UNIVERZITET METROPOLITAN BEOGRAD

SE311 - PROJEKTOVANJE I ARHITEKTURA SOFTVERA

Arhitektonske strukture, pogledi i stilovi

Lekcija 02

PRIRUČNIK ZA STUDENTE

SE311 - PROJEKTOVANJE I ARHITEKTURA SOFTVERA

Lekcija 02

ARHITEKTONSKE STRUKTURE, POGLEDI I STILOVI

UVOD

Softverska arhitektura predstavlja važnu poddisciplinu softverskog inženjerstva.

Arhitektura, grubo rečeno, predstavlja podelu celine na delove sa specifičnim odnosima između delova. Ovakva podela omogućava grupama ljudi (razvojnim timovima) u razdvojenim organizacijama (geografski i vremenski) da rade zajedno i produktivno u cilju rešavanja mnogo većeg problema nego što bi svako od njih individualno mogao da reši.

Svaka grupa razvojnog tima može razvijati modul softvera koji je u interakciji sa drugim modulima softvera drugih grupa. Kroz interakciju moguće je odrediti funkcionalnost i atribute kvaliteta kao što su sigurnost, izmenljivost, perfomanse koji su zahtevani od strane naručioca softvera. Veći i složeniji sistemi zahtevaju veću podelu i procenu arhitekture. Što su atributi kvaliteta zahtevniji to je i arhitektura softvera kritičnija.

Svaki sistem je moguće predstaviti na nekoliko različitih načina. Podela sistema rezultirana je od strane strukture same softverske arhitekture kao i različitih elemenata i veza između njih. Svaki element softverske arhitekture predstavlja rezultat pažljivog i preciznog projektovanja u cilju ispunjavanja zahteve za kvalitet softvera a i zahteve nametnute od strane logike poslovanja organizacije za koju se softver projektuje. Softverska arhitektura omogućava da setovi delova sistema rade spojeno i uspešno u jednoj funkcionalnoj celini.

Dokumentacija o softverskoj arhitekturi omogućava projektantima da donesu prave odluke a i implementerima softvera (programerima) da na pravi način primene i razviju sve zahteve naručioca softvera.

Ova lekcija vam obezbeđuje sledeće ishode učenja:

- Definisanje i razumevanje različite softverske arhitekture
- Pregled dokumentacije softverske arhitekture, definisanje krajnjih korisnika softverske arhitekture i atributa kvaliteta
- Razumevanje pogleda na softversku arhitekturu, primenu stilova softverske arhitekture i primenu skupa tehnika u cilju detaljnog predstavljanja svakog dela softverske arhitekture
- Razumevanje i primena sedam pravila za pisanje softverske arhitekture

UVODNI VIDEO LEKCIJE

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

Poglavlje 1

Softverska arhitektura i atributi kvaliteta

ARHITEKTURA I ATRIBUTI KVALITETA

Arhitektura sistema se projektuje kako bi ispunila zahteve naručilaca softvera i zahteve projektanta softvera.

Za skoro sve sisteme, atributi kvaliteta kao što su performanse, pouzdanost, bezbednost i izmenljivost su jako bitni u smislu omogućavanja pravog odgovora na zadatke korisnika. Sposobnost softverskih sistema da proizvede tačan rezultat nije od pomoći korisniku ukoliko sistemu treba dugo vremena da proizvede rezultat, ili sistem ima problem sa isporukom rezultata ili sistem otkriva rezultate konkurenciji u poslovnom svetu.

Kroz projektovanje arhitekture rešavaju se navedeni problemi. Na primer:

Ukoliko je potrebno ostvariti visoke performanse softvera potrebno je:

- ispitati potencijalni paralelizam kroz dekompoziciju sistema i sinhronizaciju sistema
- identifikovati potencijalna uska grla performansi

Ukoliko je potrebno da sistem ima visoku tačnost potrebno je izvršiti proveru toka podataka, proces definisanja podataka i njihovih vrednosti

U slučaju da je za sistem važna bezbednost potrebno je:

- postaviti ograničenja u komunikaciji između određenih delova sistema koji su važni i koji sadrže poverljive podatke
- identifikovati delove sistema gde neautorizovan pristup može napraviti najviše štete
- uvesti specijalne elemente koji će omogućiti visok stepen poverenja u korisnika koji koristi sistem

REŠAVANJE PROBLEMA PROJEKTOVANJEM I DOKUMENTOVANJEM SOFTVERSKE ARHITEKTURE

Dokumentovanje softverske arhitekture omogućava projektantu softvera da iz različitih uglova gledanja izvrši analizu budućeg softvera.

Ukoliko sistem treba da podrži izmenljivost i portabilnost, potrebno je pažljivo razdvojiti i dokumentovati određene delove sisteme tako da eventualne izmene jednog dela sistema ne utiču na druge identifikovane delove.

Kada je potreban inkrementalni razvoj sistema, kroz kreiranje većih podskupova delova sistema potrebno je omogućiti da zavisnost veza između elemenata ne dovede do sindroma "*ništa ne radi dok sve ne radi*". Rešenje ovih problema je proces planiranja arhitekture tako da projektant ima mogućnost identifikovanja rešenja i komunikacije među njima. Dokumentacija softverske arhitekture ima tri obaveze gledajući iz ugla atributa kvaliteta.

Prvo, potrebno je naznačiti koji zahtevi atributa kvaliteta definišu projektovanje. Drugo, potrebno je označiti odabrano rešenje da se ispune zahtevi atributa kvaliteta. I kao poslednje, potrebno je označiti argument zašto predloženo rešenje ispunjava potrebne atribute kvaliteta.

Cilj je obezbediti dovoljno informacija tako da arhitektura sistema može biti analizirana i da su iz navedene analize vidljive mogućnosti sistema da ispuni zahteve atributa kvaliteta.

WHAT IS SOFTWARE ARCHITECTURE? (VIDEO)

Trajanje: 3:53 minuta.

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

Poglavlje 2

Sedam pravila za pisanje softverske dokumentacije

PRAVILO 1: PISATI DOKUMENTACIJU IZ TAČKE GLEDIŠTA ČITAOCA

Projektna dokumentacija treba da ispuni krajnji cilj a to je da bude razumljiva svakom čitaocu.

Dokumentacija arhitekture softvera je slična dokumentaciji koja se piše u drugim fazama procesa projektovanja. Kao takva, pridržava se istih osnovnih pravila koja razlikuju dobru, korisnu dokumentaciju od loše, ignorisane dokumentacije. Sedam pravila dokumentacije služe kako bi prikazali smernice za pisanje i čitanje tehničke dokumentacije. Navedena pravila daju objektivne kriterijume za ocenjivanje kvaliteta dokumentacije.

Pravilo podrazumeva da projektant softvera u svakom trenutku razmišlja o krajnjem cilju dokumentacije softvera koji projektuje a to je da dokumentacija treba da bude od koristi određenim korisnicima. Često se zbog kratkih rokova za projektovanje softvera zaboravlja na ovo pravilo i dokumentacija piše vrlo brzo bez detaljnog objašnjenja sistema. Takođe, na osnovu analize koju je prikazao naučnik Edsger Dijkstra (1930–2002), često mu je trebalo i po nekoliko sati kako bi razjasnio određenu rečenicu u tekstu. Naravno, ne podrazumeva se da se ovaj pristup primeni na pisanje dokumentacije softverske arhitekture ali jako je bitna tačna gledišta navedenog pristupa. Ukoliko svakom čitaocu bude trebalo u proseku dva minuta kako bi razumeo rečenicu, to u velikoj meri može uticati na vreme potrebno da prosečan čitalac razume celokupno dokumentaciju. Potrebno je pisati što konciznije i preciznije. Čitalac softverske dokumentacije koji primeti da je dokument pisan tako da njemu bude razumljiv, čitaće dokument svaki put kada mu bude potrebna pomoć u razumevanju softvera, dok ukoliko čitalac primeti da je dokument nerazumljiv neće ga koristiti naredni put. Dokumentacija napisana za čitaoca biće čitana dok dokumentacija napisana tako da bude pogodna za pisca neće.
Pre pisanja dokumentacije arhitekture softvera potrebno je:
- Saznati ko su čitaoci, šta znaju i šta očekuju od dokumentacije. Takođe, ukoliko je to moguće, potrebno je kroz neformalan razgovor sa predstavnicima raznih vrsta čitalaca izvršiti analizu očekivanja. Potrebno je izbegavati pretpostavke šta čitaoci mogu da znaju.
- Izbegavati pisanja bez prethodnog plana. Ukoliko projektant softvera piše dokumentaciju shodno redosledu misli koje ima, bez organizacionog plana, dokument arhitekture softvera biće nepotpun. Potrebno je izvršiti analizu potencijalnih poglavlja dokumenta kao i gde je potrebno pisati određene informacije. Strukturiranjem dokumenta budućim čitaocima biće olakšano čitanje i pronalaženje potrebnih informacija.
- Izbegavati žargon koji se koristi kroz komunikaciju unutar razvojnog tima. Često se može desiti da dokumentaciju može koristiti neko van razvojnog tima ili kompanije kao i novi član

koji nije upoznat sa korišćenim žargonom. Potrebno je uvrstiti i pojmovnik za definisanje specijalnih izraza koji se u dokumentaciji koriste.
- Izbegavati korišćenje skraćenica. Ukoliko je baš neophodno korišćenje skraćenica potrebno je napisati listu skraćenica koje su korišćene u dokumentu sa detaljnim objašnjenjem.

PRAVILO 2: IZBEGAVAJTE NEPOTREBNO PONAVLJANJE

Tokom pisanja dokumentacije softverske arhitekture potrebno je izbegavati nepotrebno ponavljanje informacija.

Svaka informacija unutar dokumentacije softverske arhitekture treba da bude zabeležena na tačno jednom mestu **.** Problem može nastati u slučaju da je određena informacija izmenjena kroz razvojni proces pa je potrebno na svakom mestu, unutar dokumenta, izvršiti izmenu što može predstavljati problem za projektanta dokumentacije. Takođe, izbegava se konfuzija, informacija koja se ponavlja više puta u dokumentu će verovatno biti drugačije interpretirana u različitim delovima i samim tim čitalac može biti zbunjen. Ponavljanje informacija može biti dozvoljeno samo u slučaju kada je potrebno dodatno pojasniti određeni problem tako da se ne zahteva od korisnika da se vrati na neku od prethodnih strana.

PRAVILO 3: IZBEGAVAJTE DVOSMISLENOST

Dvosmislenost se javlja kada se dokumentacija arhitekture softvera može tumačiti na više načina a najmanje jedan od tih načina je netačan.

Najopasnija vrsta dvosmislenosti je neodređena dvosmislenost u smislu da svaki čitalac će pomisliti da razume dokument ali nesvesno će svaki čitalac doći do različitih zaključaka.
Dva pravila koja omogućavaju izbegavanje dvosmislenosti su:

- izbegavanjem nepotrebnog ponavljanja biće izbegnut "gotovo ali ne sasvim slično" oblik dvosmislenosti
- razmatranja dokumenta sa predstavnicima čitalaca može omogućiti definisanje nejasnih delova dokumenta

Dobro definisana notacija uz preciznu semantiku omogućava put ka uklanjanju dvosmislenosti iz dokumenta.

Pravilo 3a: Objasniti notaciju:
Dijagrami na kojima se nalaze kvadrati, kutije i linije koje prikazuju povezivanje između kvadrata predstavljaju jedan od najvećih izvora dvosmislenosti u dokumentaciji softverske arhitekture. Navedeni dijagrami sigurno nisu dobra dokumentacija arhitekture i treba ih izbegavati. Kod čitalaca kvadrati u dijagramu mogu biti označeni kao moduli, objekti, klase, usluge, klijenti, baze podataka ili nešto drugo. Linije koje prikazuju povezivanje između kvadrata mogu biti pozivi, protok podataka, komunikacija, migracija. Prilikom crtanja

dijagrama potrebno je uključivanje ključa koji služi da čitaocu objasni šta je predstavljeno na dijagramu. Čak i da su dijagrami standardni, često se dešava da postoji više različitih verzija.

PRAVILO 4: KORISTITE STANDARDNU ORGANIZACIJU (ŠABLON)

Pisanje dokumentacije softverske arhitekture zahteva pridržavanje određenom šablonu pisanja.

Potrebno je uspostaviti standardnu i plansku organizacionu šemu tako da se svi dokumenti procesa projektovanja i dokumentovanja softverske arhitekture pridržavaju navedene šeme i da čitaoci budu upućeni u to.

Standardna organizacija (šablon) omogućava:

- kretanje kroz dokument i brzo pronalaženje informacija
- organizovanje sadržaja projektantu softverske dokumentacije
- beleženje informacija u određenim delovima dokumenta (poglavlje 4 može biti napisano pre završavanja poglavlja 1 ili 3)
- uključivanje pravila potpunosti informacija (standardna organizacija može biti osnova za proveru validacije dokumenta u vreme pregleda)

Nakon organizovanja dokumenta softverske arhitekture potrebno je izvršiti:

- organizovanje dokumentacije za lakše upućivanje i pronalaženje jer će dokument biti referenciran stotinama puta
- proveru da li je neki deo dokumenta ostao prazan. Ukoliko određeni deo dokumenta mora ostati prazan (iz razloga nepostojanja informacija koje se odnose na specifično poglavlje) potrebno je dodati oznaku koja će podsetiti autora dokumenta da se vrati i dopuni deo dokumenta.

PRAVILO 5: SNIMITI OBRAZLOŽENJE

Kroz dokumentaciju softverske arhitekture potrebno je izvršiti detaljno obrazloženje donetih odluka u procesu projektovanja.

Arhitektura predstavlja rezultat donošenja skupa važnih odluka projektanta softvera a dokumentovanje softverske arhitekture predstavlja obrazloženje tih odluka. Za najvažnije odluke, trebalo bi detaljno definisati i objasniti razlog zašto su napravljene. Takođe, potrebno je zabeležiti važne alternative odabiru koje nisu primenjene u procesu projektovanja. Kasnije, u toku procesa razvoja, kada odluke budu analizirane, detaljna dokumentacija omogućiće odgovore na postavljena pitanja. Često se dešava da projektant softvera za nedelju, mesec ili godinu dana od projektovanja arhitekture ne može zapamtiti zašto je doneo određenu odluku a tada detaljna dokumentacija ostvaruje svoju namenu.

PRAVILO 6: ZADRŽATI DOKUMENTACIJU TRENUTNOM, ALI NE PREVIŠE TRENUTNOM

Zastarela ili nepotpuna dokumentacija ne odražava realno stanje projektovanog softvera.

Potrebno je koristiti aktuelnu i tačnu dokumentaciju. Kada se postave pitanja o određenom softveru, najjednostavniji i najtačniji način je odgovoriti kvalitetnom projektnom dokumentacijom. Dokumentacija treba da predstavlja konačni autoritativni izvor informacija o određenom softveru.
U toku procesa projektovanja odluke se donose i preispituju sa velikom učestalošću. Revizija dokumenata koji ne predstavljaju realnu sliku procesa projektovanja je nepotreban trošak. Kraj svake iteracije u procesu razvoja softvera ili sprinta može biti povezan sa revizijom projektne dokumentacije.

PRAVILO 7: PREGLEDATI DOKUMENTACIJU I PROVERITI SVRSISHODNOST

Na kraju pisanja dokumentacije softverske arhitekture potrebno je izvršiti pregled i proveru svakog dela pre objavljivanja čitaocima.

Samo čitaoci dokumentacije kojima je namenjen dokument mogu dati ocenu da li dokumentacija softvera sadrži prave informacije i da li su one predstavljene na pravi način. Pre nego što dokumentacija bude zvanično objavljena potrebno je dati na pregled predstavnicima budućih čitalaca dokumenta (članovi tima koji imaju različite uloge u timu, korisnici softvera, naručioci softvera).

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

Poglavlje 3

Pregled dokumentacije softverske arhitekture

PROJEKTOVANJE SOFTVERSKE ARHITEKTURE

Projektovanje softverske arhitekture zahteva komunikaciju između naručioca softvera, projektanta softvera i programera.

Najbolje projektovana softverska arhitektura koja odgovara određenom poslovnom procesu koji se preslikava na sistem može biti neupotrebljiva u slučaju da implementeri (programeri) koji treba da razviju sistem ne razumeju kako da je primene ili projektovanu softversku arhitekturu primene pogrešno. U tom slučaju, sav proces projektovanja i analize od strane tima zaduženog za arhitekturu sistemu je beskoristan.

Ukoliko je razvijena složena arhitektura potrebno je tako razvijenu arhitekturu objasniti što detaljnije bez dvosmislenosti, kako bi svi ostali u razvojnom timu na pravi i brz način dobili sve potrebne informacije.

Dokumentacija " **priča**" umesto projektanta softvera. *Dokumentacija objašnjava arhitekturu umesto projektanta danas*, kao i u situaciji kada se projektant bavi drugim stvarima u procesu razvoja i nema vremena da odgovara na stotine pitanja o projektovanoj arhitekturi. Takođe, dokumentacija objašnjava arhitekturu i kada projektant napusti projekat i nije dostupan za rešavanje pitanja u procesu implementacije softvera.

ZAHTEVI ZA DOKUMENTACIJU SOFTVERSKE ARHITEKTURE

Dokumentacija se često razmatra kao naknadna i kao nešto što bi projektanti trebali da urade.

Može biti zahtevana i ugovorom ili zahtevana od strane naručioca softvera. Takođe, može biti definisana određenim standardom definisanim u kompaniji koja naručuje softver za svoj određeni poslovni proces. Nijedan od navedenih zahteva za dokumentaciju ne predstavlja kreiranje dokumentacije visokog kvaliteta.

Zašto je potrebno da arhitekta provede određeno vreme i potroši energiju za kreiranje dokumentacije koju će rukovodilac projekta samo čekirati kao izvršenu?
Projektanti softvera kreiraju softversku dokumentaciju bez uslova da je ona potrebna ili tražena od klijenta jer je od važnosti za dalji razvojni tok softvera kao i za njih same.

Dokumentacija sadrži sve informacije u procesu projektovanja. Dobro projektovani dijagrami unutar dokumentacije omogućavaju da se proces projektovanja odvija kontrolisano i sistematski. Dokumentacija pomaže projektantu softvera u toku projektovanja bilo da se radi o fazi projektovanja od šest meseci ili agilnom sprintu od šest dana.

3.1 Korisnici dokumentacije softverske arhitekture

DOKUMENTACIJA SOFTVERSKE ARHITEKTURE

Potrebno je omogućiti da se dokumentacija softverske arhitekture koristi u različite svrhe shodno potrebama korisnika dokumentacije.

Potrebno je da bude dovoljno apstraktna kako bi je mogući novi članovi tima brzo shvatili. Takođe trebala bi da ima dovoljno informacija koje bi se koristile kao osnov za kasniju analizu. Dokumentacija arhitekture je obavezna i opisna. Zavisno od čitaoca ovaj vid dokumentacije propisuje šta bi trebalo da bude urađeno uz postavljena ograničenja na odluke koje treba donositi.
Dokumentacija arhitekture koja služi za analizu performansi može se razlikovati od dokumentacije arhitekture koja se predaje programeru. Razlika je u sadržaju i različitim pogledima na arhitekturu softvera.

Proces planiranja dokumentacije i kasnije analize zahteva da se omogući podrška mogućim izmenama i unapređenjima kako bi se kompletan softver ispratio na pravi način. Kao što je navedeno, veliki broj ljudi će tražiti uvid u dokument arhitekture softvera nadajući se da će im taj dokument pomoći u obavljanju posla.
Dokumentacija softverske arhitekture može se koristiti za:

1. **Arhitektura predstavlja sredstvo obrazovanja.** Upotreba na ovaj način omogućava upoznavanje različitih ljudi sa sistemom. Ti ljudi mogu biti novi članovi razvojnog tima, eksterni analitičari ili novi projektanti softvera. Nova osoba je korisnik kome se prvi put prezentuje rešenje kroz prezentaciju u nadi da će na pravi način razumeti ono što je kroz dokumentaciju opisano.
2. **Arhitektura služi kao primarno sredstvo komunikacije između zainteresovanih strana za razvoj softvera**. Upotreba arhitekture kao sredstva komunikacije zavisi od toga koliko strane zainteresovane za razvoj softvera rade na komunikaciji.

KORISNICI SOFTVERSKE DOKUMENTACIJE

Glavni korisnik dokumentacije arhitekture je projektant.

U budućnosti, ukoliko projektant softvera bude ista osoba koja je pisala dokumentaciju softverske arhitekture ili na to mesto dođe novi projektant uvek će imati dokumentaciju koje može koristiti. Novi projektanti softvera koji su zainteresovani za razumevanje softvera, analizu eventualnih nedostataka i donesenih odluka u određenom trenutku procesa projektovanja mogu to jednostavno izvršiti. Čak i ako se radi o dokumentovanju arhitekture u kratkim crtama i takav način omogućava i unapređuje proces projektovanja softverske arhitekture.

Dokumentacija omogućava namenska poglavlja u okviru kojih su predstavljene različite vrste odluka projektanta softvera. Takođe, dokumentacija predstavlja grub ali koristan način za merenje napretka preostalog rada na projektu. Kompletan okvir za projektovanje arhitekture dobijen je iz detaljne dokumentacije a ključne odluke projektanta moraju biti upisane u dokumentaciju kako bi bile razmatrane u procesu donošenja novih odluka u nastavku procesa razvoja softvera.
Arhitektura predstavlja osnovu analize i konstruisanja sistema i govori programerima šta da implementiraju. Dokumentacija o arhitekturi mora sadržati informacije potrebne za procenu različitih atributa kvaliteta (sigurnost, upotrebljivost, performanse, dostupnost i mogućnost modifikacije) i prikazati projektantima softvera modele na osnovu kojih mogu izvršiti automatsko generisanje programskog koda određenih modula softvera

3.2 Dokumentacija arhitekture i atributi kvaliteta

PET NAČINA ZA PRIKAZ ATRIBUTA KVALITETA

Kroz pet načina za prikaz atributa kvaliteta vrši se analiza da li softverska arhitektura može da ispuni očekivane zahteve.

Kao što je već navedeno, dokumentacija o softverskoj arhitekturi služi kao osnova za analizu (kao osiguranje da arhitektura može da postigne i ispuni sve potrebne atribute kvaliteta).

Pet glavnih načina za prikaz atributa kvaliteta su:

1. **Definisanje prikaza atributa kvaliteta u procesu projektovanja.** Svaki glavni pristup projektovanju arhitekture odabran od strane projektanta softvera može imati definisane atribute kvaliteta. Klijent-server pristup je dobar za skalabilnost, slojevita arhitektura je dobra za prenosivost, dekompozicija zasnovana na informacijama je dobra za izvršavanje izmena, servisna arhitektura je dobra za interoperabilnost. Različiti pristupi omogućavaju i različite načine ispunjavanja određenih atributa kvaliteta i izmena koje je potrebno izvršiti.
2. **Definisanje atributa kvaliteta za svaki identifikovani element u softverskoj arhitekturi.** Pojedinačni arhitektonski elementi često imaju granične oznake kvaliteta koje su im dodeljene. Korisnici usluga treba da imaju uvid u brzinu,

sigurnost ili pouzdanost usluga. Navedene osobine mogu se navesti kao osobine elementa koji se prikazuje.

3. **Opisivanje svakog elementa softverske arhitekture na osnovu atributa kvaliteta**. Atributi kvaliteta vrlo često daju mogućnost "govora" posmatranog elementa. Sigurnost podrazumeva uvođenje sigurnosnih nivoa, autentifikacije korisnika, korišćenje revizorske staze ili zaštitnih zidova. Performanse definišu rokove, periode, ocene događaja i distribucije, ograničenja po pitanju vremena. Dostupnost omogućava definisanje primarnih i sekundarnih mehanizama, funkcionalnosti, kritičnih i nekritičnih procesa.

4. **Povezivanje funkcionalnih zahteva sa rešenjem (elementom softverske arhitekture).** Dokumentacija arhitekture najčešće sadrži mapiranje zahteva koji prikazuju kako je zahtev (kao i atibuti kvaliteta zahteva) ispunjen.

5. **Definisanje korisničkih grupa kojima je namenjena softverska dokumentacija.** Svaki zahtev atributa kvaliteta ima određene grupe ljudi koje su zainteresovane da li će taj zahtev biti ispunjen**.** Pod zainteresovanim ljudima podrazumeva se naručilac softvera i članovi razvojnog tima koji imaju određene uloge. Projektant softvera treba da omogući posebno mesto u uvodu dokumentacije koje će objasniti gde se rešenje određenog zahteva nalazi u dokumentu. Primer može biti: "Ukoliko ste analitičar performansi, obratite pažnju na procese i njihove osobine".

ISPLATIVNOST DOKUMENTACIJE

Prilikom razvoja softvera, projektant i članovi tima teže ka ispunjavanju zahteva naručioca softvera.

U procesu razvoja softvera često je prisutna i rečenica: "*Uradiću kvalitetnu dokumentaciju arhitekture softvera ukoliko budem imao slobodnog vremena za to*".

U tom slučaju, menadžeri projekta moraju da insistiraju na dokumentovanju softverske arhitekture koju je potrebno proizvesti tokom procesa razvoja softvera. Menadžeri projekta su uglavnom voljni da ulažu u resurse u aktivnostima koje donose korist. Takođe, projektanti moraju napraviti zadatak u okviru procesa razvoja softvera koji podrazumeva izradu i održavanje dokumentacije arhitekture softvera.

Cena izrade i održavanja dokumentacije arhitekture, na osnovu pretpostavki, trebao bi da bude pokriven izbegavanjem mogućih grešaka u procesu implementacije softvera. Analizom aktivnosti i zadataka u procesu razvoja softvera, projektant softvera može identifikovati kojim aktivnostima dokumentacija omogućava moguće smanjenje troškova ispravljanja greške. Tu najčešće spadaju aktivnosti kao što su kodiranje, reinženjering ili testiranje u kojima će ušteda troškova biti značajna.

3.3 Pogledi i izvan pogleda (Views and Beyond)

GENERALNI OPIS METODE POGLEDA I IZVAN POGLEDA ("VIEWS AND BEYOND")

Metod navan pogledi i izvan pogleda je skup tehnika koje podrazumevaju određenu filozofiju pristupanja dokumentovanju arhitekture softvera.

Ovaj pristup dokumentovanju softverske arhitekture nazvan je pogledi i izvan pogleda (Views and Beyond). Ovo nije klasičan metoda, ne sadrži korake unutar procesa sa ulaznim i izlaznim parametrima za svaki korak. Filozofija je da dokument arhitekture treba da bude od pomoći ljudima koji zavise od arhitekture u izvršavanju svojih poslovnih zadataka. Tehnike koje se primenjuju mogu biti svrstane u nekoliko kategorija:

1. Utvrditi potrebe naručioca softvera. Ukoliko ovo ne bude ispunjeno dokumentacija arhitekture neće služiti nikome.
2. Obezbediti informacije kako bi se zadovoljile potrebe za zapisivanje odluka u procesu projektovanja shodno različitim pogledima na arhitekturu.
3. Proveriti dobijenu dokumentaciju u cilju ispunjavanja definisanih potreba.
4. Pakovanje informacija u određene oblike kako bi bile dostupne naručiocima softvera

Dok tačke 3 i 4 označavaju aktivnosti usmerene ka dokumentima, tačke 1 i 2 označavaju aktivnosti koje treba izvršiti u procesu projektovanja softverske arhitekture. Na taj način primena "Views and Beyond" biće ne samo kao određene metode za definisanje dokumentacije arihtekture softvera već i kao način projektovanja softverske arhitekture koji omogućava projektovanje i paralelno dokumentovanje.

METODA POGLEDA I IZVAN POGLEDA U AGILNOM OKRUŽENJU

Često se može pročitati da agilni proces razvoja softvera i projektnu dokumentacija (a posebno dokumentovanje arhitekture) nije moguće spojiti.

Pristup pogledi i izvan pogleda "*Views and Beyond*" moguće je primeniti u agilnom procesu razvoja. Primenom navedenog pristupa moguće je definisati sledeće informacije o arhitekturi softvera: strukture, elemente, odnose, ponašanje, interfejse, sistemski kontekst. Projektant u agilnom procesu razvoja softvera treba da odabere šta će od navedenih informacija predstaviti kroz dokumentaciju.

Sam pristup ne zahteva od projektanta da obuhvati sve informacije koje su navedene. Potrebno je odabrati ono što je korisno i isplativo za dokumentovanje.

ARHITEKTURA KOJA SE MENJA BRŽE NEGO ŠTO SE DOKUMENTUJE

Dokumentaciju softverske arhitekture potrebno je menjati paralelno sa izmenama izvršenim u toku procesa razvoja softvera

Kada veb pretraživač u toku rada pronađe datoteku koju nikada ranije nije koristio, predložiće korisniku automatsko preuzimanje i instaliranje sa interneta bez potrebe za dodatnim koracima. Servisno orijentisani sistemi koriste dinamičko servisno otkrivanje i povezivanje koriste isti princip.

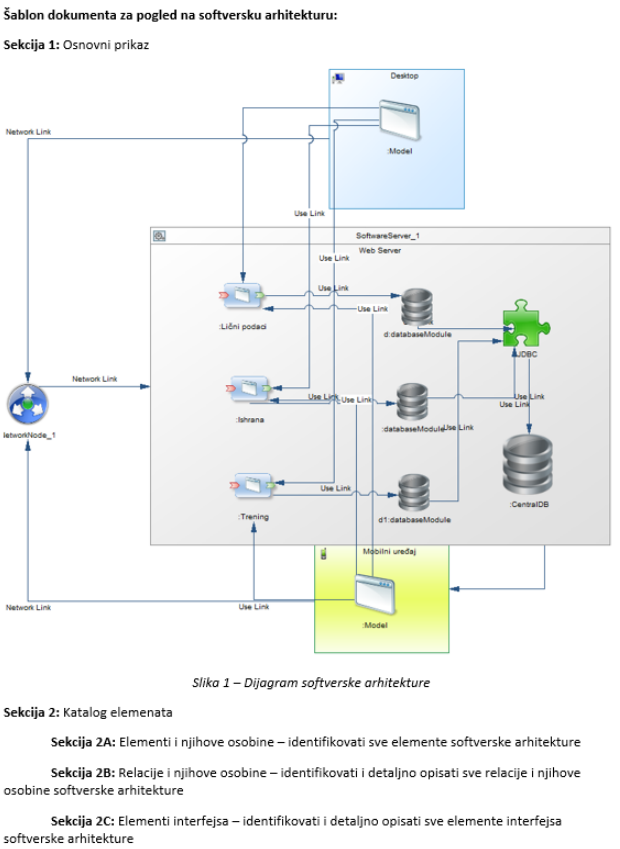
Na taj način se sistemi samoorganizaciju i rešavaju probleme na koje su naišli u toku rada. Bez obzira da li se arhitektura softvera menja tokom realizacije ili se dešava kroz određene faze razvoja softvera zaključak je isti. *Arhitektura softvera se menja mnogo brže od toka izmene dokumentacije arhitekture.* Nikada se neće desiti da implementacija čeka da se dokument arhitekture proizvede, prođe reviziju i bude zvanično odobren. Često se u praksi sve radi paralelno.

Znanje o arhitekturi sistema je jako bitno i važno i verovatno više nego za sisteme koji su razvijeni tradicionalnim pristupom. Projektant u dinamičnom okruženju (agilnom) može:

1. Dokumentovati sve verzije određenog sistema (sa posebnim stavkama šta se u svakoj verziji nalazi). Na ovaj način, dokumentovana arhitektura dobija dodatni opis, ograničenja i smernice koje moraju biti u skladu sa verzijama sistema.
2. Dokumentovati dozvoljene načine izmene arhitekture. Kroz primere navedeno je da se to najčešće radi izmenom ili dodavanjem komponenti.
3. Omogućiti da sistem zabeleži trenutnu softversku arhitekturu. Kada dođe do pada veb pretraživača ili sistema baziranog na servisno-orijentisanoj arhitekturi razvojni tim treba da zna tačno koja konfiguracija sistema je bila pokrenuta i koji problem je mogao da se desi. Ova mogućnost može da pomogne u rešavanju problema kroz prikaz komponenti i ostvarenih interakcija između njih.

POKAZNI PRIMER: ŠABLON DOKUMENTA ZA OPIS ARHITEKTURE IZ JEDNOG POGLEDA

Primer prikazuje šablon dokumenta za opis arhitekture iz jednog pogleda.



Slika 3.1.1 Šablon za dokumentovanje pogleda na softversku arhitekturu[Izvor: Nebojša Gavrlović]

Na slici 1 je prikazan šablon koji projektant softvera treba da popuni. Na početku šablona nalazi se dijagram arhitekture koji prikazuje pogled na softversku arhitekturu. Za svaku sekciju je naveden sadržaj koji je potrebno ispuniti.

Dokumentacija arhitekture softvera iz jednog pogleda sadrži:

- Primarnu prezentaciju, obično grafičkog prikaza koja opisuje primarne elemente i relacije unutar određenog pogleda
- Katalog elemenata koji objašnjava za šta određeni prikazani element služi unutar sistema i koje su njegove karakteristike
- Specifikaciju elemenata interfejsa i ponašanja
- Opis moguće varijabilnosti u sistemu, objašnjavajući tako ugrađene mehanizme za prilagođavanje arhitekture
- Obrazloženja i informacije o projektovanju

Poglavlje 4

Stilovi i pogledi softverske arhitekture - razlike

RAZLIKE IZMEĐU STILOVA SOFTVERSKE ARHITEKTURE I POGLEDA NA SOFTVERSKU ARHITEKTURU

U nastavku je dat uporedni prikaz karakteristika pogleda na softversku arhitekturu i stilova softverske arhitekture kako bi se uočila razlika između navedenih pojmova.

Pogledi na softversku arhitekturu predstavljaju reprezentaciju celokupne arhitekture koja je od značaja za jednog ili više aktera (stejkholdera) u sistemu. Softverska arhitektura može biti predstavljena brojnim različitim pogledima koji izvode ili ilustruju "značajne" elemente modela softverske arhitekture kako bi različitim čitaocima softverske dokumentacije na što bolji način približili bitne elemente arhitekture.

Pogledi na softversku arhitekturu omogućavaju:

- pregled elementa softverske arhitekture iz više različitih uglova
- dokumentovanje elemenata iz više različitih uglova
- uvid u različite atribute kvaliteta određenog elementa softverske arhitekture
- različite ciljeve i različite upotrebe softverske arhitekture

Odabir pogleda isključivo zavisi od očekivanja upotrebe dokumentacije arhitekture softvera.

Različiti pogledi ističu različite sistemske elemente ili veze između elemenata.

Stil softverske arhitekture predstavlja specifično rešenje određenog softvera koje se najčešće fokusira na način organizovanja programskog koda ili na organizovanje elemenata softverske arhitekture. Stilovi se mogu koristiti i za detaljno predstavljanje modula ili komponente. U okviru stilova koriste se šabloni (sastavljeni od predefinisanih tipova odnosa i elemenata) zajedno sa setom ograničenja koje je potrebno poštovati.

Stilovi softverske arhitekture omogućavaju:

- primenu striktno definisanih i dokumentovanih rešenja
- raspoređivanje modula ili komponenti sistema po određenom planu ograničavajući tako šta je određenom modulu ili komponenti dostupno od ostatka sistema
- deljenje komponenata ili modula do sitnih detalja u cilju prikaza strukture
- prilagođavanje različitim tipovima i potrebama sistema

Izbor stila, definiše da se u dokumentaciji softverske arhitekture prikazuju karakteristični i ograničavajući faktori koje odabrani stil daje softverskoj arhitekturi. Svaki stil softverske arhitekture ima definisan način dokumentovanja koji je potrebno poštovati

Poglavlje 5

Pogledi na softversku arhitekturu

OPIS POGLEDA NA SOFTVERSKU ARHITEKTURU

Najvažniji koncept povezivanja softverske arhitekture sa dokumentacijom je pogled.

Softverska arhitektura predstavlja kompleksan entitet koji ne može biti opisan u jednodimenzionalnom maniru.
Dokumentovanje arhitekture predstavlja pitanje dokumentovanja relevantnih pogleda na istu uz dodavanje dokumentacije koja opisuje više od jednog prikaza.
Projektant softvera treba da definiše tačne ciljeve dokumentovanja. Dokumentacija arhitekture softvera se može koristiti u velikom broju slučajeva: za implementere softvera, kao osnova za analizu, za specifikaciju za automatsko generisanje programskog koda, kao polazna tačka za razumevanje sistema ili kao plan za projektno planiranje.
Različiti pogledi na softversku arhitekturu omogućavaju uvid u različite atribute kvaliteta. Atributi kvaliteta utiču na izbor pogleda iz kog je potrebno izvršiti dokumentovanje softverske arhitekture. Pregled softverske arhitekture po slojevima može prikazati prenosivost i omogućiti razumevanje performansi i pouzdanosti sistema. Različiti pogledi omogućavaju različite ciljeve i različite upotrebe arhitekture.

Shodno navedenom u literaturi koja se bavi softverskom arhitekturom često se ne može pronaći adekvatan savet za upotrebu samo jednog pogleda na softversku arhitekturu. Odabir pogleda isključivo zavisi od očekivanja upotrebe dokumentacije arhitekture softvera. Različiti pogledi ističu različite sistemske elemente ili veze između elemenata.
Bitno je naglasiti da nijedan pogleda ne može u potpunosti predstaviti arhitekturu. Često je potrebno prikazati više različitih pogleda na softversku arhitekturu kako bi dokumentacija ispunila očekivanja od članova tima sa različitim ulogama.
Suština pogleda softverske arhitekture je da prikaže samo informacije koje su potrebne određenom članu tima ili izvršavanju određenog zadatka u toku razvojnog procesa. Svaki pogled naglašava određene aspekte sistema dok ignoriše druge aspekte a sve u interesu rešavanja problema.

KOMBINOVANJE RAZLIČITIH POGLEDA NA SOFTVERSKU ARHITEKTURU

U okviru softverske arhitekture moguće je koristiti jedan pogled ili kombinovati više različitih pogleda na softversku arhitekturu.

Dokumentacija softverske arhitekture koja sadrži kombinaciju više različitih pogleda u sebi sadrži:

- Uvodni deo kompletnog paketa, uključujući i uputstva koja pomažu čitaocu da jednostavno pronađe određeni deo unutar dokumentacije
- Opisne informacije kako su pogledi međusobno povezani, kao i način funkcionisanja kompletnog sistema
- Ograničenja i obrazloženja kompletne arhitekture opisanog sistema
- Informacije o upravljanju softverom koje mogu biti bitne za održavanje celog paketa dokumentacije

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

POKAZNI PRIMER: ŠABLON DOKUMENTA ZA OPIS ARHITEKTURE IZ VIŠE RAZLIČITIH POGLEDA

Primer prikazuje šablon dokumenta za opis arhitekture iz više različitih pogleda.

Šablon dokumenta softverske arhitekture iz više različitih pogleda:

Kako je organizovan dokument:

Sekcija 1: Plan dokumentacije – glavne instrukcije za čitaoca softverske dokumentacije o sadržaju dokumenta

Sekcija 2: Šablon pogleda – opis koji može omogućiti čitaocu brže i jednostavnije kretanje kroz dokument i pronalaženje potrebnih informacija

Šta je arhitektura:

Sekcija 3: Pregled sistema – tekstualni opis svih funkcija sistema.

Sekcija 4: Povezivanje dva različita pogleda – prikaz elemenata softverske arhitekture iz različitih pogleda na softversku arhitekturu. U okviru ove sekcije potrebno je detaljno objasniti primenjene poglede na softversku arhitekturu i prednosti odabranih pogleda.

Sekcija 5: Adresar elemenata – sekcija u kojoj je potrebno nabrojati sve elemente softverske arhitekture i detaljno ih opisati.

Sekcija 6: Rečnik korišćenih skraćenica – unutar ove sekcije potrebno je navesti sve korišćene skraćenice u dokumentu, opisati ih i navesti njihovu svrhu.

Odabrana arhitektura:

Sekcija 7: Ograničenja procesa projektovanja i objašnjenja – sekcija u kojoj je potrebno opisati zašto su odabrani pogledi dobri za softversku arhitekturu na kojoj se primenjuju. Takođe, treba obrazložiti odluku za odabir pogleda.

Slika 5.1 Šablon za dokumentovanje više pogleda na softversku arhitekturu [Izvor: Clements [2]]

Poglavlje 6

Kategorije stilova softverske arhitekture

TRI KATEGORIJE STILOVA SOFTVERSKE ARHITEKTURE

Stilovi softverske arhitekture podeljeni su u tri kategorije.

Projektanti softvera moraju istovremeno razmišljati o softveru na tri načina:

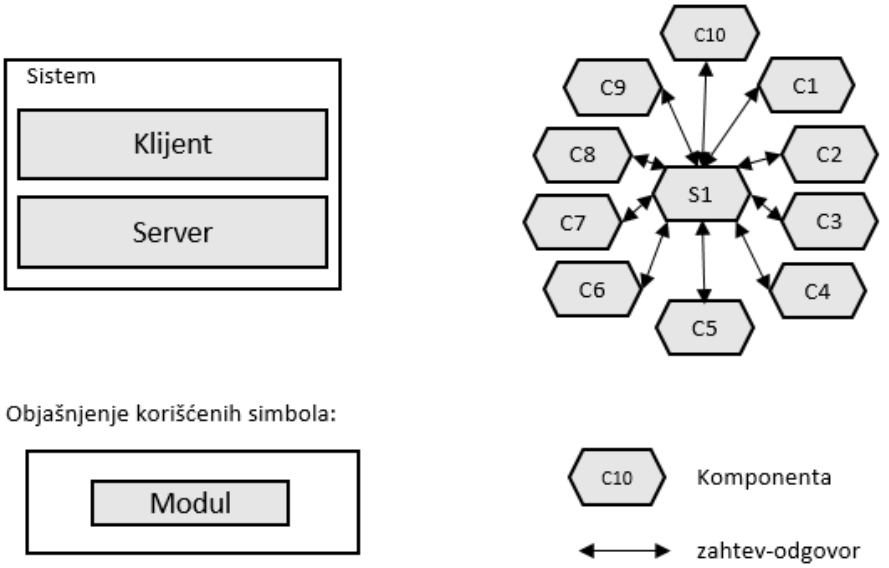
1. Na koji način je strukturiran kao skup jedinica za implementaciju
2. Na koji način je strukturiran kao skup elemenata koji imaju definisano ponašanje u radu i međusobnu interakciju
3. Na koji način se odnosi na ne-softverske strukture u svom okruženju

Navedeni stilovi softverske arhitekture spadaju u tri kategorije:

1. Stilovi modula
2. Stilovi komponenti i konektora (C&C)
3. Stil alokacije

Primena stila u okviru sistema prikazana je na slici 1 i predstavlja pogled.

Pogled stila modula predstavlja glavne jedinice implementacije sistema. Pogled stila komponenti i konektora (C&C) predstavlja jedinice izvršavanja sistema. Stilovi alokacije dokumentuju odnose između softverskog i ne-softverskog resursa sistema razvojnih i izvršnih okruženja.



Slika 6.1 Klijent-server sistem sa dva modula ali sa jedanaest komponenti [Izvor: Clements [2]]

OPIS TRI KATEGORIJE STILOVA SOFTVERSKE ARHITEKTURE

Polazna tačka projektovanja softverske arhitekture je najčešće paket odluka projektovanja.

Vrlo mali broj projektanata sistema razvija potpuno novu softversku arhitekturu. Kao što je već navedeno, često se koristi veliki broj različitih stilova arhitekture.
Potrebno je planirati dokumentaciju tako da sadrži jedan pogled stila modula, jedan pogled stila komponenta i konektor i jedan pogled stila alokacije.
Stil modula: Moduli predstavljaju primarne elemente stila modula. Moduli mogu imati oblik klase, kolekcije klasa, sloja, aspekta ili bilo kog drugog elementa procesa dekompozicije. Svaki modul sadrži određena svojstva koja su karakteristična i dodeljena samo tom modulu. Navedena svojstva imaju za cilj izražavanje bitne informacije koje prikazuju modul kao i sama ograničenja na modulu. Primer svojstva može biti odgovornost, informacije o vidljivosti u sistemu, autor ili vlasnik.
Stil komponenta i konektor (C&C): Stil komponenti i konektora prikazuju ponašanja izvršavanja u sistemu. Opisani su u pogledu komponenti i konektora. Komponenta predstavlja jednu od glavnih jedinica sistema za izvršavanje. Komponente mogu biti: usluge, procesi, teme, filteri, klijenti, serveri. Konektor je mehanizam interakcije između komponenti. Konektori mogu biti cevi, redove, protokole zahteva ili odgovora, pozive na događaje. Prednost stila komponenta i konektor je što je moguće izvršiti podelu već postojeće arhitekture stila komponente i konektori na nove komponente i konektore ukoliko je to potrebno.
Stil alokacije: Opisuju mapiranje softverskih jedinica na elemente okruženja u kome se softver razvija ili izvršava. Okolina može biti: hardver ili fajl sistemi koji podržavaju razvoj ili primenu.

ODABIR ELEMENATA I RELACIJA ZA DOKUMENTACIJU

Jedan od zadataka u dokumentovanju pogleda na arihtekturu softvera je odlučivanje koje osobine elemenata treba dokumentovati.

Svojstva gotovo uvek uključuju ime elementa, kratak opis i ulogu ili odgovornost u arhitekturi softvera. Recimo element slojevitog stila (koji spada u stil modula) treba da ima ime sloja, jedinicu softvera koju sloj sadrži i sposobnost koju sloj pruža. Kroz prikaz sloja moguće je navesti ime za svaki sloj, jedinicu softvera i mogućnosti koje svaka jedinica omogućava.

Pored ovih osnovnih osobina, postoje svojstva koja se koriste za analizu arhitekture. Ukoliko je potrebno analizirati performanse dokumentovane arhitekture onda svojstva treba da imaju vreme odgovora najboljeg i najlošijeg slučaja, maksimalan broj događaja koje određeni elemenat sistema može da obradi u jedinici vremena. Analiza bezbednosti arhitekture podrazumeva svojstva nivoa šifrovanja, pravila autorizacije za različite elemente i odnose.

Često se može desiti da i kombinacija stilova korišćenih u određenoj softverskoj arhitekturi može omogućiti različit pogled. Svaki stil ima sopstveni rečnik (tipova elemenata i odnosa). Zajedno predstavljeni stilovi treba da budu dosledno opisani i spremni za upoređivanje i selekciju. Ukoliko su stilovi predstavljeni na identičan način sa definisanim prednostima i manama projektant softvera može izvršiti analizu stilova i pronaći pravi stil za projektovanje softverske arhitekture.

NOTACIJE POGLEDA SOFTVERSKE ARHITEKTURE

Postoje različite formalne notacije za dostupnu arhitekturu softvera iako se ne može reći da su u širokoj upotrebi.

Tri glavne kategorije notacije su:

1. Neformalne **notacije.** Pogledi su prikazani (često grafičkim putem) korišćenjem određenih alata za crtanje dijagrama i prikaz vizuelnih konvencija. Semantika opisa se karakteriše prirodnim jezikom i ne može se formalno analizirati.
2. Poluformalne **notacije.** Pogledi su izraženi u standardnoj notaciji koje su propisani za grafičke elemente i pravila kreiranja. Takođe, ne pružaju potpuni semantički opis navedenih grafičkih elemenata. UML dijagrami predstavlja poluformalnu notaciju.
3. Formalne **notacije.** Pogledi su opisani u notaciji koja ima preciznu (matematički zasnovanu) semantiku. Formalna analiza je u tom slučaju moguća.

Opisane notacije obično pružaju grafički prikaz i osnovnu semantiku predstavljanja arhitekture. U nekim slučajevima dešava se da su formalne notacije specijalizovane za određene stilove. Određivanje vrste notacije za određenu upotrebu zahteva i nekoliko ustupaka u procesu projektovanja. Formalne notacije zahtevaju dosta više vremena i napora projektanta softvera ali doprinose smanjenju dvosmislenosti i većoj mogućnosti kasnije analize.

Poglavlje 7

Stilovi (šabloni) softverske arhitekture

STILOVI SOFTVERSKE ARHITEKTURE

Ponavljajući oblici u softverskoj arhitekturi česta su pojava čak i u slučaju potpuno drugačijih sistema koji služe za drugačije namene.

Navedeni oblici arhitekture nazivaju se stilovi arhitekture. Stilovi ili šabloni arhitekture omogućavaju da se unutar različitih softverskih sistema primenjuju striktno definisana i dokumentovana rešenja. Primenom slojevitog stila moguće je rasporediti module sistema po određenom planu ograničavajući tako šta je određenom modulu dostupno od ostatka sistema. Različiti sistemi imaće različit broj slojeva, različit sadržaj u svakom sloju i različita pravila koja utiču na određeni sloj. Takođe, različiti sistemi će imati različite protokole, različit broj servera i klijenata koji mogu podržati u klijent-server arhitekturi.

Neki stilovi arhitekture se mogu primeniti u svakom softveru. Svaki sistem je razdvojen u module u toku projektovanja kako bi se olakšao proces implementacije. To je stil razlaganja.

Izbor stila, definiše da se u dokumentaciji softverske arhitekture prikazuju karakteristični i ograničavajući faktori koje odabrani stil daje softverskoj arhitekturi. Obaveza dokumentovanja stila softverske arhitekture se podrazumeva u samoj softverskoj dokumentaciji. Od projektanta se očekuje da u određenom delu dokumenta opiše korišćeni stil i omogući dodatnu literaturu za čitaoce softverske dokumentacije. Opis stila unutar dokumentacije naziva se *vodič za stil softverske arhitekture.*

Obaveza dokumentovanja stila softverske arhitekture pomaže da drugi projektanti softvera imaju uvid u sam stil i dodatnu literaturu koja ga preporučuje.

PRIMENA STILOVA SOFTVERSKE ARHITEKTURE

Različiti delovi sistema mogu imati različite stilove arhitekture.

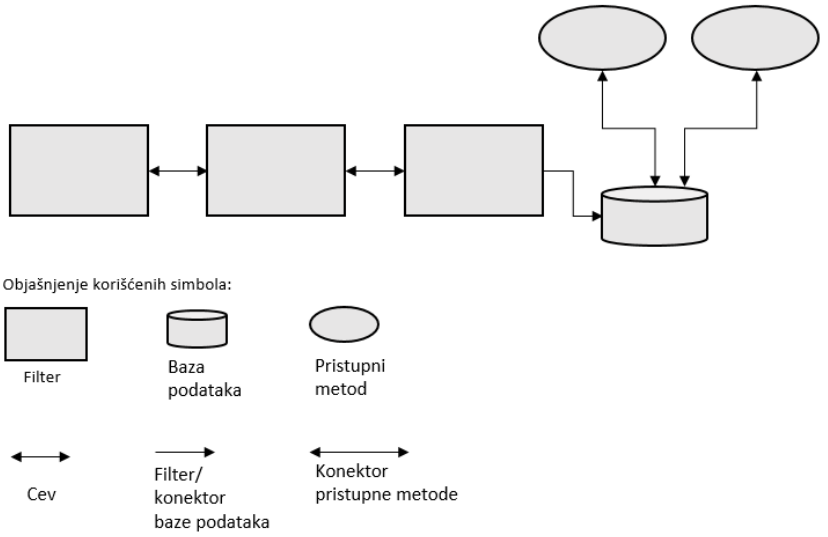
Primer upotrebe stilova može biti:

Sistem može koristiti stil "cevi i filtera" ("pipe-and-filter style") za obradu ulaznih podataka u sistem dok rezultati odlaze u bazu podataka kojoj kasnije pristupaju ostali delovi sistema. Ovakav sistem (prikazan na slici 1) bi bio mešavina stila "cevi i filtera" i stila deljenih podataka.

Stil cevi i filtera sastoji se od konektora ("pipe") i komponenata "filter" koji obrađuje podatke. Navedena komponenta "filter" može isporučiti obrađene podatke dobijene kroz "cevi" na nekoliko izlaznih portova. Stil prikazuje mogućnost izmene podataka u komponentama "filter" koji prolaze kroz "cevi". U narednim lekcijama biće detaljno obrađen navedeni stil sa dodatnim pokaznim primerima.

Primena kombinovanih stilova softverske arhitekture omogućava različit prikaz arhitekture tako da bude pregledan i dostupan različitim čitaocima softverske dokumentacije.

Dokumentacija ovakvog sistema bi u tom slučaju uključivala prikaz dva različita stila sa referencama ka eksternim dokumentima koji dodatno opisuju navedene stilove. Takođe, potrebno je obratiti pažnju da se jedan ili više elemenata (delova) sistema prikažu i u jednom i u drugom stilu kako bi čitaocima bilo jasno na koji način je obezbeđena komunikacija elemenata.



Slika 7.1 Sistem koji kombinuje cevi i filter stil sa stilom deljenih podataka [Izvor: Clements [2]]

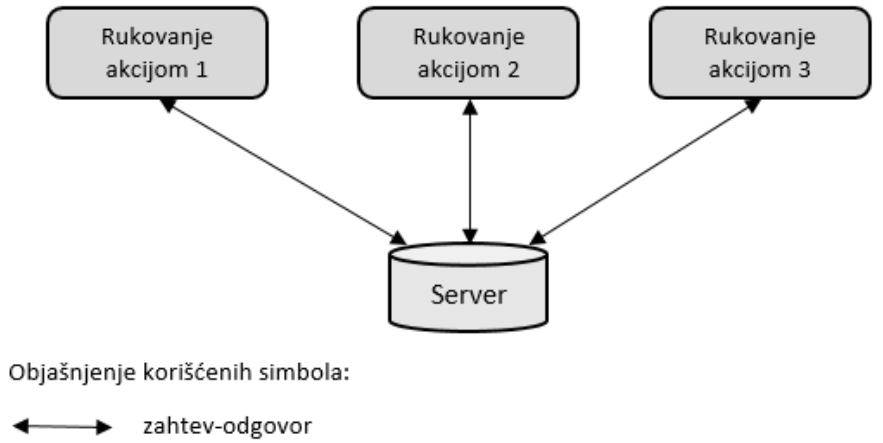
Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

KOMBINOVANJE RAZLIČITIH STILOVA SOFTVERSKE ARHITEKTURE

Projektant softvera može kombinovati različite stilove softverske arhitekture u okviru jednog sistema

Element prikazan u jednom stilu može biti sastavljen od elemenata u nekom drugom stilu softverske arhitekture. Recimo da provajder usluga u servisno-orijentisanom sistemu može biti nepoznat drugim provajderima usluga ili korisnicima usluga primenom višeslojnog stila. Dokumentovanje stila arhitekture ovog sistema obuhvatala bi prikaz servisno-orijentisane arhitekture koja opisuje celokupan sistem uz višeslojni prikaz servera na kome se sistem nalazi, videće rešenje softverske arhitekture.

Sistem koji sadrži baze podataka kao što je prikazano na slici 2 može se posmatrati kroz stil deljenih podataka ili stil klijent-server. Stil deljenih podataka omogućava prikaz komunikacije između komponenata sistema i međusobne razmene podataka.
Zavisno koji pogled odabere čitalac dokumentacije slika 2 daje prikaz softverske arhitekture sistema.



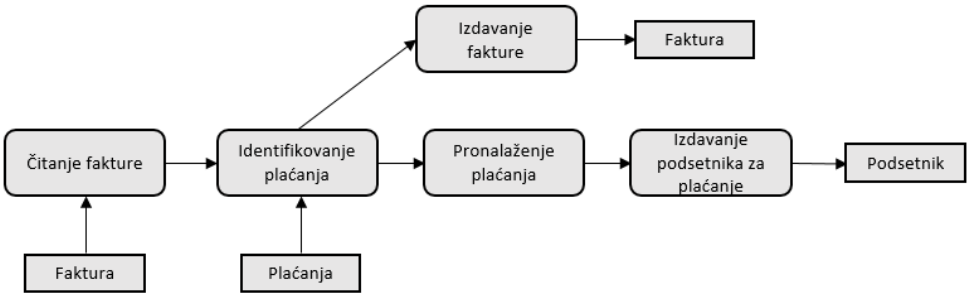
Slika 7.2 Softverska arhitektura u kojoj se koristi stil deljenih podataka ili klijent-server stil zavisno od pogleda čitaoca [Izvor: Clements [2]]

POKAZNI PRIMER: STIL CEVI I FILTERI

Dat je prikaz softverske arhitekture koja koristi stil cevi i filtera.

Kao što je već navedeno, stil cevi i filtera koristi simbole "filtera" za prikaz komponenata i simbol "cevi" za prikaz konektora između komponenata. "Filteri" u ovom slučaju prikazani na dijagramu su: čitanje fakture, identifikovanje plaćanja, izdavanje fakture, pronalaženje plaćanja, izdavanje podsetnika za plaćanje.

Podaci se šalju od jednog procesa ka drugom (kroz "cevi") a navedeni procesi predstavljaju "filtere" tih podataka.

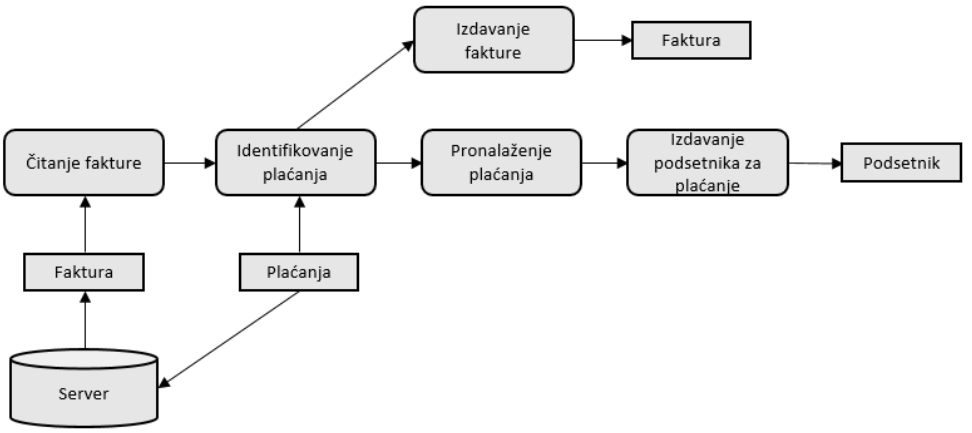


Slika 7.3 Primer stila "cevi i filteri" softverske arhitekture [Izvor: Clements [2]]

POKAZNI PRIMER: PRIMENA RAZLIČITIH STILOVA NA SOFTVERSKU ARHITEKTURU

Primena različitih stilova na softversku arhitekturu omogućava kombinovanje najboljih opcija odabranih stilova.

Na slici 4 prikazana je kombinacija stila deljenih podataka sa stilom cevi i filtera. Prethodni pokazni primer prikazuje stil cevi i filtera gde su navedeni procesi (filteri) dok na ovom primeru (na slici 4) postoji i server na kome se skladište podaci. Ovakav stil softverske arhitekture je kombinovanje stila cevi i filtera i stila deljenih podataka.



Slika 7.4 Primena dva stila različita stila na softversku arhitekturu [Izvor: Clements [2]]

TYPES OF ARCHITECTURAL STYLES (VIDEO)

Trajanje 3:08 minuta.

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

Poglavlje 8

Pokazna vežba-Dokumentovanje arhitekture softvera

PRIMER POPUNJAVANJA DOKUMENTA SOFTVERSKE ARHITEKTURE - SEKCIJA 1 I 2 - SOFTVER PERSONALNI TRENER

Na primeru softvera "Personalni trener" prikazana je dokumentacija u šablonu za prikaz pogleda na softversku arhitekturu.

Sekcija 1: potrebno je postaviti dijagram arhitekture sistema

Sekcija 2: Katalog elemenata

Sekcija 2A: Elementi i njihove osobine – Elementi softverske arhitekture su: model, lični podaci, ishrana, trening, JDBC, Central DB, database modul. Arhitektura sistema sadrži mobilni uređaj i desktop koji služe za pristupanje aplikaciji koja koristi MVC šablon arhitekture i oni predstavljaju zasebne komponente. Dalje „Model“ pristupa servisima aplikacije koji se nalaze na serveru i oni takođe predstavljaju komponente za sebe a to su „Lični podaci“, „Ishrana“, „Trening“.

Sekcija 2B: Relacije i njihove osobine – Na osnovu tih servisa oni pojedinačno pristupaju modulima baze podataka koji nakon toga pristupaju glavnoj bazi podataka putem JDBC drajvera odnosno JDBC komponente. Servis „Lični podaci“ predstavljen je kao veza „Write/ Read“ jer taj servis služi za čuvanje podataka o korisniku kao i dodatni update. Servis „Ishrana“ predstavljen je kao veza „Read“ jer samo preuzima podatke iz baze podataka i prikazuje ih korisniku, isto kao i servis „Trening“. (10min)

PRIMER POPUNJAVANJA DOKUMENTA SOFTVERSKE ARHITEKTURE - SEKCIJA 2D - SOFTVER PERSONALNI TRENER

Prikazan je primer popunjavanja sekcije 2D unutar sekcije 2 šablona dokumenta softverske arhitekture za prikaz jednog pogleda na arhitekturu softvera.

Sekcija 2C: Interfejsi za ovaj sistem nisu identifikovani.

Sekcija 2D: Odabir korisnika da li želi da se uloguje (ukoliko već poseduje nalog) ili da se registruje ukoliko ne poseduje nalog. Nakon toga sledi aktivnost unosa ličnih korisnika gde je potrebno da budu uneseni podaci kao što su ime i prezime. Sledeća aktivnost je unos telesnih vrednosti za visinu i težinu, završetkom te aktivnosti idemo na selektor (odnosno proces selekcije). Tu korisnik ima dve mogućnosti (obe mogućnosti mogu biti pozitivne). Ukoliko korisnik želi da pregleda preporučene treninge/ishrane sledi proces prikaza preporučenih treninga/ishrana. Nakon toga korisnik treba da odluči da li želi da izabere trening/ishranu ili ne. Ukoliko korisnik ne izabere trening/ishranu vraća se nazad na proces selekcije, dok ukoliko izabere trening/ishranu pristupa sledećim aktivnostima.

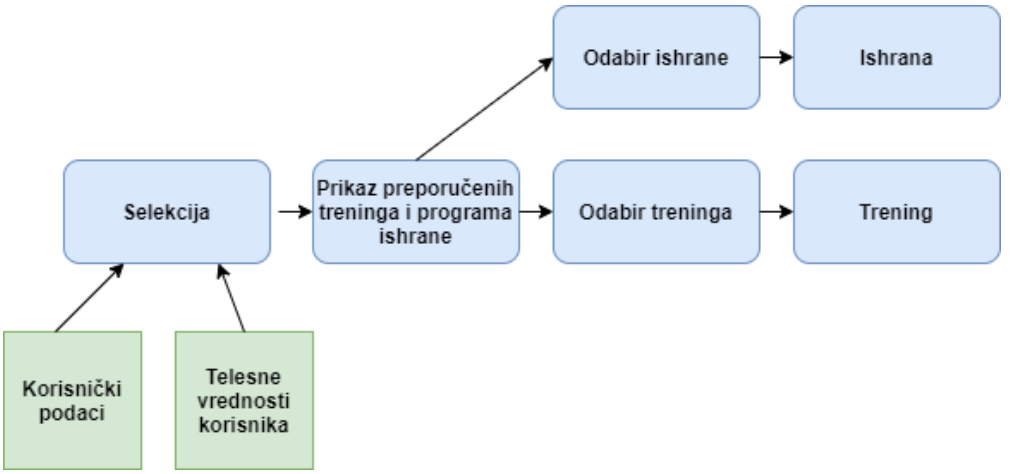
1. **Proces treninga** - Proces otvaranja materijala, komentarisanje vežbi, ocenjivanje treninga, beleženje rezultata. Po završetku treninga odnosno svih ovih aktivnosti korisnik se dovodi na proces zatvaranja dnevnika.
2. **Proces ishrane** - Proces beleženja dnevnog unosa namirnica, ocenjivanje treninga, beleženje rezultata. Po završetku beleženja ishrane odnosno svih ovih aktivnosti korisnik se dovodi na proces zatvaranja dnevnika.

Nakon završetka beleženja dnevnika korisnik dolazi u mogućnost odabira, da li želi da pregleda prethodno zatvoren dnevnik ili ne. Ukoliko korisnik ne želi više da gleda dnevnik automatski se izloguje i vraća na proces ponovnog odabira da li želi da se loguje ili registruje. Ukoliko korisnik želi da pregleda dnevnik dolazi do procesa pregleda dnevnika.(10min)

PRIKAZ SOFTVERSKE ARHITEKTURE KROZ STIL CEVI I FILTERI - SOFTVER PERSONALNI TRENER

Prikazana je primena stila cevi i filtera na softverskoj arhitekturi softvera "Personalni trener"

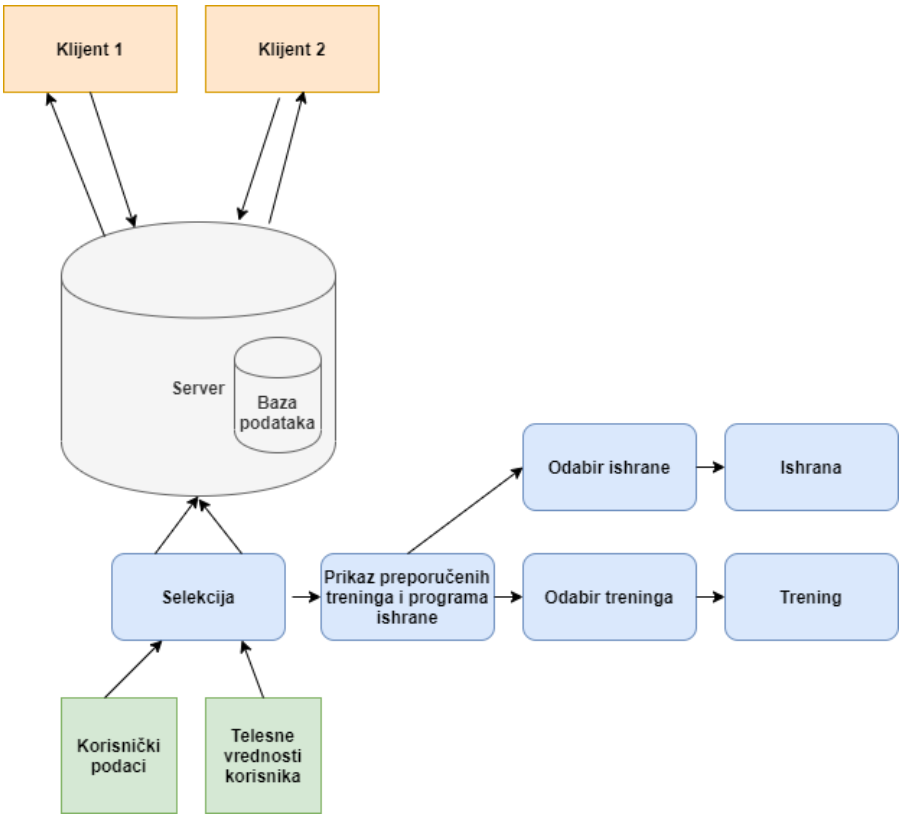
Za potrebe primera primene stila cevi i filtera odabran je jedan modul softvera. Takođe, na osnovu prethodno opisanog ponašanja softvera u sekciji 2D izvršeno je modelovanje dela softvera koji se bavi odabirom određenog treninga shodno unetim informacijama od strane korisnika. Podaci koji su potrebni procesima su u ovom slučaju: korisnički podaci i telesne vrednosti korisnika koje su potrebne za proces selekcije. Nakon toga podaci iz selekcije odlaze u prikaz preporučenih treninga i programa ishrane gde se račvaju na odabir ishrane ili odabir treninga. Krajnji rezultat je ishrana ili trening zavisno od toga šta je korisnik odabrao. (15min)



Slika 8.1 Prikaz modula softverske arhitekture za odabir tipa treninga i načina ishrane kroz stil cevi i filtera [Izvor: Nebojša Gavrilović]

PRIMENA RAZLIČITIH STILOVA NA SOFTVERSKU ARHITEKTURU - SOFTVER PERSONALNI TRENER

Primer predstavlja primenu klijent server stila softverske arhitekture i stila cevi i filtera na softver "Personalni trener".



Slika 8.2 Primer primene dva stila softverske arhitekture na softver "Personalni trener" [Izvor: Nebojša Gavrilovć]

Na slici 2 prikazana je arhitektura softvera "Personalni trener" koja sadrži dva stila softverske arhitekture. Takođe, primenom različitih pogleda na predstavljenu softversku arhitekturu moguće je identifikovati specifične delove odabranih stilova. (15 min)

Preporuka je da se za predstavljanje dva stila softverske arhitekture koristi free model (alat PowerDesigner ili draw.io).

Poglavlje 9

Individualna vežba - Zadaci

ZADATAK ZA SAMOSTALAN RAD

Na osnovu pokaznih primera u vežbama potrebno je uraditi zadatke za individualni rad za dati opis softvera.

Softver je namenjen za zaposlene u medicinskoj ustanovi (medicinska sestra, doktor, direktor klinike).

Svaki korisnik se prijavljuje i na osnovu prijave ima različite opcije korišćenja sistema. Opcije su sledeće:

- Prijavljivanje na sistem klinike
- Otvaranje kartona pacijenta
- Pristup kartonu pacijenta
- Slanje pacijentovih podataka iz kartona lekaru
- Prepisivanje terapije
- Izdavanje recepta za leka
- Davanje uputa za dalje lečenje
- Pretraživanje zaposlenih
- Potpisivanje naloga, zapisnika i pravilnika klinike
- Prikaz pravilnika klinike

Za navedeni scenario potrebno je:

1. Izvršiti dokumentovanje softverske arhitekture (popuniti šablon dokumenta za opis softverske arhitekture iz više različitih pogleda) proizvoljnog sistema kombinovanjem više različitih pogleda na softversku arhitekturu. (40 min)
2. Prikazati primenu stila cevi i filteri na softversku arhitekturu. (30 min)
3. Prikazati primenu više različitih stilova na softversku arhitekturu. (30 min)

ZADACI ZA SAMOSTALAN RAD - SEDAM PRAVILA ZA PISANJE

Primeniti sedam pravila za pisanje softverske dokumentacije.

Zadatak 1: Odabrati softversku arhitekturu proizvoljnog sistema

Zadatak 2: Primeniti sedam pravila za pisanje softverske dokumentacije u dokumentu arhitekture odabranog sistema. (20min)

ZADACI ZA SAMOSTALAN RAD - POGLEDI I IZVAN POGLEDA

Primeniti različite poglede na softversku arhitekturu i modelovati dijagrame arhitekture za svaki pogled.

Zadatak 1: Izvršiti primenu metode pogledi i izvan pogleda ("Views and Beyond") na proizvoljnoj softverskoj arhitekturi (30min).

- Predstaviti i popuniti poglavlja u okviru šablona dokumenta za jedan pogled na softversku arhitekturu.
- Detaljno dokumentovati glavne karakteristike softverske arhitekture iz opisanog pogleda.

Zadatak 2: Primeniti pet načina za prikaz atributa kvaliteta u toku pisanja softverske dokumentacije.(30min)

- Prvi način zahteva definisanje prikaza atributa kvaliteta za proces projektovanja
- Drugi način definiše atribute kvaliteta za svaki element identifikovan u softverskoj arhitekturi
- Treći način na osnovu atributa kvaliteta svakog elementa vrši dodatni opis svakog elementa
- Četvrti način povezuje funkcionalne zahteve sa rešenjem u vidu elementa unutar softverske arhitekture
- Peti način definiše grupe ljudi kojima je namenjena softverska dokumentacija

ZADACI ZA SAMOSTALAN RAD - POGLEDI NA SOFTVERSKU ARHITEKTURU

Prikazati dokumentovanje softverske arhitekture na osnovu jednog ili više različitih pogleda.

Zadatak 1: Izvršiti dokumentovanje softverske arhitekture proizvoljnog sistema kombinovanjem više različitih pogleda na softversku arhitekturu tako što je potrebno:

1. Prikazati arhitekturu softvera na osnovu pogleda za projektanta softvera i na osnovu pogleda za programera softvera. (30min)
2. Opisati svaki pogled i identifikovane razlike između pogleda (razlike u elementima softverske arhitekture koje treba da vidi projektant softvera u odnosu na programera) (15min)

Napomena: Nakon prikaza prvih pogleda ("Views") potrebno je izaći iz okvira pogleda (“Beyond”) i prikazati ostatak softverske arhitekture za programera softvera.

ZADACI ZA SAMOSTALAN RAD - KATEGORIJE STILOVA

Potrebno je primeniti grupe stilova na softversku arhitekturu.

Zadatak 1: Primeniti jednu od tri grupe stilova (stil modula, stil komponenta i konektor, stil alokacije) arhitekture na softversku arhitekturu proizvoljnog sistema. (30min)

Zadatak 2: Prikazati primere neformalne, poluformalne i formalne notacije u okviru softverske dokumentacije proizvoljnog sistema. (10min)

ZADACI ZA SAMOSTALAN RAD - STILOVI SOFTVERSKE ARHITEKTURE

Izvršiti kombinovanje različitih stilova arhitekture u okviru jednog sistema.

Zadatak 1: Odabrati stil cevi i filtera, stil deljenih podataka ili stil klijent-server i prikazati kroz softversku arhitekturu proizvoljnog sistema. (10min)

Zadatak 2: Izvršiti kombinovanje dva različita stila iz prvog zadatka i prikazati tako dobijenu softversku arhitekturu.(20min)

Zadatak 3: Primeniti način dokumentovanja softverske arhitekture shodno primeni dva različita stila. (20min)

ZADACI ZA DISKUSIJU NA VEŽBI

Otvorite diskusiju po svakom od dole postavljenih pitanja.

1. Na koji način atributi kvaliteta utiču na zahteve naručilaca softvera? (10min)

2.Ko definiše ko su korisnici dokumentacije softverske arhitekture? Da li je moguće kreirati različitu softversku dokumentaciju arhitekture istog sistema? Objasniti. (10min)

3. Navedite primer primene različitih stilova softverske arhitekture na proizvoljnom softveru. Od čega zavisi odabir stila softverske arhitekture? (10min)

4. Kada se vrši kombinovanje različitih pogleda na softversku arhitekturu softvera šta predstavlja cilj tog procesa? Da li korisnik može da primeti razliku kada je softverska arhitektura predstavljena kroz različite poglede? (10min)

5. Kada se koriste neformalne notacije, navedite primer alata koji bi bio korišćen za predstavljanje sistema neformalnom notacijom. (10min)

Poglavlje 10

Domaći zadatak br.2

PRAVILA ZA IZRADU DOMAĆEG ZADATKA

Student zadatake treba samostalno da uradi i da dostavi asistentu

Domaći zadatak br.2: Primeniti stil cevi i filtera (“"pipe-and-filter style") na minimum jedan modul softverske arhitekture odabranog sistema.
Pri radu, koristite Power Designer.

Domaći zadaci su dobijaju se po definisanim instrukcijama. Domaći zadatak se imenuje: SE311-DZ02-ImePrezime-brIndeksa gde su vrednosti Ime, Prezime i br.Indeksa vaši podaci. Domaći zadatak je potrebno poslati na adresu asistenta:

nebojsa.gavrilovic@metropolitan.ac.rs (tradicionalni studenti iz Beograda i internet studenti)
jovana.jovic@metropolitan.ac.rs (tradicionalni studenti iz Niša)

sa naslovom (subject mail-a) SE311-DZ02. Potrebno je poslati modelovane dijagrame i dokument sa opisom dijagrama ili odgovorima na pitanja.

Napomena: Domaći zadaci treba da budu realizovani u zadatku navedenom razvojnom okruženju i da predstavljaju jedinstveno rešenje svakog studenta. Prepisivanje i preuzimanje rešenja sa interneta ili od drugih studenata strogo je zabranjeno.

Poglavlje 11

Zaključak

ZAKLJUČAK

Cilj dokumentovanja arhitekture softvera je da se omogući detaljan opis kako bi drugi korisnici mogli da koriste, održavaju i izrade sistem. Dokumentacija postoji za dalju upotrebu kao sredstvo koje pomaže u učenju funkcionalnosti sistema, sredstvo komunikacije između članova razvojnog tima i kao osnova za analizu sistema. Takođe, dokumentovanje arhitekture softvera omogućava relevantne prikaze i modifikacije u cilju mogućnosti sagledavanja softvera iz različitih pogleda.

- **Stil modula** pomaže projektantima softvera da sagledaju softver kao skup implementacionih jedinica.
- **Stil komponente i konektora** omogućava da projektanti o softveru razmišljaju kao o skupu elemenata koje imaju vreme rada i proces međusobne interakcije
- **Stilovi alokacije** omogućavaju arhitektama da prikažu kako softver komunicira sa ne-softverskim strukturama u svom okruženju.

LITERATURA

U ovoj lekciji korišćena je kao obavezna literatura, referenca 2, uvodni deo(2. izdanje).

Obavezna literatura:

1. Onlajn nastavni materijal na predmetu SE311 Projektovanje i arhitektura softvera, , školska 2018/19, Univerzitet Metropolitan
2. P. Clements et ad. , Documenting Software Architectures: Views and Beyond, 2nd ed., Pearson Education (strane od 1 do 55)

Dopunska litertura:

1. D. Budgen, Software Design, 2nd ed.,Addison-Wesley, 2003.
2. T.C.Lethbrigde, R. Lagariere - Object-Oriented Software Engineering - Practical Software Development using UML and Java - 2005
3. Ian Sommervile, Software Engineering, Tenth Edition, Pearson Education Inc.,2016. ili 9th Edition, 2011
4. P.B. Kruchten, “The 4+1 View Model of Architecture,” IEEE Software, vol. 12, no. 6, 1995, pp. 42–55.
5. E. Gamma et al., Design Patterns:Elements of Reusable Object-Oriented Software, 1st ed., Addison-Wesley Professional, 1994.
6. J. Nielsen, Usability Engineering, Morgan Kaufmann, 1993.

7. Service-oriened Architecture, Concept, Technology, and Design, autora T. Erl u izdanju Pretince Hall, 2005, ISBN 0-13-185858-0.
8. P. Stevens, Using UML – Software Engineering with Objects and Components, Second Edition, Assison-Wesley, Pearson Eduction, 2006
9. R. Pressman, Software Engineering – A Practioner's Approach, Seventh Edition, McGraw Hill Higher Education, 2010

Veb lokacije :

- http://www.software-engin.com
- http://www.uml.org/