One se nazivaju i radnim tokovima.

Iteracije u RUP-u se javljaju unutar svake faze, ali i na nivou celog procesa (vraćanje na prethodne faze).



Slika 4.2 Iteracije u RUP modelu [1.3, Fig. 2.12}]

RADNI TOKOVI U RUP-U

Radni tokovi su statički pogledi na RUP

Radni tok	Opis
Modelovanje poslovanja	Poslovni procesi se modeluju primenom UML slučajeve upotrebe (UML use cases)
Zahtevi	Utvrđeni su akteri koji su interakciji sa sistemom i slučajevi upotrebe su razvijeni radi modelovanja zahteva sistema.
Analiza i projektovanja	Kreira se i dokumentuje model projektovanja upotrebom modela arhitektura, modela komponenti, objektnih modela i sekvencnih modela
Implementacija	Komponente sistema su primenjene i strukturisane u podsistemima. Automatska generacija koda iz projektnog modela ubrzavaju proces.
Testiranje	Testiranje je iterativni proces koji se sprovodi zajedno sa implementacijom. Testiranje sistema se vrši posle završetka implementacije.
Instaliranje	Napravljena je konačna verzija proizvoda, podeljena je korisnicima i instalirana na mestima upotrebe.
Konfigurisanje i upravljanje promenama	Ovaj radni tok upravlja promenama sistema.
Upravljanje projektima	Ovaj tok rada upravlja razvojem sistema.
Okruženje	Ovaj tok rada omogućava da tim za razvoj softvera koristi odgovarajućie softverske alate.

Slika 4.3 Radni tokovi u RUP-u [1.3, Fig. 2.15]

Statički pogled na RUP se fokusira na aktivnosti razvojnog procesa. One se nazivaju i radnim tokovima (eng. workflows). Postiji šest ključnih radnih tokova procesa i tri radna toka podrške (videti Tabelu 1). Kako je RUP projektovan primenom UML, to je i opis ovih radnih tokova izvršen u vidu odgovarajućih UML modela.

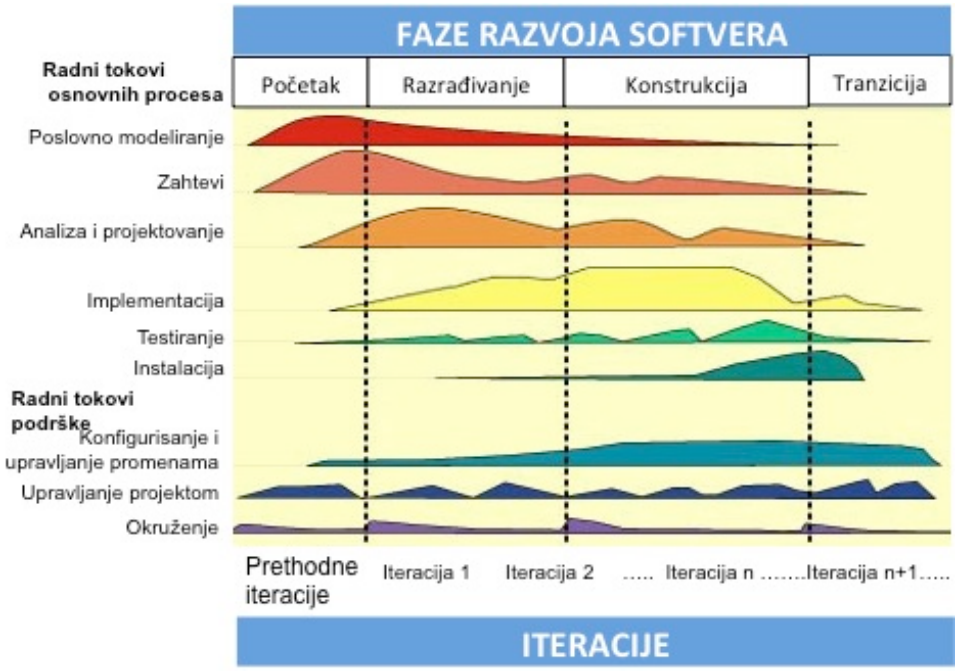
AKTIVNOSTI RADNIH TOKOVA PROCESA RAZVOJA

Aktivnosti radnih tokova procesa razvoja, a i procesa podrške, prisutne se u svim fazama razvoja softvera i izvršavaju se sa različitim intenzitetom.

Na slici 4 su prikazani radni tokovi osnovnih procesa i radni tokovi podrške tokom četiri faze razvoja softvera. Radne tokove osnovnih procesa čini šest radnih tokova, a tri radna toka čine radne tokove podrške. Aktivnosti ovih devet radnih tokova se realizuju u svim fazama, sa različitim intenzitetom. Na primer, u početnoj fazi, najintenizivniji je rad na poslovnom modelovanju (slučajevi upotrebe, scenariji) i na definisanju zahteva. U fazi razrađivanja, naintenzivnije aktivnosti su na detaljnijoj specifikaciji zahteva i na analizi sistema i projektovanju. U fazi konstrukcije softvera, najintenzivniji je rad na implementaciji projektnog rešenja, tj. na programiranju u testiranju. U fazi tranzicije najviše se radi na završnom i korisničkom testiranju.

Aktivnosti radnih tokova podrške se odvijaju tokom celog procesa različitim intenzitetom.

U svima fazama razvoja softvera, kreiraju, se primenom više iteracija, različite verziije softverskih jedinica i konfiguracije celog softverskog sistema.



Slika 4.4 Radni tokovi u RUP modelu [3.1]

NAJBOLJA PRAKSA U KORIŠĆENJU RADNIH TOKOVA RUP-A

Najveća inovacija RUP-a je odvajanje faza i radnih tokova, kao i prepoznavanje raspoređivanja (instalacije) softvera u radno okruženje korisnika.

U principu, svi radni tokovi mogu biti aktivni u svim fazama procesa razvoja. Neki su aktivniji u početnim fazama, neki u završnim fazama. Preporučuje se šest osnovnih najboljih praksi:

1. *Iterativni razvoj softvera:* Planirati inkremente sistema na osnovu prioriteta kupca.
2. *Uređivanje zahteva*: Jasno definisati zahteve kupaca i beležite promene tih zahteva. Analizirati posledice usvajanja novih zahteva na sistem, pre nego što se prihvate.
3. *Upotrebiti arhitektura baziranu na komponentama*: Strukturisati arhitekturu sa komponentama.
4. *Vizuelno modelovati softver*: Upotrebiti UML modele za predstavljanje statičkih i dinamičkih pogleda na softver.
5. *Proveriti kvalitet softvera:* Obezbediti proveru da li softver zadovoljava standarde kvaliteta organizacije.
6. *Kontrolisati promene u softveru:* Uređivati promene u softveru upotrebom sistema za upravljanje promena sistema i alate i procedure upravljanja konfiguracijom.

RUP nije pogodan za sve sisteme, kao što su na primer, ugrađeni sistemi. Najveća inovacija RUP-a je odvajanje faza i radnih tokova, kao i prepoznavanje raspoređivanja (instalacije) softvera u radno okruženje korisnika kao deo procesa. Faze su dinamičke i imaju svoje ciljeve. Radni tokovi su statički i predstavljaju tehničke aktivnosti koje nisu povezane sa

pojedinačnom fazom, već se mogu upotrebiti za vreme celog razvoja radi ostvarivanja ciljeva svake faze.

RUP je okvir koji vam omogućava da izaberete model procesa koji najviše odgovara specifičnostima softvera koji razvijate, jer možete birati stepen formalizma procesa.

RATIONAL UNIFIED PROCESS - RUP (VIDEO)

RUP - Video klip 1

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

VIDEO PREDAVANJE ZA OBJEKAT "RATIONAL UJEDINJENI PROCES (RUP)"

Trajanje video snimka: 28min 8sek

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

Poglavlje 5

Specijalizovani modeli

ŠTA SU SPECIJALIZOVANI MODELI?

Specijalizovani modeli neke specifičnosti vezane za upotrebu specijalnih tehnika i metoda kojim se rešavaju njihovi specifični problemi.

Specijalizovani modeli se razvijaju za specifične vrste projekte razvoja softvera, tj. za specifične vrste softvera. Specijalizovani modeli imaju karakteristike procesno-orijentisanih modela za razvoj softvera, ali imaju i neke specifičnosti vezane za upotrebu specijalnih tehnika i metoda kojim se rešavaju njihovi specifični problemi.

Zavisno od njihovih specifičnosti, specijalizovani model se mogu klasifikovati u nekoliko grupa, kao na primer:

1. Modeli sa komponentama (videti lekciju 2)
2. Formalni metodi
3. Razvoj u skladu sa aspektom korišćenja softvera

FORMALNI METODI

Formalni metodi koriste formalne matematiče specifikacije za definisanje zahteva i omogućavaju automatsko projektovanje, generisanje koda i testiranja

Formalni metodi upotrebljavaju formalne matematičke specifikacije. Zahtevi se definišu matematičkim jednačinama sa precizno definianim rečnikom, sintaksom i semantikom. Verifikacija (provera) zahteva, projektovanje softvera i generisanje koda se vrši potpuno automatski. Takođe, i sprovođenje testova je takođe automatizovano.

Dobro svojstvo formalnih metoda je što su nemogući dvosmisleni zahtevi, kao što je to moguće kod neformalnih metoda.

Formalni metodi su pogodni za razvoj softvera za rad u realnom vremenu, koji obično imaju kritične ciljeve (misije), kao što su to sistemi koji rade u kontrolama letenja i dr. Međutim, primena formalnih metoda zahteva puno vremena, te zahtevaju velike troškove u primeni. Takođe, zahteva od programera puno specifičnog znanja, koje obično nedostaje. Iz ovih razloga, retko se koriste.

ASPEKTNO ORIJENTISANI RAZVOJ SOFTVERA

Aspketno orijentisani razvoj obezbeđuje metodologiju za secificiranje, projektovanje, i razvoj ovih aspekata (Aspects).

Postoje softverski sistemi koji zahteva neku specifičnu funkciju, ili komponentu, ili uslugu. O tome se mora posebno voditi računa prilikom razvoja ovakvog softvera. Mora se voditi računa o tom *aspektu* softverskog sistema. Te specifičnosti su najčešće nefunkcionalni zahtevi vezani za bezbednost, keširanje ili upravljanje transakcijama. Aspekno orijentisani razvoj definiše ove specifičnosti kao aspekte –Aspects. Aspketsko orijentisani razvoj obezbeđuje metodologiju za specificiranje, projektovanje, i razvoj ovih aspekata (Aspects). Kako su oni izolovani, tj. posebno razvijani i višestruko korišćeni, održavani, mogu se uklapati u sistem .
Aspektno orijentisani razvoj softvera (AOSD- Aspect-Oriented Software Development) je rešenje za projektovanje softvera koje pomaže u rešavanju problema modularnosti koji nisu pravilno rešeni drugim softverskim pristupima, kao što je proceduralno, strukturirano i objektno orijentisano programiranje (OOP). AOSD dopunjuje, a ne zamenjuje, ove druge tipove softverskih pristupa.
AOSD je takođe poznat kao aspektno orijentisano programiranje (AOP).

Aspektno orijentisani razvoj softvera (AOSD) ima sledća svojstva:

- Smatra se podskupom tehnologija post-objektnog programiranja
- Bolja podrška projektovanu softvera kroz izolovanje poslovne logike aplikacije od pratećih i sekundarnih funkcija
- Pruža komplementarne prednosti i može se koristiti sa drugim agilnim procesima i standardima kodiranja
- Ključni fokus - Identifikacija, predstavljanje i specifikacija zabrinutosti, koja takođe može biti međusektors
- Pruža bolju podršku modularizaciji softverskog dizajna, smanjujući troškove dizajna, razvoja i održavanja softvera
- Princip modularizacije zasnovan na uključenim funkcionalnostima i procesima
- Budući da su zabrinutosti inkapsulirane u različite module, bolje se promoviše i obrađuje lokalizacija sveobuhvatnih briga
- Pruža alate i tehnike softverskog kodiranja kako bi se osigurala podrška modularnog sadržaja na nivou izvornog koda
- Promoviše ponovnu upotrebu koda koji se koristi za modularizaciju sveobuhvatnih pitanja
- Manja veličina koda, zbog rešavanja problema unakrsnog preseka
- Smanjena efikasnost zbog povećanih troškova

Poglavlje 6

Izbor modela procesa razvoja softvera

KRITERIJUMI IZBORA MODELA PROCESA

Izbor modela procesa zavisi od učestalosti promena zahteva, složenosti arhitekture softvera i veličine projekta.

Da bi se razvio dobar softver, potrebno je primeniti odgovarajući proces razvoja softvera. Postoji više različitih modela procesa razvoja softvara. Koji izabrati?

Ne postoji jedno univerzalno rešenje, tj. jedan proces koji je optimalan za sve vrste softvera. Na softver inženjerima je zadatak da na osnovu analize specifičnosti softvera koji treba da razviju, i poznavanja specifičnosti svih poznatih modela procesa, izaberu jedan koji najviše odgovara softveru koji treba da razviju. Taj proces treba da obezbedi da projektni tim dobije:

- odgovarajući softver, tj. softver koji zadovoljava potrebe kupca, tj. korisnika,
- koji ima potreban nivo kvaliteta,
- koji dobija u okviru planiranog budžeta i u planiranom roku.

Svaki model procesa je odgovarajući za određene scenarije i vrste softvera. Da bi organizacija koja se bavi razvojem softvera mogla da izabere pravi proces, potrebno je da definiše svoje kriterijume za izbor modela procesa koji najviše odgovara tipu softvera koji treba da se razvije.

Na osnovu iskustva, uspostavljene su izvesne preporuke za izbor proces razvoja softvera. Jedan od jednostavnijih metoda izboora modela procesa uzima u obzir tri faktora:

1. Brzinu promena zahteva
2. Složenost arhitekture softvera
3. Veličina projekta izražena u broju potrebnih čovek-dana, veličini tima).

Na slici 1 je prikazan primer postavljenih kriterijuma za izbor modela procesa u jednoj organizaciji.

Brzina promene zahteva	Složenost arhitekture	Veličina projekta	Preporučeni model procesa
Velika	visoka/srednja	srednji/veliki	RUP sa nisko formalizovanim procesom
Mala	visoka/srednja	srednji/veliki	RUP sa srednje formalizovanim procesom
Velika	visoka/srednja	mali	Agilni modle
Velika	Niska	mali/srednji	Agilni model
Velika	Nisoka	veliki	RUP sa nisko formalizovanim procesom

Slika 6.1 Primer kriterijuma za izbor modela procesa.[1.1.2]

IZBOR MODELA PROCESA I BUDUĆI SOFTVERSKI MODULI

Koncept razvoja budućih softverskih modula ostaje isti, bez obzira od izabranog modela procesa razvoja.

Postavlja se pitanje: Da li izbor modela procesa razvoja softvera utiče na module softvera koje ćemo u budućnosti razvijati? Da bi odgovorili na ovo pitanje, najpre ćemo naglasiti da se opšte aktivnosti razvoja softvera (specifikacija zahteva, projektovanje softvera, programiranje, testiranje, instalacija i evolucija) postoje u svim modelima procesa. Modeli procesa ove aktivnosti realizuju na svoj, specifični način, ali one postoje. Modeli procesa određuju tok aktivnosti, organizaciju razvojnog tima, uloge i odgovornosti njegovih članova, i ganularnost (veličinu) softverskog dela koji se razvija. Iz ovde iznetog, može se zaključiti da su budući softverski moduli (i primenjene aktivnosti razvoja i metodi) nezavisni od izabranog modela procesa. Mogu se razlikovati redosledi izvršenja razvojnih aktivnosti, njihov tok, odgovornosti i uloge članova tima, kao i veličina softverske jedinice koja se razvija, ali se sam koncept razvoja softvera ne menja.

Na primer, aktivnost *Specifikacija zahteva*, uvek je prisutna. U sekvencijalnom modelu vodopada, ona se u celosti završi pre nego što se pređe u sledeću fazu razvoja. U slučaju RUP modela, specifikacija zahteva se odvija u svim fazama. Slično je sa aktivnošću projektovanja softvera i sa drugim.

U slučaku agilnih modela procesa, tj. agilnih metoda razvoja, u svakoj iteraciji se primenjuju sve glavne aktivnosti razvoja softvera (specifikacija zahteva, projektovanje softvera, programiranje i testiranje), samo što je uklonjena oštra granica među njima, te se one ne odvijaju određenim redosledom. Projektni tim iterativno radi na razvoju jednog inkrementa, bez posebnog naglaska na neku aktivnost razvoja softvera i bez posebno definisanog redolsleda odvijanja tih aktivnosti. Umesto linearnog, primenjuju nelinearni model odvijanja aktivnosti procesa, tako da u uskoj međusobnoj kolaboraciji, obavljaju neophodne razvojne aktivnosti, manje-više simultano.

Znači, kocept razvoja softvera baziran na navedenim osnovnim aktivnostima ostaje isti, samo se menjaju načini primene tih aktivnosti. Izbor modela procesa utiče na tok aktivnosti i na poseban naglasak na neke od njih.

PREGLED PREPORUKA ZA UPOTREBU UOBIČAJENIH MODELA PROCESA

Model procesa se bira u odnosi na vrstu softvera koji se razvija. Međutim, osnovne aktivnosti razvoja su u svima njima prisutne, samo na različite načine.

U sledećoj tabeli, daje se pregled osnovnih svojstava i preporuka za upotreba uobičajenih modela procesa.

Model procesa	Osnovna svojstva i preporuke za primenu
Model vodopada	Ststematski i sekvencijalni prist procesu. Nije dobar kada se zahtevi često menjaju. I kada kupac želi da učestvuje u razvoju. Retko se koristi jer se traže više evolutivni modeli procesa razvoja softvera.
Interativni/ inkrementalni	Pruža osnovu za evotutivne modele procese (npr. Agilni, RUP). Obezbeđuje dodavanje novih inkremenata, koji se iterativno poboljšavaju.
Prototipovi	Koriste se da bi se testirao izabrani koncept ili proces. Retko s ekoristi kao za modelovanje celog procesa. Najčešće se koristi u okviru drugih modela (npr. RUP, spiralni)
Spriralni	Vođen je analizom rizika. Primenjuje više iteracija, tj. petlji koje predtavljaju fazu razvoja. Dosta koristi prototipove za analizu rizika. Primenjuje se kod velikih i složenih projekata.
Agilni	Pogodan za manje projekte. Primenjuje postavljene vrednosti i principe. Primenjuje timski rad, saradnju sa kupcem, kontinualnu isporuku softvera, i reaguje na promene. Obezbeđuje brz, inkrementalni i iterativni razvoj, sa niskim troškovima režije.
RUP	Vođen je rizicima i slučajevima korišćenja. Arhitektura je na centralnommestu. Kao okvir za procese, omogućava primenu različitih stepene formalizacije procesa. Koristi 4 faze i za svaku se utvrđuje rizik. Naglasak je na postavljanju slučajeva upotrebe, projektovanje i arhitekturu. Prilagodljiv potrebama projektu. Nije pogodan za ugrađene sisteme.
Specijalizovani	Primenjujuspecifične metode ili tehnike za određene probleme. Model sa komponentama se koristi za uobičajene sisteme. Formalni metodi se koriste za osetljive sisteme (npr. navigacija aviona). Aspektno orijentisan razvoj izouluje određeni aspekt softvera i upravlja razvojem u odnosu na njega.

Slika 6.2 Tabela-2 Pregled najčešćih model procesa i preporuke za njihov izbor [1.3]

IZBOR MODELA SOFTVERSKOG PROCESA (VIDEO)

Izbor modela procesa razvoja softvera - vidoe klip 1

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

KLASIČNE GREŠKE PRI IZBORU SOFTVERSKOG PROCESA (VIDEO)

Izbor modela procesa razvoja softvera - video klip 7

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

Poglavlje 7

Poboljšanje softverskog procesa

PRISTUPI POBOLJŠANJU PROCESA

Poboljšanje procesa podrazumeva razumevanje postojećih procesa i promenu ovih procesa kako bi se povećao kvalitet proizvoda i/ili smanjili troškovi i vreme razvoja.

Danas postoji stalna potražnja industrije za jeftinijim, boljim softverom, koji se mora isporučiti u sve kraćim rokovima. Shodno tome, mnoge softverske kompanije su se okrenule poboljšanju softverskih procesa kao načinu poboljšanja kvaliteta svog softvera, smanjenja troškova ili ubrzanja procesa razvoja. Poboljšanje procesa podrazumeva razumevanje postojećih procesa i promenu ovih procesa kako bi se povećao kvalitet proizvoda i/ili smanjili troškovi i vreme razvoja.

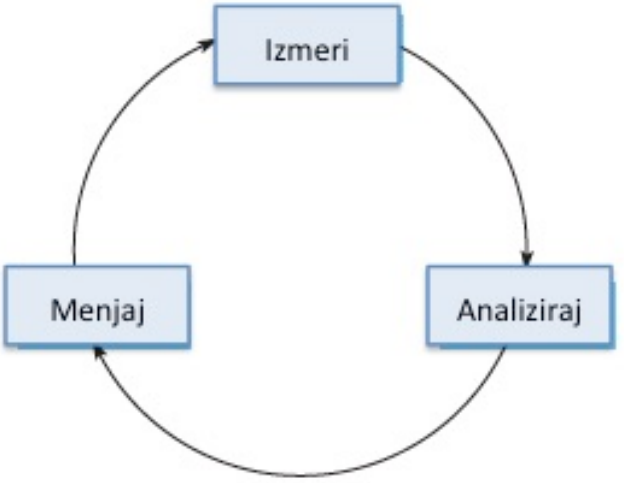
Koriste se dva sasvim različita pristupa poboljšanju i promeni procesa:

1. **Pristup zrelosti procesa**, koji se fokusirao na poboljšanje upravljanja procesima i projektima i uvođenje dobre prakse softverskog inženjeringa u organizaciju. Nivo zrelosti procesa odražava stepen do kojeg je dobra tehnička i upravljačka praksa usvojena u procesima razvoja organizacionog softvera. Primarni ciljevi ovog pristupa su poboljšani kvalitet proizvoda i predvidljivost procesa.
2. **Agilni pristup,** koji se fokusirao na iterativni razvoj i smanjenje troškova u softverskom procesu. Primarne karakteristike agilnih metoda su brza isporuka funkcionalnosti i odziv na promenljive zahteve kupaca. Filozofija poboljšanja je da su najbolji procesi oni sa najnižim troškovima i agilnim pristupima to mogu postići.

CIKLUS POBOLJŠANJA SOFTVERA

Postupak koristi Imzeri-Analiziraj-Menjaj faze poboljšanja softvera

Ljudi koji su entuzijastični i posvećeni svakom od ovih pristupa generalno su skeptični prema prednostima drugog. Pristup zrelosti procesa je ukorenjen u razvoju vođenom planom i obično zahteva povećane „opšte troškove“, u smislu da se uvode aktivnosti koje nisu direktno relevantne za razvoj programa.
Agilni pristupi se fokusiraju na kod koji se razvija i namerno minimizira formalnost i dokumentaciju.Opšti proces poboljšanja procesa koji leži u osnovi pristupa zrelosti procesa je cikličan proces, kao što je prikazano na slici 1



Slika 7.1 Ciklus poboljšanja softvera [1.2, Fig.2.11]

Faze u ovom procesu su

1. **Merenje procesa**: Merite jedan ili više atributa softverskog procesa ili proizvoda. Ova merenja čine osnovnu liniju koja vam pomaže da odlučite da li su poboljšanja procesa bila efikasna. Dok uvodite poboljšanja, ponovo merite iste atribute, za koje se nadamo da će se na neki način poboljšati.
2. **Analiza procesa:** Procenjuje se trenutni proces i identifikuju se slabosti i uska grla procesa. Modeli procesa (ponekad se nazivaju mape procesa) koji opisuju proces mogu se razviti tokom ove faze. Analiza se može fokusirati razmatranjem karakteristika procesa kao što su brzina i robusnost.
3. **Promena procesa:** Promene procesa se predlažu da bi se rešile neke od identifikovanih slabosti procesa. Oni se uvode i ciklus se nastavlja radi prikupljanja podataka o efektivnosti promena.

Bez konkretnih podataka o procesu ili softveru razvijenom pomoću tog procesa, nemoguće je proceniti vrednost poboljšanja procesa. Međutim, malo je verovatno da će kompanije koje započinju proces poboljšanja procesa imati dostupne podatke o procesu kao osnovu za poboljšanje. Stoga, kao deo prvog ciklusa promena, možda ćete morati da prikupite podatke o softverskom procesu i da izmerite karakteristike softverskog proizvoda.

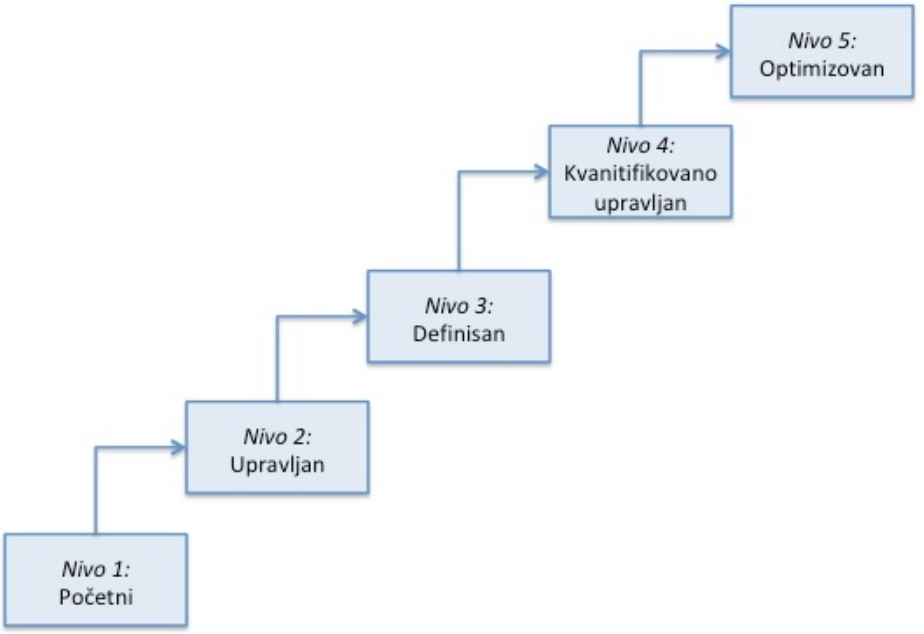
NIVOI ZRELOSTI OSPOSOBLJENOSTI PROCESA RAZVOJA SOFTVERA

Pet nivoa zrelosti procesa: početni, upravljan, definisan, kvanitifikovano upravljan i optimizovan

Unapređenje procesa je dugoročna aktivnost, tako da svaka od faza u procesu poboljšanja može trajati nekoliko meseci. To je takođe kontinuirana aktivnost jer, bez obzira na to koji novi

procesi budu uvedeni, poslovno okruženje će se promeniti i novi procesi će sami morati da evoluiraju kako bi te promene uzeli u obzir.

Pojam zrelosti procesa uveden je kasnih 1980-ih kada je Institut za softversko inženjerstvo (SEI) predložio svoj model zrelosti procesa . *Zrelost procesa softverske kompanije odražava upravljanje procesima, merenje i korišćenje dobrih praksi softverskog inženjeringa u kompaniji*. Ova ideja je uvedena kako bi Ministarstvo odbrane SAD moglo da proceni sposobnost softverskog inženjeringa odbrambenih izvođača, sa ciljem da ograniči ugovore na one izvođače koji su dostigli potreban nivo zrelosti procesa. Predloženo je pet nivoa zrelosti procesa. kao što je prikazano na slici 2. One su se razvile i razvile tokom poslednjih 25 godina ali su fundamentalne ideje SEI modela i dalje osnova za procenu zrelosti softverskog procesa.



Slika 7.2 Nivoi zrelosti osposobljenosti procesa razvoja softvera [1.2, Fig.2.12]

OPIS NIVOA ZRELOSTI OSPOSOBLJENOSTI PROCESA

Rad na nivoima zrelosti imao je veliki uticaj na softversku industriju

Nivoi u modelu zrelosti procesa su:

1. **Početni** : Ciljevi koji se odnose na procesnu oblast su zadovoljeni, a za sve procese je eksplicitno određen obim posla koji treba da se obavi i saopštava članovima tima.
2. **Upravljani**: Na ovom nivou, ciljevi povezani sa procesnom oblasti su ispunjeni, a uspostavljene su organizacione politike koje definišu kada svaki proces treba da se koristi. Moraju postojati dokumentovani projektni planovi koji definišu ciljeve projekta. Upravljanje resursimai procedure praćenja procesa moraju biti uspostavljene u celoj instituciji.
3. **Definisan**: Ovaj nivo se fokusira na organizacionu standardizaciju i implementaciju procesa. Svaki projekat ima upravljani proces koji je prilagođen zahtevima projekta iz definisanog skupa organizacionih procesa. Sredstva procesa i merenja procesa moraju se prikupiti i koristiti za buduća poboljšanja procesa.

4. **Kvantitativno upravljan**: Na ovom nivou, postoji organizaciona odgovornost za korišćenje statističkih i drugih kvantitativnih metoda za kontrolu podprocesa. To jest, prikupljena merenja procesa i proizvoda moraju se koristiti u upravljanju procesima.
5. **Optimizovan:** Na ovom najvišem nivou, organizacija mora da koristi merenja procesa i proizvoda kako bi pokrenula poboljšanje procesa. Trendovi se moraju analizirati i procesi prilagođavati promenljivim poslovnim potrebama.

Rad na nivoima zrelosti procesa imao je veliki uticaj na softversku industriju. Fokusirao je pažnju na procese i prakse softverskog inženjeringa koji su korišćeni i doveo do značajnih poboljšanja sposobnosti softverskog inženjeringa.

Međutim, *za mala preduzeća postoje preveliki troškovi u poboljšanju formalnog procesa, a procena zrelosti sa agilnim procesima je teška.* Shodno tome,*samo velike softverske kompanije sada koriste ovaj pristup* unapređenju softverskih procesa fokusiran na zrelost.

Rad na nivoima zrelosti procesa imao je veliki uticaj na softversku industriju. Fokusirao je pažnju na procese i prakse softverskog inženjeringa koji su korišćeni i doveo do značajnih poboljšanja sposobnosti softverskog inženjeringa.
Međutim, *za mala preduzeća postoje preveliki troškovi u poboljšanju formalnog procesa, a procena zrelosti sa agilnim procesima je teška*. Shodno tome, *samo velike softverske kompanije sada koriste ovaj pristup unapređenju softverskih procesa fokusiran na zrelost.*

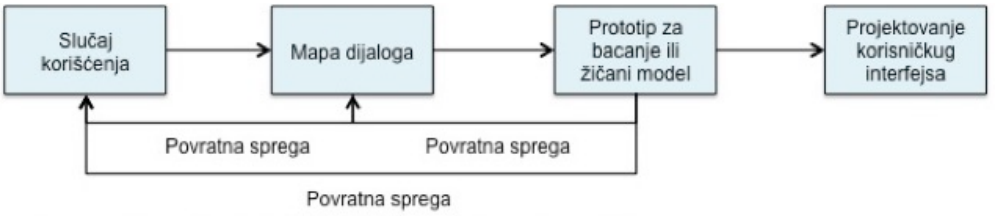
Poglavlje 8

Vežba - Pokazni primeri

POKAZNI PRIMER 1: PROCES KREIRANJA DIZAJNA KORISNIČKOG INTERFEJSA

Proces kreiranja dizajna korisničkog interfejsa

Na slici prikazan je jedan mogući niz razvojnih aktivnosti kojim se kreće od slučajeva korišćenja do detaljnog dizajna korisničkog interfejsa uz pomoć prototipa koji odbacuje.
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.



Slika 8.1 Proces kreiranja korisničkog interfejsa [Izvor:Karl Wiegers, Joy Beaty, Software Requirements, 3rd ed. Microsoft , 2013]

POKAZNI PRIMER 2: VEB STRANICA "PEARLS FROM SAND" -SLUČAJEVI KORIŠĆENJA

Slučaj korisnika pokazuje jedan scenario (redosled aktivnosti) koji reliazuje jedan funkcionalni zahtev softverskog procesa

Da bismo ceo ovaj proces učinili opipljivijim, pogledajmo stvarni primer, malu veb lokaciju za promociju knjige, memoara životnih lekcija pod nazivom "Biseri iz peska" ("Poears 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.

Jedan slučaj korišćenja pored glavnog scenarija koji realizuje određenu funkcionalnost softvera, sadrži i više pomoćnih scenarija, koji opisuju procese koji treba da se realizuju ako dođe do nekih poremećaja koji ometaju realizaciju glavnog scenarija.
Slika pokazuje slučajeve korišćenja i aktere, tj. učesnike u njemu, predstavljenih u vidu klasa (kategorija) korisnika

Klase korisnika	Slučajevi korišćenja
Posetilacc (Visitor)	Dobij informaciju o knjizi Dobij informaciju o autoru Čitanje uzorke poglavlja Čitanje bloga Kontkt 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

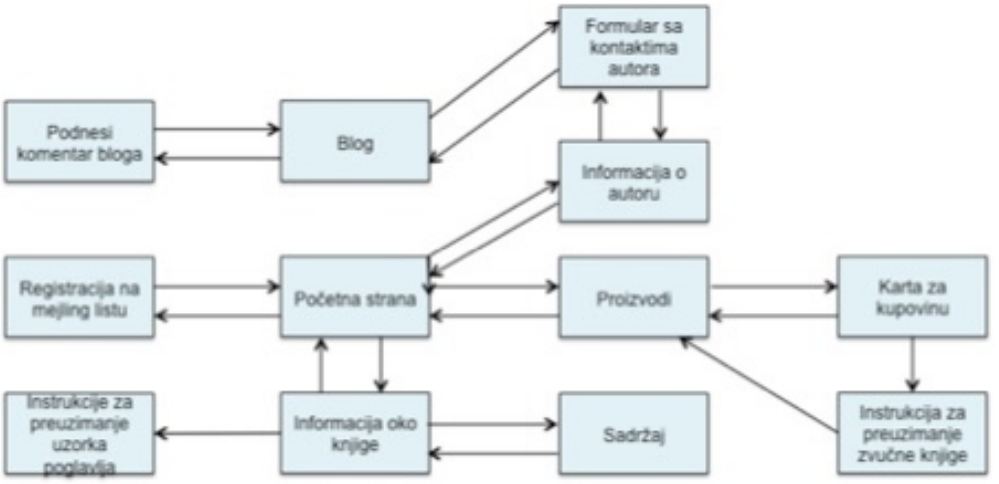
Izvor: Karl Wiegers, Joy Beaty, Software Requirements, 3rd ed., Microsoft, 2013

Slika 8.2 Slučajevi korišenja dobjienih od strane klasa korisnika

POKAZNI PRIMER 2: PLANIRANJE STRANICA

Plan stranica pokazuje redosled stranica prikaza u korisničkom intefejsu koji omogućava dijalog korisnika sa softverom.

Sledeći korak je bio planiranje stranica na kojima bi se obezbedile veb stranice i zamisliti navigacione staze među njima.
Svaki okvir predstavlja stranicu koja bi doprinela pružanju usluga identifikovanih u slučajevima korišćenja. Strelice predstavljaju veze koje omogućavaju navigaciju sa jedne stranice na drugu.

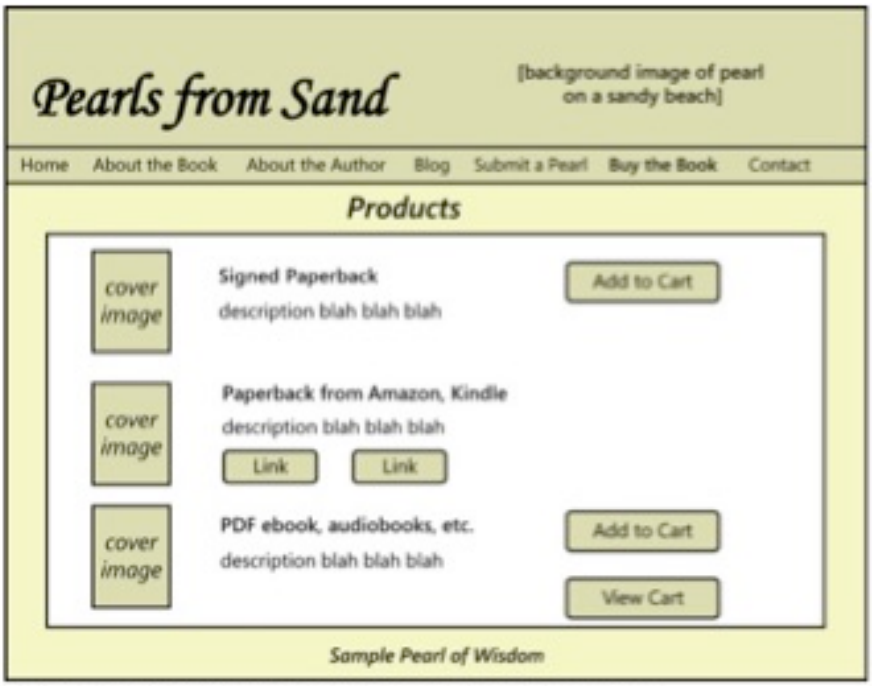


Slika 8.3 Planiranje potrebnih stranica [Izvor:Karl Wiegers, Joy Beaty, Software Requirements, 3rd ed. Microsoft , 2013]

POKAZNI PRIMER 2: ŽIČANI PROTOTIP - SKICA VEB STRANICE INTERFEJSA

Žičani model je skica sadržaja stranice koja se prikazuje korisniku. Time se vrši raspoređivanje sadržaja na stranici radi usklađivanja se željama korisnika.

Sledeći korak je bio konstruisanje prototipa ili tzv. žičanog okvira (skica) 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.



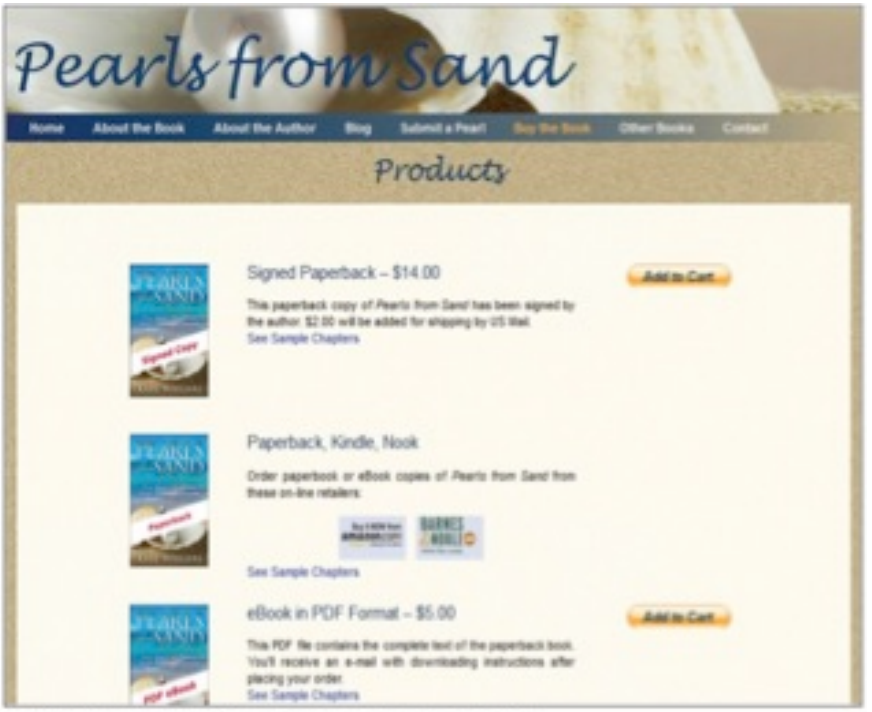
Izvor: Karl Wiegers, Joy Beaty, Software Requirements, 3rd ed., Microsoft, 2013

Slika 8.4 Skica rasporeda elemenata na veb stranica korisničkog interfejsa [Izvor:Karl Wiegers, Joy Beaty, Software Requirements, 3rd ed. Microsoft , 2013]

POKAZNI PRIMER 2: FINALNI IZGLED STRANICE KORISNIČKOG INTERFEJSA

Finalni izgled je konačan grafički izgled veb stranice koji korisnik sofvera vidi.

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



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

Slika 8.5 Finalni izgled veb stranice korisničkog interfejsa [Izvor:Karl Wiegers, Joy Beaty, Software Requirements, 3rd ed. Microsoft , 2013]

POKAZNI PRIMER 3: RUP MODEL SOFTVERSKOSKOG PROCESA RESTORAN

Opis osnovnih faza RUP modela softverskog sistema restorana

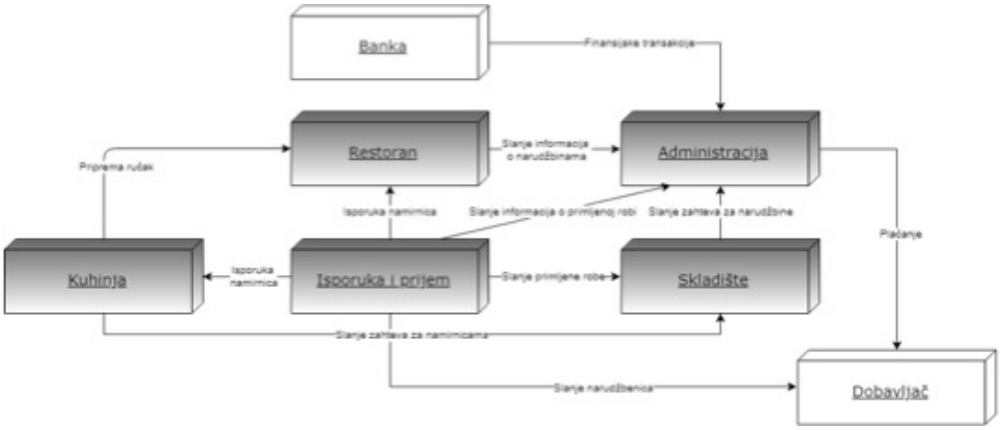
Korišćenjem RUP procesa, primenjuju se sledeće faze u razvoju:

1. **Početak** - inicijalna evaluacija se sprovodi u cilju utvrđivanja da li je projekat vredan pažnje, kreira se biznis model, definisan kroz viziju sistema. Poslovni plan je detaljno definisan, a treba definisati i procenu troškova razvoja i rasporeda. Postignut je dogovor o obimu projekta sa svim zainteresovanim stranama.
2. **Razrađivanje** - u toku razrade, vrši se detaljnija evaluacija, kreira se plan razvoja i ublažavaju ključni rizici. Razvojni tim piše 80% svih slučajeva korišćenja, stvara sistemsku arhitekturu i plan razvoja.
3. **Konstrukcija** - u toku izgradnje, stvara se softverski sistem - kod je napisan i testiran. Tim takođe testira rezultujući softver. Ključni izlaz ove faze je operativni softver
4. **Tranzicija** - u fazi tranzicije, softver se izdaje krajnjem korisniku. Kada se prijava prihvati, projekat je formalno objavljen, a tim nastavlja da ga održava.

Da bi projekat nesmetano mogao da se razvija definisaćemo komunikacioni plan projekta i prikazati sva dokumenta relevantna za projekat. RUP je okvir koji vam omogućava da izaberete model procesa koji najviše odgovara specifičnostima softvera koji razvijate, jer možete birati stepen formalizma procesa!

POKAZNI PRIMER 3: RESTORAN - OPIS SISTEMA

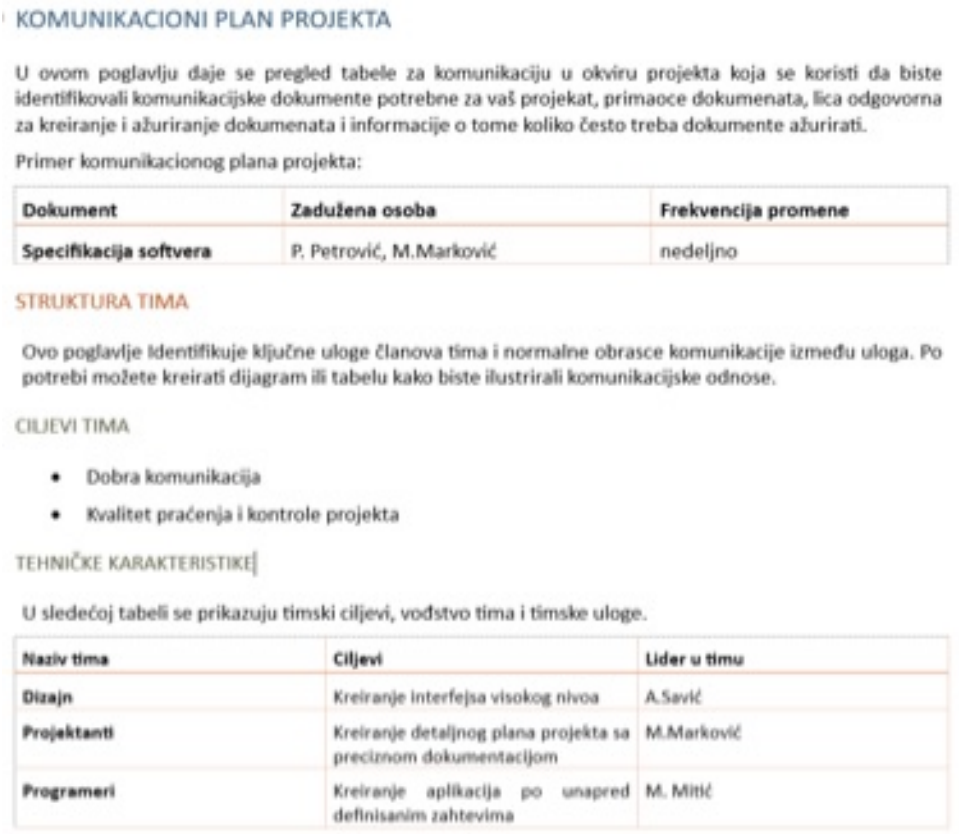
Struktura sistema pokazuje aktere, tj. učesnike (organiyacione jedinice korisnika softvera i spoljne organizacije sa kojim softver komunicira) u softverskom procesu



Slika 8.6 Struktura sistema za restoran [Izvor: Autor]

POKAZNI PRIMER 3: FAZA 11 - POSTAVLJANJA PLANA KOMUNIKACIJA

Plan komunikacje daje pregled ko sa kim komunicira, te se koristi u struktura projektnog tima.



KOMUNIKACIONI PLAN PROJEKTA

U ovom poglavlju daje se pregled tabele za komunikaciju u okviru projekta koja se koristi da biste identifikovali komunikacijske dokumente potrebne za vaš projekat, primaoce dokumenata, lica odgovorna za kreiranje i ažuriranje dokumenata i informacije o tome koliko često treba dokumente ažurirati.

Primer komunikacionog plana projekta:

Dokument	Zadužena osoba	Frekvencija promene
Specifikacija softvera	P. Petrović, M.Marković	nedeljno

STRUKTURA TIMA

Ovo poglavlje identifikuje ključne uloge članova tima i normalne obrasce komunikacije između uloga. Po potrebi možete kreirati dijagram ili tabelu kako biste ilustrirali komunikacijske odnose.

CILJEVI TIMA

- Dobra komunikacija
- Kvalitet praćenja i kontrole projekta

TEHNIČKE KARAKTERISTIKE

U sledećoj tabeli se prikazuju timski ciljevi, vođstvo tima i timske uloge.

Naziv tima	Ciljevi	Lider u timu
Dizajn	Kreiranje interfejsa visokog nivoa	A.Savić
Projektanti	Kreiranje detaljnog plana projekta sa preciznom dokumentacijom	M.Marković
Programeri	Kreiranje aplikacija po unapred definisanim zahtevima	M. Mitić

Slika 8.7 Komunikacijoni plan prototipa

POKAZNI PRIMER 3: FAZA 1 - BIZNIS MODEL RESTORANA

Biznis model pokazuje korisnike i njihove relacije, ključne aktivnosti i komponente sistema, i rspored budžeta projekta po glavnim fazama razvoja softvera.

Biznis model - restoran

Datum: 11.09.2018.

Verzija: 1.1

Korisnici	Ključne aktivnosti	Relacije između korisnika	Finansijski segment
Gost restorana Konobar Supervizor Kuvar	Priprema hrane. Nabavka Skladištenje namirnica Kontora i revizija naručivanja i isporuke hrane. Naručivanje hrane. Plaćanje. Kontrola procesa.	Supervizor ima sve opcije ostalih korisnika: gosta, konobara i kuvara.	Faza 1 - Početak - 20% ukunog budžeta Faza 2 - Razrađivanje - 30% ukunog budžeta Faza 3 - Konstrukcija - 40% ukunog budžeta Faza 4 - Tranzicija - 10% ukunog budžeta
	Ključne komponente Restoran Kuhinja Administracija Skladište Isporuka i prijem Banka Dobavljači		

Slika 8.8 Biznis model restorana

Poglavlje 9

Vežba - Zadaci za individualni rad

ZADACI

Kombinovanje grupnih diskusija i individualnih zadataka.

1. Objasnite zašto je promena neizbežna u složenim sistemima i navedite primere (osim izrade prototipa i inkrementalne isporuke) aktivnosti softverskog procesa koje pomažu u predviđanju promena i čine softver koji se razvija otpornijim na promene.
2. Objasnite zašto sistemi razvijeni kao prototipovi ne bi trebalo normalno da se koriste kao proizvodni sistemi.
3. Objasnite zašto je spiralni model prilagodljiv model koji može podržati i izbegavanje promena i aktivnosti tolerancije promena. U praksi, ovaj model nije imao široku primenu. Predložite zašto bi to mogao biti slučaj.
4. Koje su prednosti pružanja statičkih i dinamičkih pogleda na softverski proces kao u Rational Unified Process?
5. Istorijski gledano, uvođenje tehnologije je izazvalo duboke promene na tržištu rada i, barem privremeno, udaljilo je neke ljude sa posla. Razgovarajte o tome da li će uvođenje opsežne automatizacije procesa verovatno imati iste posledice za softverske inženjere. Ako mislite da neće, objasnite zašto ne. Ako mislite da će to smanjiti mogućnosti zapošljavanja, da li je etički da se pogođeni inženjeri pasivno ili aktivno opiru uvođenju ove tehnologije?

Napomena: Studenti koji urade bar deo ovih zadataka posle predavanja, a pre vežbanja, i dostave ih saradniku koji vodi vežbe pre vežbanja, dobiće poene na zalaganje u nastavi.

DOMAĆI ZADATAK BR. 2

Tekst domaćeg zadatka broj 2.

Dajte primer primene uvođenja softera u poslovanje neke organizacije ili neka radnje ili e-usluge (vama poznate) i uradite sledeće:

1. Postavite softerski proces razvoja potrebnog softvera.
2. Izaberite model softverskog procesa koji smtarate da najviše odgovara tipu softvera koji treba razviti.
3. Objasnite razloge za izbor modela softverskog procesa koji ste izvršili

Napomena: Domaći zadatak bi trebalo da uradite i rešenje pošaljete saradniku koji drži vežbe u roku od 7 dana od dana vežbi na koji je zadatak objavljen. Ukoliko ne ispunite taj rok,

gubite 50% predviđenih poena koje dobijate za vaše rešenje domažeg zadataka. Krajnji rok za predaju domaćeg zadatke je 10 dana pre dana polaganja ispita.

Poglavlje 10

Zaključak

ZAKLJUČAK

1. Procesi treba da obuhvataju aktivnosti za suočavanje sa promenama. Ovo može uključivati fazu izrade prototipa koja pomaže da se izbegnu loše odluke o zahtevima i dizajnu. Procesi mogu biti strukturirani za iterativni razvoj i isporuku tako da se promene mogu vršiti bez ometanja sistema u celini.
2. Rational objedinjeni proces je moderan generički model procesa koji je organizovan u faze (početak, razrada, konstrukcija i tranzicija) ali odvaja aktivnosti (zahteve, analizu i dizajn, itd.) od ovih faza.
3. Poboljšanje procesa je proces poboljšanja postojećih softverskih procesa radi poboljšanja kvaliteta softvera, smanjenja troškova razvoja ili smanjenja vremena razvoja. To je cikličan proces koji uključuje merenje procesa, analizu i promenu.

LITERATURA

Literatura korišćena u ovoj lekciji

1. Obavezna literatura:

1. Nastavni materijal za e-učenje na predmetu *SE101 Razvoj softvera i inženjera softvera*, Univeziitet Metropolitan, školska 2022/23. godina
2. **Ian Sommervile,** *Software Engineering - Chapter 2 Software processes, poglavlje 2.3, 2.4* , Tenth Edition, Pearson Education Inc., 2016.
3. **Ian Sommerville**, *SOFTWARE ENGINEERING*, Chapter 2, poglavlje 2.3.3., 2.4 Ninth Edition, Pearons, 201

2. Dopunska litertura:

1. King Graham, Wang Yingxu, Software Engineering Processes - Principles and Applications,CRC Press (2000)
2. Ian Sommervile,Software Products - An Introduction to Modern Software Engineering, Global Edition, Pearson (2021)

3. Veb lokacije :

1. https://sceweb.uhcl.edu/helm/RationalUnifiedProcess /
2. https://www.techopedia.com/definition/205/aspect-oriented-software-development-aosd
3. https://www.guru99.com/what-is-software-engineering.html
4. https://www.guru99.com/difference-system-software-application-software.html#2