Softversko inženjerstvo

Elektronski fakultet Niš

Evolucija softvera

Sadržaj

- Evolucioni proces
  - Proces promene softverskih sistema
- Dinamika evolucije programa
  - Razumevanje evolucije softvera
- Održavanje softvera
  - Menjanje softvera koji je u operativnoj upotrebi
- Upravljanje nasleđenim (“legacy”) sistemima
  - Donošenje odluka u pogledu softverskih izmena

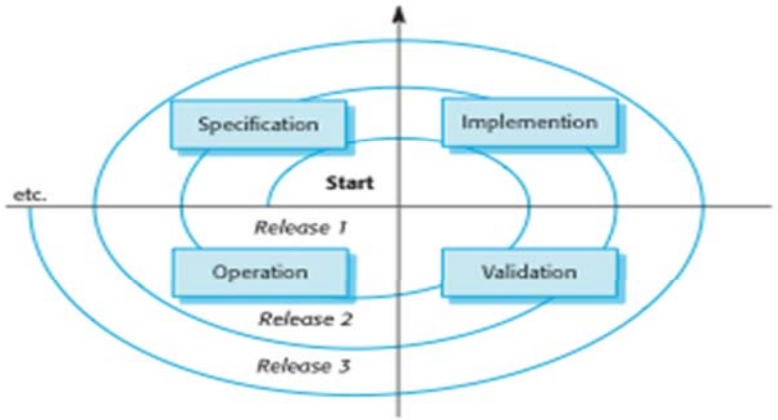
Izmena softvera

- Promene softvera su neizbežne
  - Javljaju se novi zahtevi u toku upotrebe softvera;
  - Menja se poslovno okruženje;
  - Otkrivaju se greške koje potrebno ispraviti;
  - Novi računari i oprema se dodaju u sistem;
  - Performanse i/ili pouzdanost sistema se moraju unaprediti.
- Ključni problem za sve organizacije je implementacija i upravljanje izmenana nad postojećim softverskim sistemima.

Značaj evolucije

- Organizacije ulažu veliki novac u svoje softverske sisteme jer su najčešće kritični za njihovo poslovanje.
- Da bi se očuvala vrednost softvera potrebno je da se softver menja i ažurira.
- Glavnica budžeta za softver u velikim kompanijama je namenjena izmeni i evoluciji postojećih softverskih sistema, pre nego razvoju novih softvera.

Spiralni model razvoja i evolucije softvera



Evolucija i servisiranje

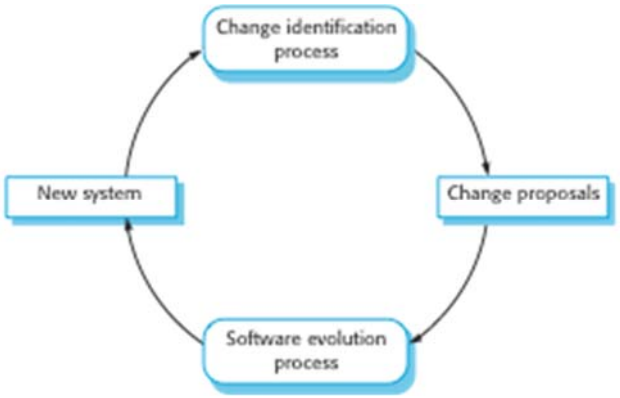
Evolucija i servisiranje

- Evolucija
  - Faza u životnom ciklusu softverskog sistema gde se softver nalazi u operativnoj upotrebi i evoluira kako se novi zahtevi implementiraju u sistem.
- Servisiranje
  - U ovaj fazi softver se i dalje operativno koristi, a izmene koje se vrše imaju za cilj održavanje njegove operativnosti.
  - Ovo praktično znači da se ne dodaju nove funkcionalnosti, već se samo otklanjaju greške i vrše izmene usled promena u okruženju softvera.
- Postepeno ukidanje (eng. *phaseout*)
  - Softver se može i dalje koristiti, ali se nad njim ne vrše više nikakve izmene.

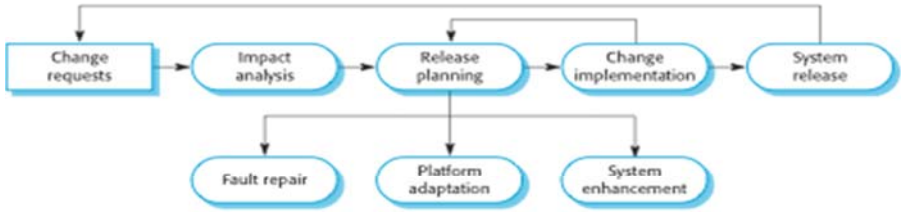
Evolucioni proces

- Proces evolucije softvera zavisi od
  - Tipa softvera koji se održava;
  - Korišćenog procesa razvoja softvera;
  - Veština i iskustva ljudi uključenih u proces.
- Predlozi za izmenana su pokretač evolucije sistema.
  - Treba ih povezivati sa komponentama na koje izmena utiče, a sa ciljem procene cene i uticaja odgovarajuće izmene.
- Identifikacija izmena i evolucija se nastavljaju u toku životnog veka sistema.

Identifikacija izmena i evolucioni proces



Proces evolucije softvera



Implementacija izmene

Implementacija izmena

- Reč je o iteraciji razvojnog procesa gde se projektuju, implementiraju i testiraju izmene.
- Glavna razlika u odnosu na osnovnu implementaciju je u tome što je neophodno poznavanje softvera koji se menja. Ovo je naročito kritično kada izmene ne vrši tim koji je razvio originalni softver.
- U toku faze upoznavanja sa softverom, neophodno je upoznati se sa strukturom softvera, na koji način su realizovane funkcionalnosti i na koji način će predložena izmena uticati na softver.

Zahtevi za hitnim izmenama

- Hitne izmene se mogu implementirati bez prolaska kroz sve faze procesa razvoja softvera. Vrše su u sledećim slučajevima:
  - Kada imamo ozbiljni otkaz sistema koji mora da se popravi kako bi sistem mogao da se vrati u normalnu upotrebu;
  - Ukoliko promene u okruženju sistema (npr. ažuriranje OS-a) izazovu neočekivane efekte;
  - Ukoliko se jave promene u poslovanju takve da je neophodan brz odgovor (npr. izlazak konkurentskog proizvoda).

Proces hitne popravke

Agilne metode i evolucija

- Agilne metode su zasnovane na inkrementalnom razvoju, tako da je prelazak sa razvoja na evoluciju vrlo jednostavan.
  - Evolucija je zapravo nastavak procesa razvoja zasnovan na čestom izdavanju novih verzija.
- Automatizovano testiranje je posebno značajno kada se izvrši izmena sistema.
- Izmene se mogu predstaviti kao dodatne korisniče priče (termin iz SCRUM-a).

Problemi kod primopredaje

- Nastaju u slučajevima kada je razvojni tim koristio agilni razvoj, a tim za evoluciju preferira planom vođeni razvoj.
  - Tim za evoluciju može da očekuje detaljnu dokumentaciju kao podršku evolucionom procesu, a ona ne postoji kao rezultat agilnog procesa.
- Kada se razvoj vrši planom vođenim procesom, a evolucioni tim preferira agilne metode.
  - Tim za evoluciju će morati da razvija automatizovane testove od nule, a i kod najverovatnije nije refaktorisan i uprošćen kako se očekuje kod agilnog razvoja.

Dinamika evolucije programa

- Dinamika evolucije programa je studija procesa izmene sistema.
- Nakon nekoliko emirijskih studija, Liman i Belejdi su predložili određen broj “zakona” koji se mogu primeniti na sisteme u toku evolucije.
- Reč više o zapažanjima nego zakonima koja se mogu primeniti na velike sisteme razvijene od strane velikih organizacija.
  - Nije najjasnije da li se mogu primeniti na druge tipove softverskih sistema.

Limanovi zakoni

Zakon	Opis
Kontinuirane izmene	Program koji se koristi u realnom okruženju nužno mora da se povremeno menja ili će u suprotnom postepeno postajati sve manje upotrebljiv.
Povećanje složenosti	Kako se program koji evoluira menja, njegova struktura postaje sve složenija. Potrebno je izdvajanje dodatnih resursa za održavanje i uprošćavanje strukture.
Evolucija velikih programa	Evolucija programa je samoregulišući proces. Atributi sistema kao što su veličina, vreme između izdavanja novih verzija i broj prijavljenih grešaka se najčešće ne menjaju od verzije do verzije.
Organizaciona stablinost	U životnom veku programa stopa razvoja je uglavnom konstantna i ne zavisi od količine resursa dodeljenih za razvoj.

Limanovi zakoni

Zakon	Opis
Očuvanje familijarnosti	U životnom veku sistema, inkrementalne izmene za svaku novu verziju su približno konstantne.
Stalni rast	Funkcionalnosti koje sistem nudi moraju stalno da rastu da bi se održalo zadovoljstvo korisnika.
Opadanje kvaliteta	Kvalitet sistema će opadati ukoliko se ne menja na osnovu izmena u radnom okruženju.
Sistem za dobijanje povratnih informacija	Evolucioni proces uključuje sistem za dobijanje povratnih informacija koje se koriste u cilju postizanja značajnih unapređenja proizvoda.

Rezime

- Razvoj i evolucija softvera se mogu posmatrati kao integrisani, iterativni proces predstavljen korišćenjem spiralnog modela.
- Za sisteme po narudžbini, cena održavanja najčešće prevazilazi cenu originalnog razvoja softvera.
- Proces evolucije softvera je vođen zahtevima za izmenama i uključuje analizu uticaja izmene, planiranje novih verzija (izdanja) i implementaciju izmena.
- Limanovi zakoni, kao što je tvrđenje da su promene kontinuirane, opisuju određena saznanja dobijena dugoročnim studijama evolucije sistema.

Evolucija softvera

Čas 2

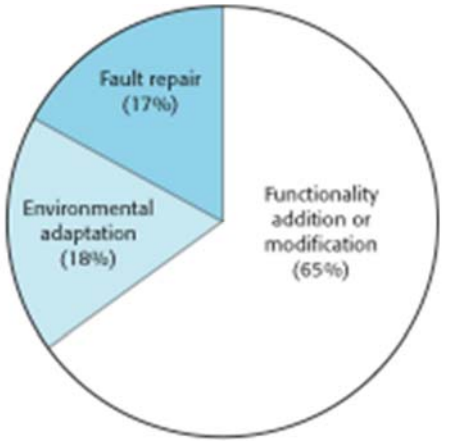
Održavanje softvera

- Izmena programa nakon puštanja u upotrebu.
- Ovaj termin se uglavnom koristi da opiše izmene softvera razvijenog po narudžbini. Za generičke softverske proizvode se kaže da evoluiraju kroz nove verzije.
- Održavanje najčešće ne uključuje velike izmene u arhitekturi sistema.
- Izmene se implementiraju modifikacijom postojećih komponenti i dodavanjem novih komponenti u sistem.

Tipovi održavanja

- Održavanje u cilju ispravke softverskih grešaka
  - Izmena sistema kako bi se otklonili nedostaci koji sprečavaju da sistem radi u skladu sa svojom specifikacijom.
- Održavanje sa ciljem prilogođenja softvera za drugačije radno okruženje
  - Promena sistema tako da može da radi u drugačijem okruženju (računar, OS, itd.) od onog za koje je inicijalno implementiran.
- Održavanje u cilju dodavanja nove ili izmene postojeće funkcionalnosti sistema
  - Modifikacija sistema kako bi zadovoljio nove zahteve.

Raspodela truda za održavanje



Cena održavanja

- Najčešće je veća od cene razvoja (2 do 100 puta u zavisnosti od primene).
- Na nju utiču i tehnički i netehnički faktori.
- Cena raste sa trajanjem održavanja. Održavanjem se narušava struktura softvera, pa dalje održavanje postaje sve složenije.
- Star softver najčešće ima visoku cenu održavanja (npr. stari jezici, kompajleri, itd.).

Faktori koji utiču na cenu održavanja

- Stabilnost tima
  - Cena održavanja se smanjuje ako održavanje vrši isti tim duže vreme.
- Ugovorna odgovornost
  - Ukoliko ne postoji ugovorna obaveza sa razvojnim timom za buduće održavanje, ne postoji ni podsticaj da se razvije softver koji je pogodan za održavanje.
- Veštine zaposlenih
  - Zaposleni na održavanju su najčešće manje iskusni i imaju limitirano domensko znanje.
- Starost programa i struktura
  - Kako program stari, njegova struktura se degradira pa time postaje teža za razumevanje i izmenu.

Predviđanje održavanja

- Predviđanje održavanja se bavi procenom koji delovi sistema su potencijani uzročnici problema i mogu imati viskoku cenu održavanja
  - Prihvatanje izmena zavisi od pogodnosti za održavanje komponenti na koje izmena utiče;
  - Implementacijom izmena se degradira sistem i smanjuje se njegova pogodnost za održavanje;
  - Cena održavanja zavisi od broja izmena, a cena izmene zavisi od toga koliko je sistem pogodan za održavanje.

Metrika složenosti

- Pogodnost za održavanje sa može predvideti procenom složenosti komponenti sistema.
- Istraživanja su pokazala da se najveći deo napora oko održavanja potroši na relativno malom broju komponenti sistema.
- Složenost zavisi od
  - Složenosti upravljačkih struktura;
  - Složenosti struktura podataka;
  - Veličine klasa, metoda i modula.

Metrika procesa

- Pogodnost za održavanje se može predvideti korišćenjem metrike procesa
  - Broj zahteva za ispravkama;
  - Srednje vreme potrebno za analizu uticaja;
  - Srednje vreme potrebno da se implementira izmena;
  - Broj istaknutih zahteva za izmenama.
- Ako neka ili sve ove metrike rastu, to predstavlja indikaciju da opada pogodnost za održavanje.

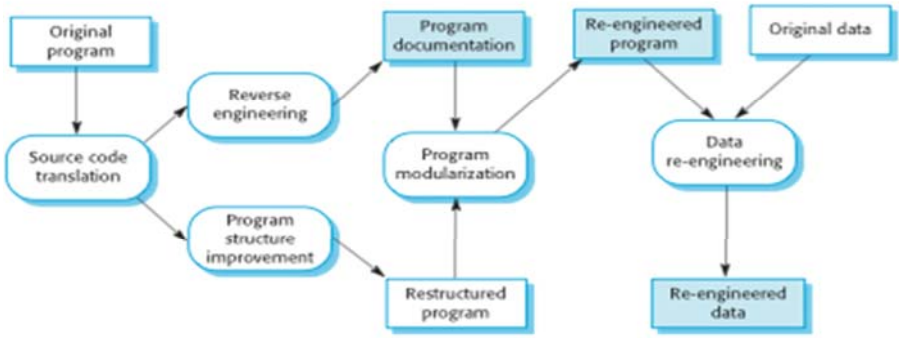
Reinženjering sistema

- Reč je o restruktuiranju ili ponovnom pisanju dela ili celog nasleđenog sistema bez promene njegove funkcionalnosti.
- Primenjiv je onda kada neki ali ne svi podsistemi većeg sistema zahtevaju često održavanje.
- Reinžinjering predstavlja ulaganje truda sa ciljem lakšeg održavanja. Sistem može biti restruktuiran i redokumentovan.

Prednosti reinženjeringa

- Smanjenje rizika
  - Kod razvoja novog softvera postoji veliki rizik. Mogu da se jave problemi kod razvoja, problemi sa zaposlenima, kao i problemi sa specifikacijom.
- Smanjenje cene
  - Cena reinženjeringa je najčešće znatno manja od cene razvoja novog softvera.

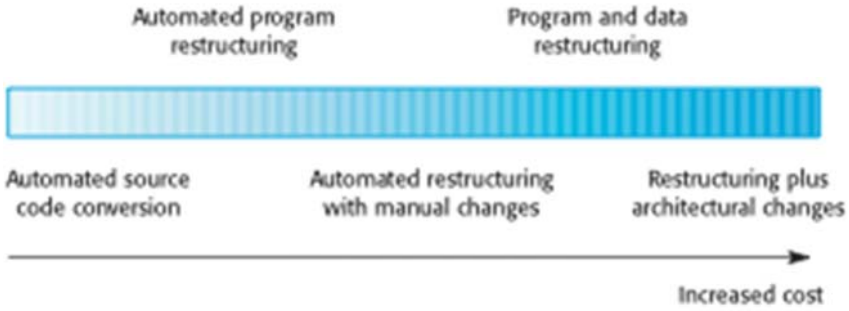
Proces reinženjeringa



Aktivnosti u procesu reinženjeringa

- Prevođenje izvornog koda
  - Prebacivanje koda u novi jezik
- Reverzni inženjering
  - Analiza programa u cilju boljeg razumevanja
- Unapređenje strukture programa
  - Automatsko restruktuiranje programa
- Modularizacija programa
  - Reorganizacija strukture programa
- Reinženjering podataka
  - Čišćenje i restruktuiranje podataka

Pristupi u reinženjeringu



Faktori koji utiču na cenu reinženjeringa

- Kvalitet softvera koji je predmet reinženjeringa
- Dostupnost alata za reinženjering
- Obim podataka za koje je potrebna konverzija
- Dostupnost domenskih eksperata za reinženjering
  - Ovo može biti problem kod starih sistema koji koriste tehnologije koje više nisu u širokoj upotrebi

Preventivno održavanje kroz refaktorisanje

- Refaktoring je proces unapređenja programa kako bi se smanjila njegova degradacija kroz izmene
- Refaktoring predstavlja “preventivno održavanje” koje smanjuje probleme pri budućim izmenama
- Refaktoring uključuje izmenu programa kako bi se poboljšala njegova struktura, smanjila složenost ili povećala razumljivost
- Kada se refaktoriše program treba izbegavati dodavanje novih funkcionalnosti, već se skoncentrisati na njegovo unapređenje

Refaktorisanje i reinženjering

- Reinženjering se obavlja nakon što je sistem bio održavan neko vreme i cena održavanja je porasla
  - Reinženjering nasleđenih sistema se vrši korišćenjem automatskih alata, kako bi novodobijeni sistem bio lakši za održavanje
- Refaktorisanje je kontinuirani proces unapređivanja kroz razvoj i evoluciju
  - Koristi se kako bi se izbegla degradacija strukture i koda, a u cilju umanjenja cene i složenosti održavanja sistema

Loša praksa u kodiranju

- Kopiranje (dupliranje) koda
  - Isti ili vrlo sličan kod koji se javlja na više mesta u programu. Ovo se razrešava pravljenjem novog metoda koji sadrži zajednički kod i koji se poziva sa odgovarajućih mesta po potrebi
- Predugački metodi
  - Ako je metod predugačak, treba ga razdeliti u više manjih metoda
- Grupice podataka
  - Javlja se kada se ista grupa podataka (atributi u klasama, parametri u metodama) iznova javlja na nekoliko mesta u programu. U ovom slučaju treba dodati novu klasu koja ih enkapsulira.

Upravljanje nasleđenim (legacy) sistemima

- Organizacije koje se oslanjaju na nasleđene (legacy) sisteme moraju da izaberu neku od strategija za evoluciju ovih sistema
  - U potpunosti otpisati sistem i izmeniti poslovni proces tako da se više ne oslanja na njega;
  - Nastaviti sa održavanjem sistema;
  - Transformisati sistem reinženjeringom kako bi bio lakši za održavanje;
  - Zameniti sistem sa novim sistemom.
- Izbor strategije treba da zavisi od kvaliteta sistema i poslovne vrednosti.

Kategorije nasleđenih sistema

- Loš kvalitet, mala poslovna vrednost
  - Ovakve sisteme treba otpisati.
- Loš kvalitet, velika poslovna vrednost
  - Ovi su bitni, ali je skupo održavanje. Treba primeniti reinženjering ili ih zameniti nekim drugim prikladnim sistemom.
- Visok kvalitet, mala poslovna vrednost
  - Zameniti sa gotovim rešenjem, otpisati ili održavati.
- Visok kvalitet, velika poslovna vrednost
  - Nastaviti sa korišćenjem uz normalno održavanje.

Procena poslovne vrednosti

- Ova procena treba da uzme u obzir različita gledišta
  - Krajnje korisnike sistema;
  - Poslovne korisnike;
  - Niže rukovodioce;
  - IT rukovodioce;
  - Više rukovodioce.
- Obavlja se intervjuisanjem različitih zainteresovanih strana i sravnjivanjem rezultata.

Pitanja pri proceni poslovne vrednosti

- Upotreba sistema
  - Ako se sistem koristi samo povremeno od strane malog broja ljudi, onda je njegova poslovna vrednost mala.
- Poslovni proces koji zahteva
  - Sistem može imati malu poslovnu vrednost ako zahteva korišćenje neefikasnog poslovnog procesa.
- Pouzdanost sistema
  - Ako sistem nije pouzdan i problemi direktno pogađaju poslovne korisnike, sistem ima malu poslovnu vrednost.
- Izlazi iz sistema
  - Ako posao zavisi od izlaza koje daje sistem, onda sistem poseduje veliku poslovnu vrednost.

Procena kvaliteta sistema

- Procena poslovnog procesa
  - U kojoj meri poslovni proces podržava tekuće ciljeve organizacije?
- Procena okruženja
  - U kojoj meri je efikasno okruženje u kom sistem radi i koliko je skupo njegovo održavanje?
- Procena aplikacije
  - Koji je kvalitet aplikativnog softvera?

Faktori koji se koriste u proceni okruženja

- Stabilnost dobavljača opreme
- Frekvencija otkaza opreme
- Starost opreme
- Performanse
- Tehnička podrška
- Cena održavanja okruženja
- Interoperabilnost sa drugim sistemima

Faktori koji se koriste u proceni aplikacije

- Razumljivost izvornog koda
- Dokumentacija
- Podaci
- Performanse
- Programski jezik
- Upravljanje konfiguracijom (verzijama)
- Test podaci
- Ljudstvo obučeno za održavanje

Rezime

- Postoje 3 tipa održavanja softvera: ispravka grešaka, izmena softvera za rad u novom okruženju i implementacija novih ili promena postojećih funkcionalnosti.
- Reinženjering softvera se bavi restruktuiranjem i redokumentovanjem softvera kako bi ga učinio lakšim za razumevanje i izmenu.
- Refaktorisanje predstavlja izmenu programa uz očuvanje postojećih funkcionalnosti, a sa ciljem preventivnog održavanja.
- Da bi se odlučilo da li će neki nasleđeni (legacy) sistem biti zamenjen, transformisan ili održavan, potrebno je proceniti poslovnu vrednost i kvalitet aplikacije.