PT&SCHE

Introduction of part-time and short cycle studies in Serbia

Funded by the Erasmus+ Programme of the European Union

This project has been funded with support from the European Commission. This publication [communication] reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

UNIVERZITET METROPOLITAN BEOGRAD

KI206 - PROCES I METODOLOGIJE RAZVOJA SOFTVERA

Šta je softversko inženjerstvo

Lekcija 01

PRIRUČNIK ZA STUDENTE

KI206 - PROCES I METODOLOGIJE RAZVOJA SOFTVERA

Lekcija 01

ŠTA JE SOFTVERSKO INŽENJERSTVO

UVOD

Softversko inženjerstvo je inženjerska disciplina koja se bavi razvojem i proizvodnjom softvera

Cilj ove uvodne nastavne jedinice da vam objasni šta je softversko in\enjerstvo. Na kraju, očekuje se da ćete

- razumeti šta je softversko inženjerstvo i zašto je ono važno,
- razumeti da razvoj softvera različitog tipa zahteva primenu različitih tehnika softverskog inženjerstva,
- razumeti neka etička i profesionalna pitanja koja su važna ya inženjere softvera,

Upoynaćete se i sa primerima tri različita tipa softverskih sistema koji će se analizirati u svim lekcijama ovog predmeta, kao pokazni slučajevi softverskih sistema.

Poglavlje 1

KP-Šta je softver?

SOFTVERSKI SISTEM

Softver čine računarski programi i njihova dokumentacija

Svuda gde se koriste računari, koristi se i neki softver, jer računar bez softvera ne može da radi. No, kako su danas računari svuda oko nas (u automobilu, u mobilnom telefonu, u sistemu upravljanja liftovima, u alatnim mašinama itd.), to su i softverski sitemi postali deo našeg života, jer su svuda oko nas. Možete li zamislite život savremenog bez njihovog postajanja ? Način života bi morao da nam se vrati bar pola veka unazad.

Softverski sistemi su apstraktni i nedodirljivi. Ne zavise od svojstava nekog materijala, ne zavise od zakona fizike, ili od nekog proizvodnog procesa, što je sve karakteristično za fizičke proizvode. Zbog svog nefizičkog karaktera, nisu ograničeni zakonima fizike, tako da softverski sistemi mogu da brzo postanu i vrlo složeni, teško razumljivi i skupi za održavanje, tj. za menjanje.

Postoje puno različitih vrsta (tipova) softverskih sistema, počev od vrlo jednostavnih, tzv. ugrađenih sistema (npr. sistem upravljanja pojedinim podsistemima u automobilima) do vrlo kompleksnih globalnih informacionih sistema (npr. sistemi za rezervaciju hotelskog smeštaja, kupovinu avionskih karata i dr.).

Navešćemo, tri slučaja razvoja softvera, da bi ilustrovali kako se opisuje neki problem ya koji je potrebno ratviti odrgovarajući softver. Ova tri slučaja ćemo kasnije koristiti da bi opisali detaljnije rayličite aktivnosti procesa rayvoja softvera. Primenom ova tri slučaja, pokazaćemo kako se opisuje problem za koji je potrebno razviti softver.

UZROCI GREŠAKA U SOFTVERU

Greške u softveru su posledica povećanih očekivanja korisnika, s jedne strane, i niskog stepena primene metoda i tehnika softverskog inženjerstva, s druge strane.

Do mnogih grešaka u radu softvera dolazi iz sledećih razloga:

1. **Povećani zahtevi**. Razvoj softvera sve je više izložen novim zahtevima koji čine da softver bude sve složeniji. Ti zahtevi traže da se u što kraćem vremenskom roku realizuju. Postojeće metode softverskog inženjerstva ne mogu da se nose sa ovakvim zahtevima, jer ne mogu da ih zadovolje. Moraju se razviti nove tehnike softverskog inženjerstva koje mogu da zadovolje ove zahteve.

2. **Niska primena metoda softverskog inženjerstva:** Moguć je razvoj računarskih programa i bez primene metoda i tehnika softverskog inženjerstva. Često, organizacije počnu sisteme koji rade nepouzdano, te razvoj programa za svoje potrebe jer im je to potrebno. Najčešće iz neznanja, ne koriste metode i tehnike softverskog inženjerstva, već njihovi programeri počnu izradu programa bez odgovarajućih pripremnih aktivnosti, a i bez kasnijih odgovarajućih testiranja. Kao rezultat, često često dobijaju softverske sisteme koji rade nepouzdano, te moraju stalno da nalaze i otklanjaju greške u softveru, a to utiče na povećanje troškova razvoja i održavanja softvera. Međutim, te organizacije mnogo veće troškove imaju u oblasti primene softvera, gde kvarovi u softverskim sistemima dovode do prekida ili ometanja u njihovom svakodnevnom poslovanju (na primer, zaustavljanje proizvodnje ili montaže proizvoda). Troškovi posledica do kojih dovode zastoji u softverskim sistemima su mnogo veći, nego troškovi otklanjanja tih grešaka u samom softveru. Očigledno, nedostaje veći stepen obrazovanja iz softverskog inženjerstva i obuke ljudi koji razvijaju softverske sisteme da bi se izbegao problem razvoja skupih (zbog stalnih prepravki i dorada) , a nepouzdanih softverskih sistema (jer često ulaze u neočekivane zastoje, zbog grešaka u softvreu, tzv. bagove).

1.1 Slučaj softvera za upravljanje insulin pumpom

OPIS PROBLEMA

Softver je ugrađen sistem koji kontroliše rad pumpe sa insulinom, tako što prikuplja informacije sa senzora i kontroliše pumpu koje pacijentu daje kontrolisane doze insulina.

Ljudi koji imaju šećernu bolest moraju u određenim intervalima da primaju određenu dozu insulina, koji stimuliše rad pankreasa, koiji ne proiyvodi dovoljno hormon koji se nayiva insulin. Razvijen je uređaj koji treba da prati stanje pacijenta, i da u određenim trenucima ubacuje insulin pacijentu u venu. Taj uređaj ovde nazivamo insulin pumpom. Znači, **insulin pumpa** je medicinski sistem koji stimuliše rad pankreasa.

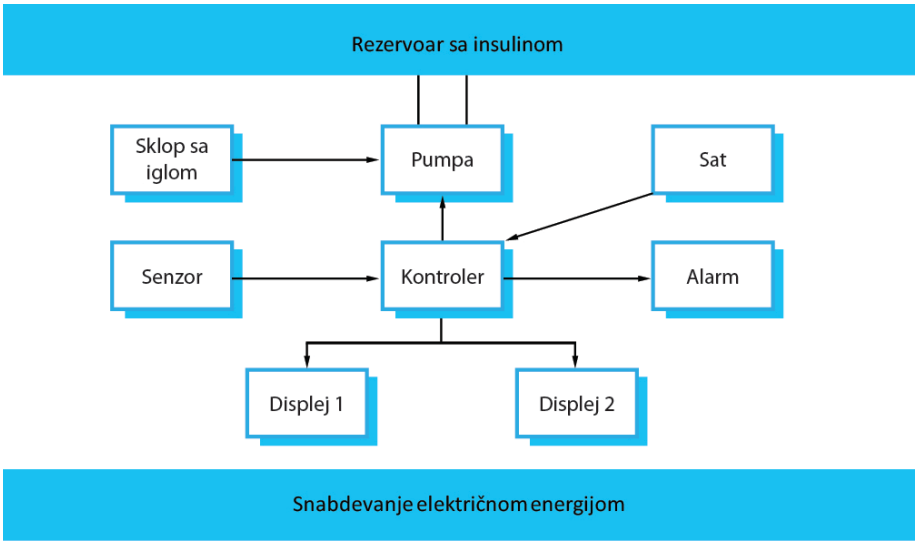
Softver koji kontroliše taj sistem je jedan ugrađen sistem, koji prikuplja informacije sa senzora i kontroliše pumpu koje pacijentu daje kontrolisane doze insulina. **Korisnici sistema** su dijabetičari, čiji pankreas ne proizvodi dovoljno hormona koji se naziva insulin. On vrši metabolizu glukoze (šećera) u krvi. Klasična terapija dijabetičara traži od njih da regularno uzimaju inekcije insulina kako bi regulisali sadržaj glukoze u krvi. Tačno doziranje insulina (periodika davanja i količina) je vrlo bitna kako se ne bi dogodilo da u krvi pacijenta bude previše ili premalo glukoze, jer dolazi do oštećenja tkiva i funkcija organa. Mali sadržaj glukoze u krvi može dovesti do oštećenja u mozgu i smrti pacijenta.
Sistem koji se ovde razmatra treba da prati stanje pacijenta i da određuje tačne i optimalne doze insulina pacijenta, kao i vreme davanja tih doza.
Pacijent prima insulin preko igle koja je povezana sa insulin pumpom. Sistem, preko senzora,

prati sadržaj šećera u krvi i povremeno daje pacijentu određenu dozu insulina. Sada, ovaj sistem radi samo u bolničkim uslovima, a u budućnosti, dijabetičari će moći da ga koriste i van bolnica, jer će biti stalno povezani sa mobilnom verzijom sistema.

OPIS RADA SOFTVERSKOG SISTEMA

Svaki impuls poslat pumpi, prouzrokuje davanje jedne jedinice insulina

Signali (o električnoj provodiljivosti krvi) iz senzora dolaze do kontrolera rada insulin pumpe. Kontroler, na bazi informacija sa senzora, računa nivo šećera u krvi i računa potrebnu dozu insulina. Onda šalje signal maloj pumpi koja isporučuje insulin preko stalno povezane igle sa venom pacijenta. Slika 1 pokazuje hardverske komponente i organizaciju insulin pumpe.

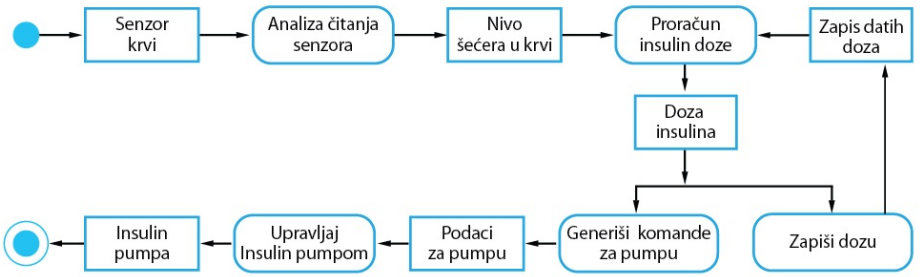


Slika 1.1.1 Hardver insulin pumpe

Svaki impuls poslat pumpi, prouzrokuje davanje jedne jedinice insulina. Veličina doze se, znači, određuje brojem poslatih impulsa pumpi. Na slici 2 prikazan je UML dijagram aktivnosti sistema. Očigledno, ovde se radi o bezbednosno-kritičnom sistemu. Ako pumpa ne radi dobro, može da dođe do oštećenja zdravlja pacijenta, te može doći do kome. Zbog toga, sistem mora da zadovolji dva bitna zahteva:

1. **Raspoloživost:** Sistem treba da obezbedi davanje insulina pacijentu uvek kada mi je on potreban.
2. **Pouzdanost:** Sistem mora da radi pouzdano i da isporučuje tačne doze insulina zavisno od trenutnog nivoa šećera u krvi.

Sistem treba tako projektovati i razviti da se zadovolje ovi uslovi.



Slika 1.1.2 Model aktivnosti insulin pumpe

ZADATAK ZA VEŽBU

Utvrđivanje bitnih faktora koji obezbeđuju pouzdan i stalan rad sistema

1. Da bi sistem radio pouzdano, mora da obezbedi da pacijent na vreme dobije odgovarajuću dozu insulina. Da bi to ostvario ovaj zahtev, koji podaatak sistem mora da ima o pacijentu? Kako sistem dolazi do tog podatka?
2. Kako bi trebalo da sistem obezbedi raspoloživost, tj. da obezbedi odgovarajuću dozu insulina uvek kada mu je to potrebno? Koji element sistema je najvažniji i bez njegovog rada, sistem ne bi mogao da radi?

Svoje odgovore dostavite svom instruktoru mejlom.

1.2 Slučaj softvera MHC-PMS sistema

OPIS SLUČAJA - MHC-PMS SISTEM

MHC-PMS ima dva opšta cilja: ocenjivanje performansi u pružanju zdravstvenih usluga i obezbeđenje informacija o pacijentima zdravstvenom osoblju.

Mnogi pacijenti sa mentalnim zdravstvenim problemima leče se ambulantno. Lekari im zakazuju periodične preglede na kojima im prepisuju odgovarajuću terapiju. U mnogim mestima postoje centri sa ambulantama za ove pacijente. Da bi se bolje pratila istorija bolesti svakog pacijenta, korisno je koristiti namenski informacioni sistem koji bi lekaru davao sve informacije o istoriji bolesti pacijenta koga prima na pregled.

Informacioni sistem o pacijentima sa mentalnim problemom (Mental Health Care – Patient Management System, ili kraće, MHC-PMS) koristi centralizovanu bazu sa informacijama o pacijentima. Korisnici sistema su distribuirani centri sa ordinacijama za pregled pacijenata sa mentalnim problemima. Centri koriste PC računarepreko kojih mogu da pristupe centralnoj bazi podataka na nivou zemlje. Centri mogu i da prebace bazu podataka na svoj računar, da ne bi koristili prenos podataka preko mreže. To rade centri koji nemaju mogućnost korišćenja zaštićene mreže. Ako imaju mogućnost korišćenja zaštićene mreže, oni onda mogu

koristiti udaljenu centralnu bazu, a prekopirati i lokalno koristiti i lokalne kopije podataka o pacijentima, ali samo sa informacijama vezanim za mentalne probleme.

Sistem MHC.PMS ima dva opšta cilja:

1. **Generisanje** informacija koje omogućavaju menadžerima zdravstvenih usluga da ocenjuju performanse usloga svojih centara u odnosu na postavljene lokalne i vladine ciljeve
2. **Obezbeđivanje**, po potrebi, ydravstvenom osoblju informacije koje su im neophodne za lečenje pacijenata.

KLJUČNA SVOJSTVA MHC-PMS SISTEMA

Ključna svojstva sistema treba da budu: upravljanje negom pacijenata, osmatranje pacijenata i priprema i distribucija administrativnih izveštaja.

Ključna svojstva sistema su:

1. *Upravljanje individualnom negom pacijenta*: Lekar kreira elektronski dosije za svakog pacijenta, vrši promene na sistemu, pregledava istorijat bolesti pacijenta itd. Sistem daje skraćene izveštaje o pacijentu koga evidentira, kako bi videli istorijat njegove bolesti i propisanih terapija.
2. *Osmatranje pacijenata:* Sistem, regularno osmatra bazu podataka o pacijentu i generiše upozorenja ako utvrdi neke probleme, kao na primer, ne pojavljivanje pacijenta na zakazane preglede. Posebno se prate bolesnici koji mogu biti opasni po okolinu.
3. *Administrativni izveštaji:* Sistem generiše mesečne izveštaje o broju lečenih pacijenata na klinici u toku jednog meseca, o novim pacijentima, o pacijentima koji se više ne leče, o izolovanim pacijentima, o prepisanim lekovima i troškovima, itd.

Sistem radi u saglasnosti sa dva zakona:

Jedan se bavi **zaštitom podataka o ličnosti**, a drugi o **zdravstvenoj nezi i izolaciji pacijenata opasnim po okolinu**, jer se ovi pacijenti mogu zadržati u bolnici i mimo njihove volje. Kao i kod drugih zdravstvenih sistema, i ovaj mora da vodi računa o privatnosti pacijenata, o zaštiti podataka o njihovoj bolesti i dr.

Projektno rešenje sistema treba da zadovolji zahteve koji se odnose na privatnost i sigurnost.

Kako se, pored centralne baze, na pojedinim klinikama (centrima) koriste i lokalne kopije baza (koje treba regularno sinhronizovati), što je i mera zaštite od prekida veza sa centralnom bazom podataka.

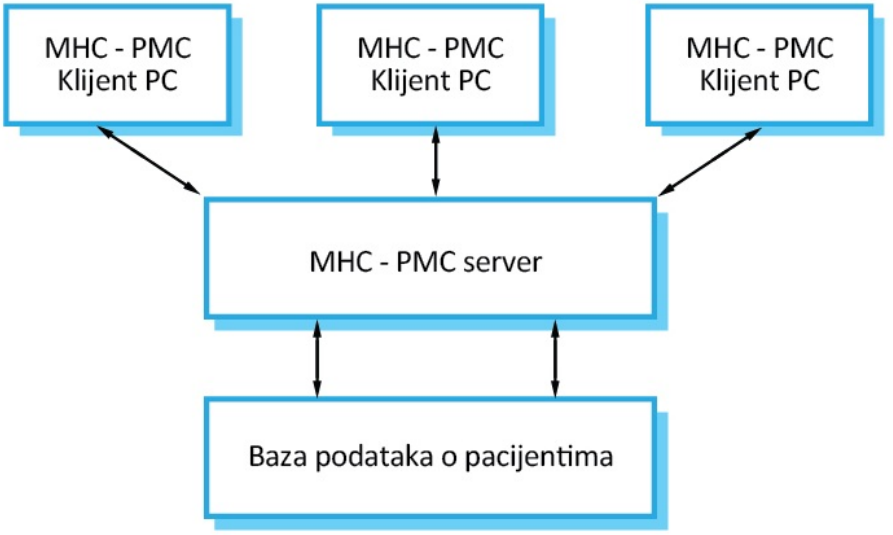
ORGANIZACIJA MHC-PMS SISTEMA

Sistem je baziran na klijent-server arhitekturi softverskog sistema.

Zbog prirode bolesti, pacijenti su često dezorganizovani, te mogu da ispuste zakazane termine za pregled, izgube recepte i lekove, zaborave dobijene instrukcije ili postavljaju nerazumne zahteve zdravstvenom osoblju. Mnogi često menjaju adrese stanovanja ili su beskućnici. Neki mogu biti i opasni za okolinu, te se zbog toga, zadržavaju u osiguranim bolnicama rad osmatranja i nadzora.

Korisnici sistema su zaposleni u klinikama (centrima) – doktori, medicinske sestre i osoblje kućne nege, a od nemedicinskog osoblja – recepcioneri (koji zakazuju termine za preglede), administratori (čuvaju zdravstvene kartone pacijenata, odnosno, brinu o bazi podataka o njima), i administrativni službenici koji pripremaju razne izveštaje.

Sistem čuva informacije o pacijentima (ime, adresa, godine starosti i dr.), o pregledima (datum, doktor, procena stanja pacijenta itd.), o uslovima i terapijama. Izveštaji se pripremaju regularno za medicinsko osoblje i za zdravstvene menadžere i vlasti. Slika 1 pokazuje organizaciju informacionog sistema MHC-PMS.



Slika 1.2.1 Organizacija PHC-PMC sistema

ZADACI ZA VEŽBU

Razumevanje cilja i svrhe rada MHC-PMC informacionog sistema

1. Navedite koje bi informacije o svakom pacijentu, a namenjene medicinskom osoblju, očekivali da daje informacioni sistem? Napišite u MS Word-u ili MS Power Point-u, sa buletima, sve informcije koje mislite da su potrebne lekarima i drugom zdravstvenom osoblju.
2. Navedite sve informacije koje bi po vašem mišljenju, trebalo da pruža informacioni sistem menadžmentu sistema ambulanti,da bi ono moglo da ocenjuje uspešnost njihovog rada.

Svoje odgovore pošaljite vašem instruktoru mejlom..

1.3 Slučaj udaljenih meteoroloških stanica

OPIS SLUČAJA - UDALJENE STANICE ZA PRIKUPLJANJE METEREOLOŠKIH PODATAKA

Sistem meri meteorološke podatke, prikuplja ih sa svih stanica, vrši lokalnu obradu i prenosi obrađene podatke u sistem za prikupljanje podataka od stanica.

Sistem povezuje nekoliko stotina udaljenih stanica za prikupljanje podataka iz instrumenata koji mere temperaturu, pritisak, osunčanost, padavine, brzinu vetra i smer duvanja vetra. Stanice su deo većeg sistema za obezbeđenje informacija o vremenskim prilikama. Prikupljene informacije se predaju drugim sistemima na dalju obradu. Svaka stanica ima instrumente za merenje sledećih veličina:

- Merenje brzine i smera vetra
- Merenje temperatura vazduha i tla
- Merenje vazdušnog pritiska
- Merenje nivoa padavina u poslednjih 24 sata

Svaki od instrumenata je kontrolisan softverskim sistemom koji periodično uzima podatke o izmerenim veličinama

Pored merenja, lokalni sistem u meteorološkim stanicama vrši i jedan deo obrade podataka, i to:

.

- Prikuplja podatke o izmerenim veličinama u zadatim intervalima (temperatura se meri svakog minuta)
- Vrši lokalnu obradu i agregaciju podataka
- Prenosi obrađene podatke sistemu za prikupljanje podataka od stanica
- Ako su veze u prekidu, lokalni sistem čuva podatke do uspostavljanja normalne veze. Stanice koriste baterije koje puni lokalni solarni sistem ili sistem sa vetrenjačom

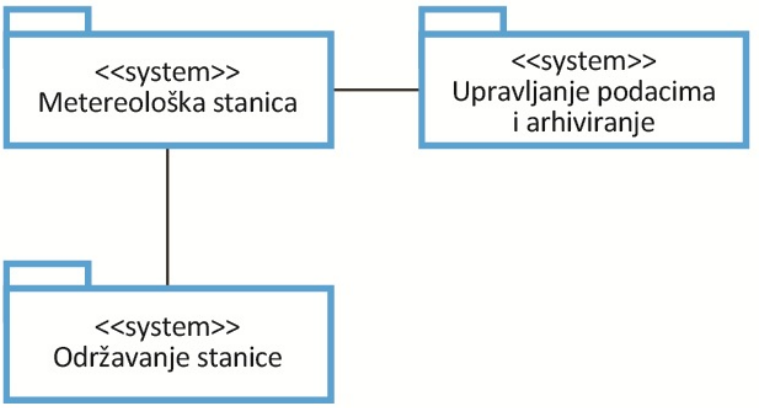
TRAŽENA SVOJSTVA SISTEMA ZA UPRAVLJANJE STANICAMA

Pored prikupljana podataka sistem prati rad instrumenata, upravlja energetskim napajanjem i podržava dinamičko rekonfigurisanje delova softvera.

Pored prikupljanja podataka, sotver radi i sledeće:

1. Prati rad instrumenata, hardver za napaja energijom i za komunikaciju i izveštava o primećenim greškama
2. Upravlja energetskim napajanjem sistema
3. Podržava dinamičko rekonfigurisanje delova softvera, zamena sa novim verzijama i obezbeđuje kopije podataka za obezbeđenje ako dođe do pada sistema ili grešaka u njegovom radu

.



Slika 1.3.1 Arhitektura sistema za prikupljanje podataka

ZADATAK ZA VEŽBU

Utvrđivanje funkcionalnosti sistema

Nabrojite funkcije koje sistem za upravljanje radom udaljenih stanica za prikupljanje udaljenih meteoroloških podataka treba da ima.

Poglavlje 2

Razvoj profesionalnog softvera

PROFESIONALNI SOFTVER I SOFTVERSKO INŽENJERSTVO

Profesionalni softver je softver koji ima određenu poslovnu svrhu. Softversko inženjerstvo je inženjerska disciplina koja se bavi svim aspektima proizvodnje softvera

Profesionalni softver je softver koji ima određenu poslovnu svrhu. On može biti ugrađen u različite uređaje (npr. mobilni telefoni, avion) ili može biti poseban softverski proizvod, kao što su informacioni sistemi, CAD sistemi i dr. Korisnik profesionalnog softvera nije onaj koji ga je razvio, već onaj koji ga koristi za obavljanje svog posla. Profesionalni softver ne razvija pojedinca, već tim softverskih inženjera, tj. inženjera za razvoj softvera.

Softversko inženjerstvo je inženjerska disciplina koja se bavi svim aspektima proizvodnje softvera. Softversko inženjerstvo podržava timski, a ne individualni razvoj profesionalnog softvera i uključuje tehnike koje podržavaju specifikaciju programa, njegovo projektovanje i evoluciju. U Tabeli 1 dato je objašnjenje pojmova koje se najčešće koriste i softverskom inženjerstvu [1].

.

Pitanje	Odgovor
Šta je softver?	Računarski programi i njihova dokumentacija. Softverski proizvodi mogu biti razvijeni za posebnog kupca ili za tržište.
Šta su atributi dobrog softvera?	Dobar softver treba da ostvari korisniku zahtevani funkcionalnost i performanse , i trebalo bi da bude što lakši za održavanje, pouzdan i koristan.
Šta je softversko inženjerstvo?	Softversko inženjerstvo je inženjerska disciplina koje se bavi svim aspektima proizvodnje softvera.
Koje su osnovne aktivnosti softverskog inženjerstva?	Specifikacija softvera, razvoj softvera, validacija softvera i evolucija softvera
Koja je razlika između softverskog inženjerstva i računarske nauke?	Računarska nauka se fokusira na teoriju i teorijske osnove; softversko inženjerstvo se bavi praksom razvoja i isporuke korisnog softvera.
Koja je razlika između softverskog inženjerstva i sistemskog inženjerstva?	Sistemsko inženjerstvo se bavi svim aspektima razvoja sistema oslonjenih na primenu računara, tj. računarskih sistema, uključujući hardver, softver i inženjerstvo procesa. Softversko inženjerstvo je deo ovog opštijeg procesa.
Koji su ključni izazovi sa kojima se suočava softversko inženjerstvo?	Rad sa sve većim različitostima, zahtevi za skraćivanje rokova isporuke, razvoj pouzdanog softvera.
Koji su troškovi primene softverskog inženjerstva?	Oko 60% troškova čine troškovi njegovog razvoja, a 40% troškovi testiranja. Za softver razvijen po narudžbini, troškovi evolucije često nadmašuju troškove razvoja .
Koje su najbolje tehnike i metode softverskog inženjerstva?	Iako su svi softverski projekti profesionalno vođeni, a softver profesionalno razvijen, koriste se različite tehnike razvoja, zavisno od vrste softvera. Na primer, igre se uvek razvijaju korišćenjem više prototipova, dok kritični sistemi upravljanja sistemima zahtevaju razvoj kompletne i analizirane specifikacije. Ne može se reći da je jedan metod bolji od drugog.
Koje su posledice po softverski inženjerstvo imao Internet?	Internet (ili Veb) je doveo do korišćenja softverskih servisa i pružio je mogućnost razvoja distribuiranih sistema baziranih na primeni servisa. Razvoj ovih, tzv. veb sistema, doveo je do značajnog napretka programskih jezika i u višestrukom korišćenju softvera.

SOFTVERSKO INŽENJERSTVO

Softversko inženjerstvo je inženjerska disciplina koja se bavi svim aspektima proizvodnje softvera

Zato što je softver svuda oko nas, softversko inženjerstvo je bitno za funkcionisanje društva, kako na nacionalnom, tako i na internacionalnom nivou. Zbog vrlo širokog opsega i veličine i složenosti softverskih sistema, prosto je nemoguće uspostavite univerzalni sistem označavanja, metoda ili tehnika softverskog inženjerstva, jer različite vrste softvera zahtevaju različite pristupe u njihovom razvoju i proizvodnji. Proizvodnja različitih vrsta softverskih sistema zahteva korišćenje različitih tehnika softverskog inženjerstva.

Svako od nas je više puta imao problema sa svojim računarom, tj. nekim softverom u njemu. Sistem se ukočio, prestao da reaguje na naše komande. U praksi, na žalost, često dolazi do vidljivih manifestacija grešaka u softverskim sistemima. Softversko inženjerstvo je mlada inženjerska disciplina koja još nije uspela da dostigne zrelost drugih inženjerskih disciplina (industrija automobila, građevinarstva i dr.) u kojima su vrlo retki slučajevi da njihovi proizvodi imaju „ugrađene greške“. Kada bi automobili i avioni bili razvijeni sa takvim procentom grešaka, kao što je čest slučaj u softverskom inženjerstvu, verovatno svakog dana bi bili svedoci mnogih padova avion, zaustavljanja automobila zbog kvarova i sl. Očigledno, softversko inženjerstvo još nije dostiglo nivo razvijenosti, tj. zrelosti, koji bi omogućio proizvodnju softvera na onom nivou pouzdanosti kao što je to slučaj u drugim inženjerskim disciplinama. Zašto je to tako?

PROFESIONALNI SOFTVERSKI SISTEMI

Profesionalni softverski sistemi se moraju razvijati primenom svih pravila, metoda i tehnika softverskog inženjerstva

Profesionalni softverski sistemi se moraju razvijati primenom svih pravila, metoda i tehnika softverskog inženjerstva. Znati programirati i na osnovu toga, krenuti u razvoj softvera, vodi ka razvoju računarskih programa koji ne mogu biti i softverski proizvodi, jer najčešće nemaju neophodnu pouzdanost. I ne samo to, često nemaju one karakteristike koje se od njih očekuju, a i ne zadovoljavaju zahteve korisnika. Programiranje je samo jedan mali dao softverskog inženjerstva. Ako se negde razvoj softvera svede samo na aktivnosti programiranja, onda je normalno da dođe do nezadovoljstva kod korisnika tih programa, jer ne zadovoljavaju očekivanja svojih korisnika i ne obezbeđuju neophodan nivo pouzdanosti. Zbog toga, važno je da svi koji se bave programiranjem, shvate da im je, pored programiranja, potrebno da znaju i mnoge druge stvari, da bi primenom tog znanja, uspeli da naprave sobar softverski proizvod, a to znači softver koji zadovoljava sve zahteve i očekivanja korisnika tog softvera, i sa aspekta funkcionalnosti i sa aspekta pouzdanosti.

Bez primene dobrog softverskog inženjerstva, nema ni razvoja i proizvodnje dobrog softvera.

ŠTA ČINI SOFTVER?

Softver čine i program, ali i njegova dokumentacija, kao i podaci o konfigurisanju softvera koji omogućavaju da ovi program normalno rade.

Softver nije isto što i program. Softver čine i program, ali i njegova dokumentacija, kao i podaci o konfigurisanju softvera koji omogućavaju da ovi program normalno rade. Kako profesionalni softver najčešće koristi više programa, neophodno je da postoji informacija o konfiguraciji softvera, tj. o programima koje treba pozivati i izvršavati tako da softver ostvari svoju funkciju. Zato mora da postoji dokumentacija o sistemu koja opisuje strukturu softverskog sistema, korisničku dokumentaciju, koja objašnjava kako se koristi sistem, kao i informacija o veb sajtovima sa kojih korisnik može da preuzme najnovije informacije o proizvodu.

To je osnovna razlika između razvoja profesionalnog softvera i razvoja amaterskog softvera. Kako se profesionalni softver koriste ljudi koji nisu razvijali softver, to je neophodno da se on dokumentuje tako da može biti pravilno korišćen.

Postoje dve vrste softverskih proizvoda:

1. **Generički proizvodi:** To su softverski proizvodi koji su razvijeni radi prodaje na tržištu. Mogu ih koristiti različiti korisnici koji nisu poznati u vreme razvoja. Organizacija koja razvija softver definiše programsku specifikaciju. Primeri: obrađivači tekstova, sistemi za upravljanje baza podataka, CAD sistemi.

2. **Proizvod razvijeni po narudžbini:** Firma koja razvija softver, to radi po zahtevu kupca, koji definiše specifikaciju za softver. Primeri: kontrolni sistemi elektronskih uređaja, sistemi za upravljanje određenim poslovnim procesima, sistemi za upravljanje saobraćajem.

U praksi, ova podela ne mora uvek da bude tako jasna. Na primer, ERP sistemi se prodaju kao softverski proizvodi. Međutim, da bi se uspešno primenili u određenoj organizaciji, moraju se prilagoditi njenim zahtevima, tj. njenim procesima.

KVALITET PROFESIONALNOG SOFTVERA

Kvalitetan profesionalni softver mora, pored funkcionalnih zahteva, da zadovolji i nefunkcionalne zahteve.

Kada se govori o kvalitetu profesionalnog softverskog proizvoda, ne govori se samo i njegovoj funkcionalnosti, već i o zadovoljenje i nefunkcionalnih zahteva, tj. o performansama, kao što su vremena odziva i razumljivost programskog koda. Ti nefunkcionalni zahtevi, koji određuju i njegov kvalitet, su različiti za različite vrste softvera. Na primer, kod bankarskih sistema, taj zahtev je sigurnost i bezbednost informacija. Kod interaktivnih igara – to je vreme odziva, a kod digitalnih telefonskih centrala – pouzdanost. U Tabeli 2 su prikazani atributi profesionalnog softverskog sistema.

Pitanje	Odgovor
Sposobnost održavanja	Softver treba biti tako napisan da može da se lako menja kako bi se zadovoljili zahtevi korisnika za njihovu promenu. Ovo je kritičan atribut, jer je promena softvera neminovni zahtev u poslovnim okruženjima koja se brzo menjaju.
Pouzdanost i sigurnost	Pouzdanost softvera obuhvata niz karakteristika, kao što su: pouzdanost, sigurnost i bezbednost. Pouzdanost softvera ne bi trebalo da prouzrokuje nikakvu fizičku ili ekonomsku štetu u slučaju pada sistema. Neovlašćena lica ne bi mogla da pristupe sistemu i da mu naprave štetu.
Efikasnost	Softver ne bi trebalo da beskorisno troši resurse sistema, kao što su memorija i ciklusi procesora. Efikasnost obuhvata: odziv sistema, vreme obrade, upotrebu memorije i dr.
Prihvatljivost	Softver mora biti prihvatljiv za kategorije korisnika za koje je projektovan. Ovo znači da mora da bude razumljiv, koristan i kompatibilan sa sistemima koje korisnici upotrebljavaju.

PRIMER

Profesionalni softver nije ne čine samo programi koji su razvijeni za naručioca

Objasnite zašto profesionalni softver nije ne čine samo programi koji su razvijeni za naručioca?
Dobar / profesionalni softver nije samo program koji se razvija za kupca, već se sastoji od izvršnog koda i povezan je sa dokumentacijom i konfiguracijom podataka koji su potrebni da bi ovi programi ispravno radili. Profesionalno razvijen softver za softver je često više

od jednog programa. Sistem se obično sastoji od više odvojenih programa i konfiguracionih datoteka koje se koriste za postavljanje ovih programa. Može da sadrži sistemsku dokumentaciju koja opisuje strukturu sistema; korisničku dokumentaciju, koja objašnjava kako da koristi sistem i veb stranice za korisnike da preuzmu najnovije informacije o proizvodu. Primer sistema za obradu teksta sastoji se od izvršnog programa, korisničkog priručnika i dokumenta kao što su zahtevi i dizajn potreban za izradu izvršnog programa

PITANJA ZA DISKUSIJU

Utvrđivanje specifičnosti profesionalnog softvera, kao i izazova s kojima se suočava softversko inženjerstvo.

Odgovorite na sledeća pitanja

1. Koja je najvažnije razlika između generičkog softverskog proizvoda i softvera razvijenog po zahtevu određenog kupva?
2. Koja su četiri važna atributa koja mora da poseduju svi profesionalni softveri?
3. Predložite još četiri atributa koji mogu da budu do izvesne mere značajni.

Svoje odgovore pošaljite instruktoru.

Poglavlje 3

Softversko inženjerstvo

ŠTA JE SOFTVERSKO INŽENJERSTVO?

Softversko inženjerstvo je inženjerska disciplina koja se bavi svim aspektima proizvodnje softvera, od ranih faza specifikacije sistema, do održavanja nakon početka njegove upotrebe

Softversko inženjerstvo je inženjerska disciplina koja se bavi svim aspektima proizvodnje softvera, od ranih faza specifikacije sistema, do održavanja nakon početka njegove upotrebe. To je softverska disciplina jer obezbeđuje rešavanje problema i u nedostatku svih potrebnih teorija i metoda, pri čemu se uzimaju u obzir i organizacijska i finansijska ograničenja. Softversko inženjerstvo se bavi svim aspektima proizvodnje softvera, jer se ne bavi samo procesima razvoja tehničkih procesa razvoja softvera, već uključuje i aktivnosti, kao što su upravljanje softverskim proizvodom, kao i alatima, metodima i teorijama koje podržavaju proizvodnju softvera.

Inženjerstvo se uvek bavi dobijanjem rezultata zahtevanog kvaliteta, u datim rokovima, a i u okviru planiranog budžeta. Ovo podrazumeva neophodnost pravljenja kompromisa. U inženjerstvu, glavna aktivnost se svodi na izbor metoda koji najviše odgovaraju u određenim okolnostima, tako da primeni kreativniji, manje formalan pristup razvoju koji je efektivniji u određenim okolnostima.

Softverski inženjeri primenjuju sistematski i organizovani pristup u svom poslu, jer je to najčešće i najefektniji način proizvodnje softvera visokog kvaliteta. Manje formalan razvoj je posebno prikladan pri razvoju veb sistema, koji zahtevaju mešavinu veštinu u razvoju softvera i grafičkog dizajna.

SOFTVERSKI PROCES

Softverski proces je redosled aktivnosti koje vodi proizvodnji softverskog proizvoda.

Softversko inženjerstvo je važno iz dva razloga:

1. Neophodnost ekonomičnog razvoja pouzdanog softvera za što kraće vreme.
2. Na duže staze, jeftinije je primeniti metodu softverskog inženjerstva nego samo pisati programe, kao što se to radi kod razvoja ličnog softvera, jer su najveći troškovi najčešće vezano za aktivnosti promene u softveru.

Sistemski pristup koji se koristi u softverskom inženjerstvu se često naziva i softverskim procesom. **Softverski proces** je redosled aktivnosti koje vode proizvodnji softverskog proizvoda. Softverski proces čine četiri osnovne aktivnosti, koje su prisutne u svim softverskim procesima:

1. **Specifikacija softvera -** kada kupci i inženjeri definišu softver koji treba da se proizvede, kao i ograničenja u njegovom radu.
2. **Razvoj softvera -** kada se vrši projektovanje softvera i programiranje**.**
3. **Validacija softvera -** kada se proverava da li softver zadovoljava sve ono što kupac zahteva**.**
4. **Evolucija softvera -** kada se vrši promena softvera u skladu sa promenjenim zahtevima kupca i tržišta**.**

Različite vrste sistema zahtevaju različite razvojne procese. Na primer, u slučaju sistema u realnom vremenu za upravljanje avionom, treba da bude specificiran u celosti pre nego što njegov razvoj počne. Kod sistema e-poslovanja vrši se istovremen razvoj specifikacije i programa.

IZAZOVI PRI RAZVOJU SOFTVERA

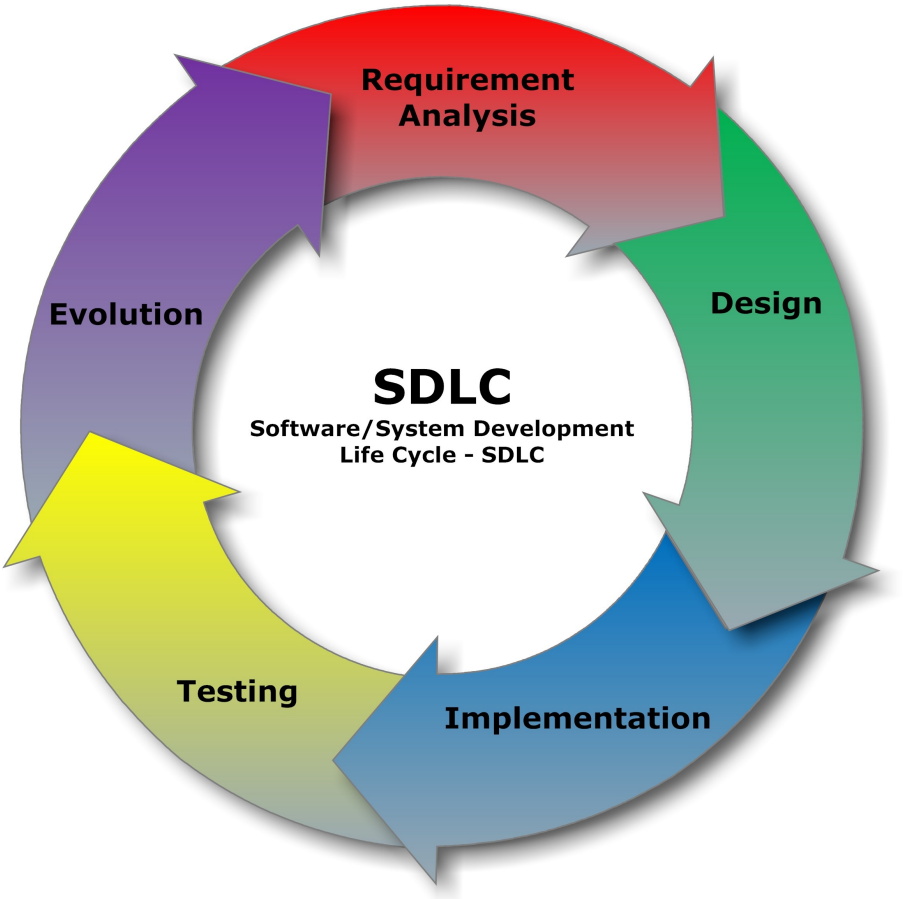
Izazovi koju su prisutni pri razvoju različitih vrsta softvera su: rad u heterogenom okruženju, brzo prilagođavanje novim zahtevima i bezbednost informacija.

Postoji puno različitih vrsta softvera. Nema univerzalnog metoda softverskog inženjerstva koji je primenljiv za sve te vrste. Ipak, postoje opšta pitanja koja su relevantna za mnoge različite vrste softvera:

1. **Heterogenost.** Sistemi rade kao distribuirani sistemi na mreži koja uključuje različite tipove kompjutera i mobilnih uređaja. Softver, pored rada u kompjuterima opšte namene, mora da radi i na mobilnim telefonima. Često se softver mora da integriše sa starim sistemima pisanim različitim programskim jezicima. Ovde je izazov da se razviju tehnike izrade pouzdanog softvera koji je dovoljno fleksibilan da radi u heterogenom okruženju.
2. **Poslovne i društvene promene:** Poslovanje i društvo se vrlo brzo menjaju sa razvojem novih ekonomija i kako nove tehnike postaju dostupne. Postojeći softveri moraju da se zamenjuju novim, koji mora da se brzo razvijaju. Mnoge tradicionalne tehnike softverskog inženjerstva traže puno vremena, te izrada novog softvera često potraje duže nego što je planirano. One moraju da se usavršavaju kako bi omogućile brži razvoj softvera koji obezbeđuje vrednost kupcu.
3. **Bezbednost i poverenje.** Da bi verovali softveru, moramo ga zaštiti od neovlašćenih lica i dbiti njihove napade radi zadržavanja bezbednosti informacija.

PRIMER

Profesionalni softver ne čine samo programi razvijeni za naručioca



Slika 3.1 Životni ciklus stvaranja profesionalnog softvera

PITANJA ZA DISKUSIJU

Utvrđivanje specifičnosti profesionalnog softvera, kao i izazova s kojima se suočava softversko inženjerstvo.

Odgovorite na sledeća pitanja

1. Objasnite, u jednoj rečenici, šta je softversko inženjerstvo?
2. Šta je softverski proces?
3. Navedite četiri glavne aktivnosti softverskog procesa.
4. Šta je heterogeno okruženje softvera? Zašto je radu u heterogenom okruženju izazov za razvoj softvrea?

Svoje odgovore pošaljite instruktoru.

Poglavlje 4

Različitost softverskog inženjerstva

VRSTE APLIKACIJA I SOFTVERSKO INŽENJERSTVO

Softversko inženjerstvo je sistematski pristup proizvodnji softvera koji uzima u obzir troškove, pitanja pouzdanosti i potrebe kupaca i proizvođača softvera.

Nema univerzalnih metoda softverskog inženjerstva koje odgovaraju svim sistemima i svim organizacijama**.** Izbor najpogodnije metode zavisi najviše od vrste aplikacija za koju se softver razvija:

1. **Samostalna aplikacija:** Ovi sistemi se izvršavaju izolovano na nekom računaru, npr. PC.
2. **Interaktivne transakcione aplikacije:** Aplikacija se izvršava na udaljenom računaru kojoj se pristupa sa korisničko PC računara ili terminala. To su najčešće veb aplikacije (za e-poslovanje), poslovni sistemi, i drugi sistemi kojima se pristupe preko veb prikazivača. Interaktivne aplikacije obično sadrže velike baze podataka koje se osvežavaju transakcijama koje iniciraju korisnici.
3. **Ugrađeni upravljački sistemi (Embedded Control Systems):** Ugrađeni sistemi (hardver i softver) upravljaju radom hardverskim uređajem. Ovo su najbrojniji računarski sistemi. Primeri: mobilni uređaji, televizori i dr.
4. **Sistemi sa paketnom obradom:** Ovo su poslovni sistemi projektovani da vrše masovnu obradu podataka. Na osnovu velikog broja pojedinačnih unosa podataka, kreiraju se odgovarajući izveštaji. Primeri: priprema računa za struju.
5. **Zabavni sistemi:** Razvijeni sa ličnu upotrebu s namerom da zabave korisnika (video igre). Najvažniji deo sistema je interakcija sa korisnikom.
6. **Sistemi za modelovanje i simulaciju:** Ovi sistemi razvijaju naučnici i inženjeri da bi modelovali fizičke procese ili situacije. Ovi sistemi rade intenzivne proračune i zahtevaju paralelne sisteme visokih performansi za svoje izvršenje**.**
7. **Sistemi za prikupljanje podataka:** Ovi sistemi prikupljaju podataka iz svog okruženja preko odgovarajućih senzora. Podaci se zatim obrađuju**.**
8. Sistemi sistema: **Ovo su sistemi koji čine veći broj drugih sistema.**

FUNDAMENTALNE METODE RAZVOJA SOFTVERA

Postoje fundamentalne metode razvoja softvera koje se primenjuju bez obzira na kategoriju i vrstu softvera koji se razvija.

Kako su softverski sistemi navedenih kategorija vrlo različiti, to se i za njihov razvoj koriste različite tehnike softverskog inženjerstva. Na primer, sistemi koji se ugrađuju u automobile se unose u ROM čip, te se vrlo teško kasnije menjaju. Zato pri njihovom razvoju, mora da se posebna pažnja posveti verifikaciji i validaciji sistema. Takođe, oni imaju nikakvu ili minimalnu interakciju sa korisnikom, te pri razvoju nije potrebno koristiti prototipove korisničkog interfejsa. Međutim, postoje neke fundamentalne metode koje se odnose na tipove softverskog sistema:

1. *Svi sistemi bi trebalo da se razvijaju procesom koji je upravljiv i razumljiv.* Mora se napraviti plan procesa razvoja koji treba da definiše ko, šta i kada treba da završi svoj deo posla na razvoju softvera. Za različite tipove softvera, koristi se različiti procesi.
2. *Pouzdanost i performanse su važna svojstva za sve tipove softvera.* Softver bi trebalo da se ponaša kao što se od njega i očekuje, bez grešaka, i treba biti upotrebljiv što pre. On mora da bude pouzdan i otporan na spoljne napade. Sistem mora da radi efikasno i ne bi trebalo da nepotrebno troši resurse računara.
3. *Razumevanje i upravljanje specifikacijom softvera i zahtevima*. Treba da znate šta različiti kupci i korisnici sistema očekuju od njega, i onda treba da takav sistem razvijete, a u okviru planiranog budžeta i roka isporuke.
4. *Neophodno je efektivno koristiti postojeće resurse*. To se postiže višestrukom upotrebom softverskih komponenti, umesto da se razvija novi softver.

Ovo su osnovni pristupi definisanja procesa, kojima se obezbeđuje pouzdanost, specifikacija zahteva, upravljanje i ponovna upotreba softvera. Oni su osnova svih metoda razvoja profesionalnog softvera. Oni ne pokrivaju implementaciju i programiranje. Ovde se ne bavimo specifičnim tehnikama programiranja, jer se one vrlo razlikuju u zavisnosti od tipa softvera.

PRIMER

Primeri različitih vrsta aplikacija

1. **Samostalna aplikacija:** Microsoft Word, Excel....
2. **Interaktivne transakcione aplikacije:** Email, Dropox...
3. **Ugrađeni upravljački sistemi (Embedded Control Systems):** Sistem kontole saobraćaja, Automatski upravljački sistem...
4. **Sistemi sa paketnom obradom:** Bankarski transakcioni sistem, Sistem praćenja inventara...
5. **Zabavni sistemi:** Igrice
6. **Sistemi za modelovanje i simulaciju:** Virtual reality and augmented reality sistemi
7. **Sistemi za prikupljanje podataka:** Crowdsorcing sistemi

PITANJA ZA DISKUSIJU

Utvrđivanje specifičnosti profesionalnog softvera, kao i izazova s kojima se suočava softversko inženjerstvo.

Odgovorite na sledeća pitanja:

1. Objasnite, sa primerima, zašto različiti tipovi aplikacija zahtevaju primenu različitih tipova softvera?
2. Navedite i objasnite fundamentalne ideje softverskog inženjerstva koje važe za sve tipove softverskih sistema.

Odgovore pošaljite instruktoru.

Poglavlje 5

Softversko inženjerstvo i veb

SPECIFIČNOSTI RAZVOJA VEB SISTEMA

Prebacivanje programa na veb servere, umesto u računarima korisnika, smanjuje troškove održavanja.

Veb sistemi su softverski sistemi koje korisnici koriste preko veb prikazivača. Pored toga, oni najčešće koriste i veb servise za obezbeđenje potrebne funkcionalnosti. Omogućavanje da pretraživači sadrže i izvršavaju male programe i da vrše lokalnu obradu, dovelo je do evolucije u razvoju poslovnih i organizacijskih softverskih sistema. Ovo je omogućilo, da se koristi softver na veb serveru, a ne na računaru korisnika. To je znatno pojeftinilo instalaciju i održavanje softvera, jer se to radi samo na serveru, a ne na računarima korisnika. Smanjeni su troškovi vezani za razvoj korisničkog interfejsa. Zbog ovih pogodnosti, svuda gde je moglo, poslovne aplikacije su prešle u veb sisteme.

Kao što je rečeno, pored upotrebe veb prikazivača, veb sistemi koriste i tzv. veb servise. **Veb servisi** su softverske komponente koje isporučuju specifičnu, korisničku funkcionalnost kojima se može pristupiti preko Interneta. Aplikacije se prave integracijom neophodnih veb servisa. Veb servise mogu da nude različite kompanije. Zbog dinamičkog povezivanja veb servisa, omogućava da se koriste različiti veb servisi kod svakog izvršenja aplikacije.

SPECIFIČNOSTI RAZVOJA SOFTVERA SA PRIMENOM VEB SERVISA I RAČUNARSKIH OBLAKA

Primena veb servisa i računarskih oblaka dovela je do promene i načina razvoja veb sistema: veća upotreba komponenti, inkrementalnog razvoja i GUI na veb prikazivačima.

Danas se mnogi veb servisi nude preko tzv. računarskih oblaka (cloud-based systems). Sada su softverski sistemi vrlo distribuirani i retko se razvijaju u celosti, već se koriste komponente. Sve ove izmene u razvoju softverskim sistemima dovelo je do pojave nuđenja “softvera kao servisa”, jer se softver ne izvršava na lokalnom kompjuteru, već u “računarskom oblaku”, tj. udaljenom jednom ili više računara kojima se pristupa preko Interneta. **Računarski oblak (computing cloud)** čini jedan veliki broj povezanih računarskih sistema koji opslužuju veliki broj korisnika. Korisnici ne kupuju softver, već plaćaju njegovo korišćenje, ili ga besplatno koriste jer se prikazuje zajedno sa reklamnim oglasima.

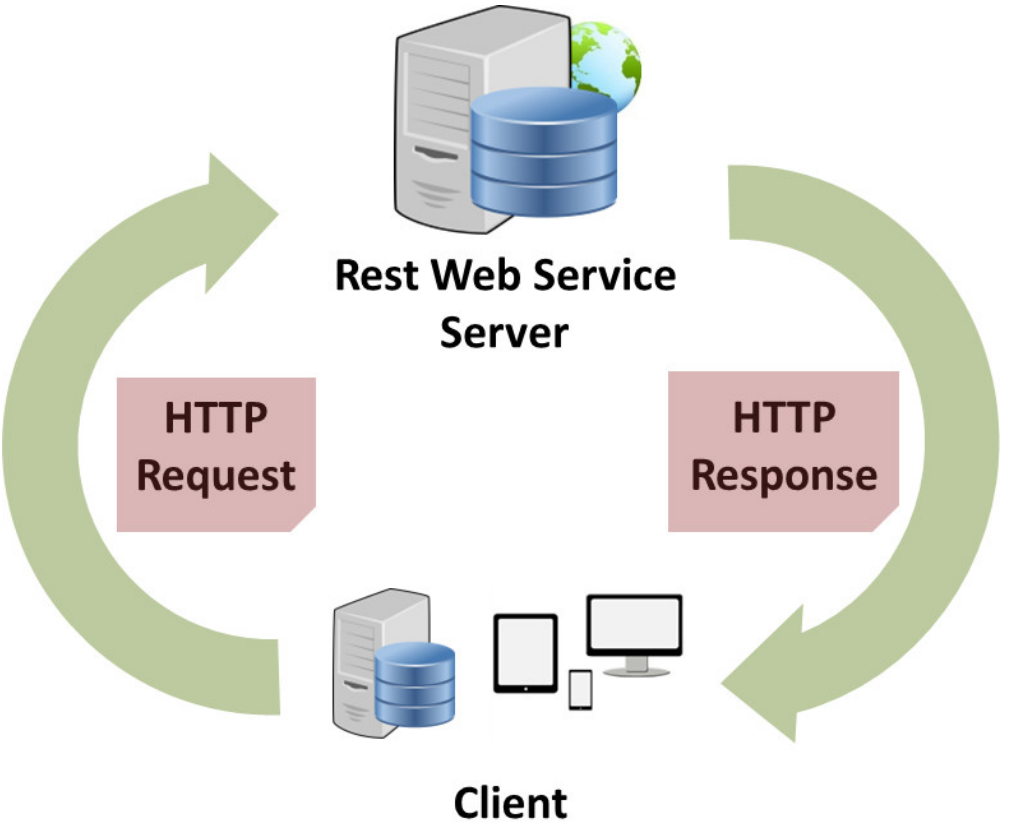
Radikalna promena u organizaciji softvera, dovela je i do promena u načina razvoja veb sistema. Na primer:

1. Ponovna upotreba ranije razvijenog softvera (software reuse) postao je dominantan način razvoja veb sistema. Razvoj se dominantno odvija izborom softverskih komponenta i njihovom integracijom u sistem.
2. Ustanovljeno je da nije praktično da se unapred definišu svi zahtevi. Veb sistemi se bolje razvijaju inkrementalnim razvojem i isporukom.
3. Korisnički interfejs se oslanja na mogućnosti veb prikazivača. Iako postoje tehnologije, kao AJAX, koje omogućuju korišćenje sadržajnog korisničkog interfejsa i korišćenjem veb prikazivača, njihovo korišćenje nije jednostavno. Najčešće se koriste veb forme definisane lokalnim skriptom. Zato su interfejsi veb sistema siromašniji u odnosu na specijalno projektovani korisnički interfejsi koji koriste softver koji se izvršava na klijentskom računaru (PC).

Navedene specifičnosti u razvoju veb sistema ne znače da se i kod razvoja veb sistema ne primenjuju fundamentalni principi softverskog inženjerstva, karakteristični za razvoj velikih sistema. Ti principi su primenljivi i kod veb sistema.

PRIMER

Primer arhitekture zasnovane na web servisima



Slika 5.1

PITANJA

Dati odgovore na postavljena pitanja

1. Objasnite kako je univerzalna upotreba Interneta promenila softverske sisteme.

Poglavlje 6

Etika softverskog inženjerstva

PRAVILA STRUKE

Softverski inženjeri moraju da se ponašaju na etički i moralno odgovoran način da bi mogli da uživaju ugled softverskog inženjera.

U mnogim inženjerskim disciplinama, postoje društvena i zakonska pravila koje inženjeri u tim disciplinama moraju da poštuju, zbog zaštite društva od eventualnih propusta i neprofesionalnog i neetičkog ponašanja inženjera. To isto važi i za softversko inženjerstvo, jer softver ima veliki značaj u svakodnevnom životu, te softverski inženjeri treba da prihvate svoju značajnu odgovornost. Oni moraju da se ponašaju na etički i moralno odgovoran način da bi mogli da uživaju ugled softverskog inženjera.

Ako ste softverski inženjer, očekuje se da primenjujete sve uobičajene standarde poštenja i integriteta ličnosti. Ne bi trebalo da upotrebite vaše znanje i veštine na nepošteni način koje mogu ugroziti reputaciju softverskog inženjerstva. Pored zakonskih ograničenja, postoje i pravila struke koje treba poštovati:

1. **Poverljivost:** Morate poštovati i čuvati poverljive informacije vašeg poslodavca i vaših klijenata, nezavisno od toga da li ste potpisali sporazum o poštovanju poslovne tajne.
2. **Kompetencija:** Ne smete da ističete kompetencije koje nemate. Ne smete prihatiti posao za koji niste kompetentni**.**
3. **Prava intelektualne svojine:** Morate biti upoznati sa zakonima koji regulišu pitanja intelektualne svojine, kao što je korišćenje patenata i autorska prava. Morate poštovati autorska prava poslodavca i vaših klijenata.
4. **Zloupotreba kompjutera:** Ne bi smeli da koristite vaše tehničke veštine i da zloupotrebite pristup računarima drugih. Bilo da se radi o trivijalnoj zloupotrebi (igranje igrica) pa do ozbiljnih dela (širenje virusa).

OSAM ETIČKIH PRINCIPA ACM I IEEE

Osam etičkih principa ACM i IEEE treba da odrede ponašanje svih inženjera u oblasti računarstva.

Profesionalne i etičke standarde ponašanja obično određuju njihova profesionalna udruženja. U SAD, u slučaju računarskih disciplina, to su organizacije: ACM i IEEE (Institute of Electrical and Electronic Engineers), a u Velikoj Britaniji, to je BCS (British Computer Society). Oni pri učlanjenju, traže od svojih novih članova da prvo potpišu izjavu da prihvataju etički i

profesionalni kod ponašanja koje su ove organizacije usvojile. Etički kod koji su zajednički usvojili ACM i IEEE predviđa **osam principa** koje treba da odrede ponašanja svih inženjera u oblasti računarstva:

1. **JAVNI INTEREST:** Softverski inženjeri treba da stalno deluju u skladu sa javnim interesom**.**
2. **KLIJENT I POSLODAVAC:** Softverski inženjeri treba da deluju na način koji je u najboljem interesu njihovih klijenata i poslodavca, a koji su u saglasju sa javnim interesom.
3. **PROIZVOD:** Softverski inženjeri obezbeđuju da njihovi proizvodi i povezane modifikacije ostvaruju najviše moguće profesionalne standarde.
4. **RASUĐIVANJE:** Softverski inženjeri treba da održe integritet i nezavisnost u svom profesionalnom rasuđivanju.
5. **MENADŽMENT:** Menadžeri i lideri u softverskom inženjerstvu treba da se izjasne o poštovanju i promociji etičkih pristupa u menadžmentu u razvoju softvera i u održavanju.
6. **PROFESIJA:** Softverski inženjeri će unapređivati integritet i reputaciju profesije u saglasnosti javnim interesom.
7. **KOLEGE:** Softverski inženjeri treba da bude fer prema svojim kolegama i da ih podržavaju.
8. **LIČNO:** Softverski inženjeri treba da učestvuju u celoživotnom učenju iz prakse svoje profesije i treba da promovišu etički pristup u sprovođenju svoje profesije

MORALNE, ETIČKE I PROFESIONALNE DILEME

Držeći se osam etičkih principa, softverski inženjeri treba da deluju i kada su izloženi situacijama koje ih dovode u moralne, etičke i profesionalne dileme.

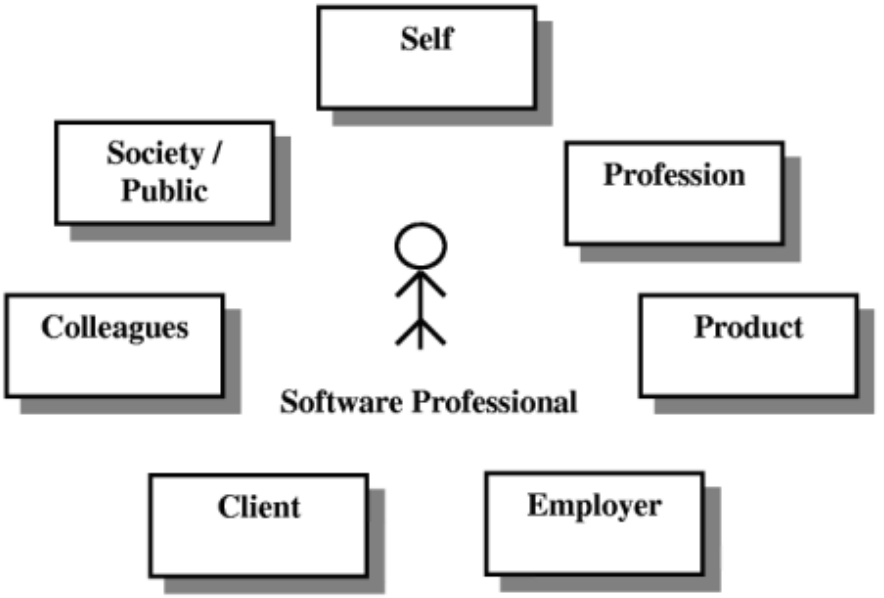
Držeći se osam etičkih principa, softverski inženjeri treba da deluju i kada su izloženi situacijama koje ih dovode u moralne, etičke i profesionalne dileme, kao na primer:

- Ako se ne slažete sa poslovnom politikom rukovodstva vaše firme, da li treba da reagujete ?
- Ako ste svesni problema u vašem softverskom projektu, da li sa tim treba da upoznate rukovodstvo firme ?
- Da li je odgovornost inženjera da zadrži poslovnu informaciju za sebe ili da upozori klijenta ili javnost, ako je sistem koji se isporučuje, nebezbedan?

Svako mora sam da zaključi kako treba da reaguje u ovim i sličnim situacijama. Sigurno je da treba voditi računa o šteti koju, bilo kakva odluka, može da nanese ljudima ili organizacijama.

PRIMER

Softverski inženjeri su u obavezi da deluju etički i kada su izloženi situacijama koje ih dovode u moralne, etičke i profesionalne dileme.



Slika 6.1 Etički principi ACM i IEEE

PITANJA ZA DISKUSIJU

Utvrđivanje specifičnosti profesionalnog softvera, kao i izazova s kojima se suočava softversko inženjerstvo.

Odgovorite na sledeća pitanja

1. Dajte po jedan primer za svaku klauzulu ACM/IEEE etičkog kodeksa. .

Poglavlje 7

Zaključak

REZIME

Softversko inženjerstvo je inženjerska disciplina koja se bavi svim aspektima proizvodnje softvera

Iz svega izloženog u ovoj lekciji, može se uočiti sledeće:

1. Softversko inženjerstvo je inženjerska disciplina koja se bavi svim aspektima proizvodnje softvera.
2. Softver nije samo program, već program sa dokumentacijom. Bitni atributi softverskog proizvoda su: sposobnost održavanja, pouzdanost, bezbednost, efikasnost i prihvatljivost.
3. Softverski proces uključuje sve aktivnosti razvoja softvera. Opšte aktivnosti specifikacije, razvoja, validacije i evolucije su deo svih softverskih procesa.
4. Fundamentalna stanovišta softverskog inženjerstva su univerzalno primenljiva za sve vrste razvoja sistema. Ova fundamentalna stanovišta uključuju softverske procese, pouzdanost, bezbednost, zahteve i ponovnu upotrebljivost.
5. Ima puno različitih vrsta softverski sistema i svaki od njih zahteva odgovarajući alat softverskog inženjerstva i tehnike za njihov razvoj. Ipak, ima samo nekoliko specifičnih tehnika projektovanja i implementacije koje se primenjuju za sve vrste sistema.
6. Softverski inženjeri su odgovorni za inženjersku profesiju u njen položaj u društvu. On ne treba da se samo bave tehničkim pitanjima.
   Profesionalna udruženja objavljuju kodove ponašanja koji definišu standarde ponašanja koja očekuju od svojih članova.