UNIVERZITET METROPOLITAN BEOGRAD

SE311 - PROJEKTOVANJE I ARHITEKTURA SOFTVERA

Stilovi povezivanja softverskih komponenata

Lekcija 04

PRIRUČNIK ZA STUDENTE

SE311 - PROJEKTOVANJE I ARHITEKTURA SOFTVERA

Lekcija 04

STILOVI POVEZIVANJA SOFTVERSKIH KOMPONENATA

UVOD

Pogled komponenta i konektor (C&C) prikazuje elemente koji imaju određeno vreme izvršavanja (proces, objekti, klijenti, serveri). To su komponente. Takođe pogled komponenta i konektor uključuje elemente interakcije kao što su komunikacione veze, protokoli, tokovi informacija i pristup deljenom skladištu. Navedeni elementi su konektori i predstavljeni su simbolima konektora u C&C pogledima.

Pogled komponenti i konektora je sveprisutan u predstavljanju softverske arhitekture i uglavnom se sastoji iz kutije i linije koja prikazuje povezanost između kutija. Pristup modelovanju softverske arhitekture korišćenjem simbola kutije i linije predstavlja neformalni pogled i kao takav cesto može biti nejasan čitaocima softverske dokumentacije. Problemi koji se mogu javiti kroz neformalni prikaz softverske arhitekture cesto se pojavljuju u toku vizuelnog predstavljanja izvršne strukture sistema korišćenjem pogleda komponente i konektora.

Ova lekcija vam obezbeđuje sledeće ishode učenja:

- Razumevanje stilova povezivanja softverskih komponenata (iz pogleda komponente i konektora). Pregled različitih stilova pogleda komponenata i konektora.
- Razumevanje stilova: cevi i filtera, klijent-server, peer-to-peer, servisno-orijentisanog stila, stila-objavi-pretplati se i stila deljenih podataka. Primena navedenih stilova na konkretnim primerima softverske arhitekture i razumevanje načina upotrebe u konkretnim situacijama.
- Razumevanje razlike u predstavljanju softverske arhitekture različitim stilovima pogleda komponente i konektora.

UVODNI VIDEO LEKCIJE

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

Poglavlje 1

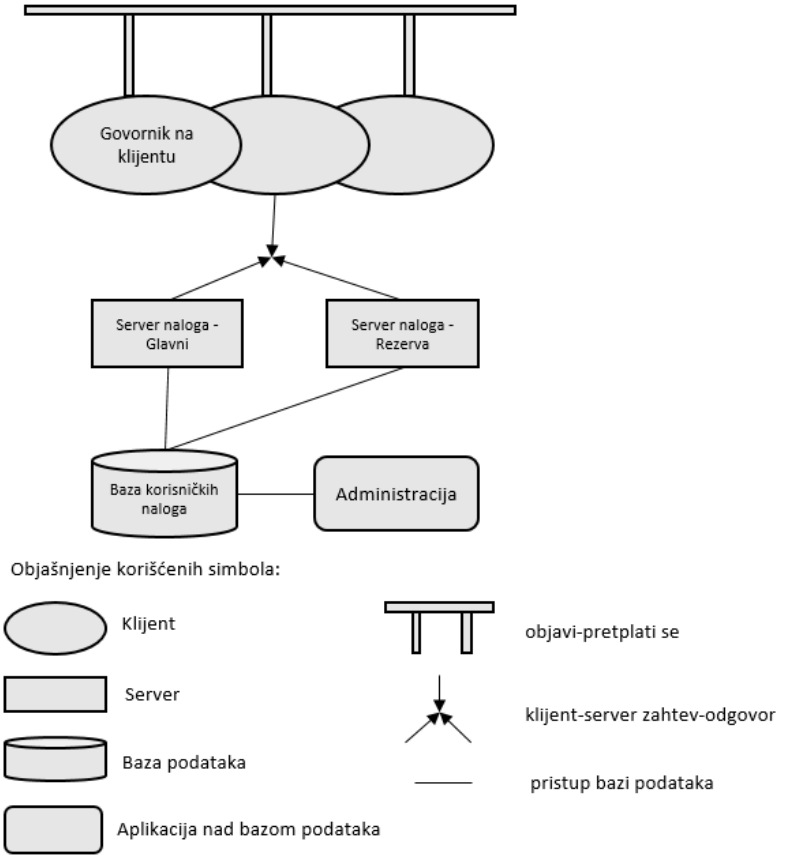
Pogled komponente i konektora

UPOTREBA POGLEDA KOMPONENTE I KONEKTORA

Pogled komponente i konektora omogućava prikaz softverske arhitekture shodno definisanim pravilima stila.

Slika 1 prikazuje pogled komponente i konektora na softversku arhitekturu sistema u toku izvršavanja. Prikazana softverska arhitektura sadrži deljeno skladište računa klijenata ("Baza korisničkih naloga") kome pristupaju dve server komponente i administrativna komponenta ("Administracija"). Grupa klijenata ("Govornik na klijentu") komunicira sa nalog serverima ("Server naloga - Glavni") koristeći klijent-server stil a takođe nalozi u okviru ove komponente mogu komunicirati između sebe korišćenjem konektora. Razlog za postojanje dva servera je poboljšavanje dostupnosti, u slučaju da glavni server prestane sa radom, rezervna kopija servera može preuzeti rad. Komponenta "Administracija" omogućava administratoru da pristupi i održava skladište deljenih podataka. Na slici 1 prikazane su tri vrste različitih konektora koji omogućavaju različite oblike interakcije između povezanih delova.

- **konektori klijenta i servera** - omogućava set klijenata koji dobijaju podatke kroz servisne zahteve od servera. Ovakav pristup omogućava prebacivanje podataka sa glavnog servera na rezervni u slučaju otkazivanja
- **konektor za pristup bazi podataka** - podržava transakcijski, autentifikovan pristup za čitanje, pisanje i praćenje baze podataka
- **konektor za "objavljivanje-pretplatu"** (publish-subscribe) - omogućava obaveštavanje (na osnovu pretplaćivanja) i objavljivanje asinhronih događaja



Slika 1.1 Pregled sistema iz "ptičije perspektive" u toku izvršavanja programa [Izvor: Clements [2]]

GRAFIČKO PREDSTAVLJANJE POGLEDA KOMPONENTE I KONEKTORA

Predstavljanje dijagrama kroz softversku dokumentaciju podrazumeva tekstualni opis svih identifikovanih elemenata.

Navedeni konektori predstavljaju složen oblik interakcije i zahtevaju kompleksne mehanizme u toku implementacije. Tip konektora klijent-server predstavlja protokol interakcije koji propisuje kako klijent započinju sesiju između klijenta i servera i opisuju način preusmeravanja ili prekidanja sesije. Implementacija klijent-server konektora uključuje mehanizme izvršavanja koji otkrivaju trenutni status servera, postavljaju zahteve klijenata, upravljaju dodavanjem i odvajanjem klijenata sa serverom. Konektori ne moraju biti binarni. Dva od tri tipa konektora na slici 1 mogu uključiti više od dva učesnika "objavljivanje-pretplata" magistrala ("bus") i klijent-server konektori.
Kvalitativne i kvantitativne analiza svojstava sistema (performanse, pouzdanost, sigurnost) mogu biti izvršene upotrebom pogleda komponenta i konektor.
Grafičko predstavljanje i dokumentovanje pogleda komponenta i konektor na slici 1:

- deluje kao ključ za prateću dokumentaciju gde se mogu naći detalji o elementima, relacijama i njihovim svojstvima
- unutar dijagrama koristi se samo ključ predviđen za pogled komponenta i konektor

- prikazuje broj i vrstu interfejsa na svojim komponentama i konektorima
- koristi apstrakcije komponenti i konektora koji mogu imati složenu implementaciju

Pored glavnog dokumenta, dodatna dokumentacija treba da objasni dostupnost sistema ili eventualno izvrši razlaganje sistema na podsisteme. *Razlaganjem sistema biće omogućen detaljan pregled svih elemenata sistema i razlaganje postojećih elemenata na nove elemente.*

ELEMENTI, RELACIJE I SVOJSTVA POGLEDA KOMPONENTE I KONEKTORA

Elementi ovog pogleda na softversku arhitekturu su komponente i konektori.

Svaki element ima određenu manifestaciju i koristi određene resurse u procesu izvršavanja.

Pregled	Komponente i konektori
Elementi	Komponente: glavne procesne jedinice i skladišta podataka. Komponenta ima skup portova preko kojih se ostvaruje komunikacija sa drugim komponentama (preko konektora) Konektori: način interakcije između komponenata. Konektori imaju skup uloga koje ukazuju kako komponente koriste konektor u interakcijama.
Relacije	Konekcija: Portovi komponenti su povezani sa konektorima kako bi se predstavio grafikon komponenti i konektora. Interfejs: U nekim situacijama dešava se da su portovi komponenti povezani sa jednim ili više portova „interne“ podarhitekture.
Ograničenja	Komponenta može biti povezana samo za konektor a ne za druge komponente. Konektori mogu biti povezani samo za komponente a ne za druge konektore. Dodaci mogu biti napravljeni samo između kompatibilnih portova i uloga. Konektori ne mogu biti prikazani kao izolovani simboli. Konektor mora biti povezan sa komponentom
Upotreba	Prikaz načina funkcionisanja sistema Rukovođenje procesom razvoja softvera kroz specifikaciju strukture i ponašanja elemenata u toku izvršavanja programa Pomoćno sredstvo za projektanta softvera u svrhu razumevanja atributa kvaliteta sistema (performanse, pouzdanost i dostupnost) u toku izvršavanja programa.

Slika 1.2 Elementi, relacije i svojstva pogleda komponente i konektora [Izvor: Clements [2]]

ELEMENT KOMPONENTA U POGLEDU KOMPONENTE I KONEKTORA

Komponente predstavljaju glavne računske elemente i skladišta podataka koji su prisutni u toku izvršavanja programa.

Bitno je naglasiti da svaka komponenta ima ime na osnovu koje je moguće odrediti funkciju komponente. Takođe, ime može omogućiti jednostavnije dokumentovanje grafičkog prikaza.

Komponente imaju interfejse koji se nazivaju portovi. Port predstavlja specifičnu tačku potencijalne interakcije komponente sa okruženjem. Najčešće ima eksplicitan tip koji definiše različite načine interakcije. Komponenta može imati više portova istog tipa. Takođe, moguće je da komponenta ima nekoliko portova istog tipa koji će upravljati ulaznim podacima ili komponenta server može imati više portova koji se koriste za omogućavanje interakcije sa klijentima. Pored portova moguće je postaviti oznaku "[3]" koja označava broj slučaja porta unutar sistema (u ovom slučaju tri replikacije). Port označen sa "[od 0 do 10]" znači da se port postoji od nula do 10 slučaja istog porta. Ovakav način obeležavanja portova je koristan prilikom definisanja tipova komponenti. Portove komponenti treba precizno dokumentovati, povezati ih na dijagramu koji predstavlja grafički prikaz i omogućiti detaljan opis u pratećoj dokumentaciji softverske arhitekture.

Komponenta unutar komponente i konektor pogleda može predstavljati i kompleksan podsistem koji se i sam može predstaviti kroz navedeni pogled. Pod arhitektura može biti predstavljena grafički u slučaju da nije previše kompleksna. Pod arhitektura komponente može biti predstavljena drugim stilom softverske arhitekture u odnosu na komponentu. Prilikom predstavljanja pod arhitekture komponente potrebno je prikazati vezu između internih i eksternih portova.

ELEMENT KONEKTOR U POGLEDU KOMPONENTE I KONEKTORA

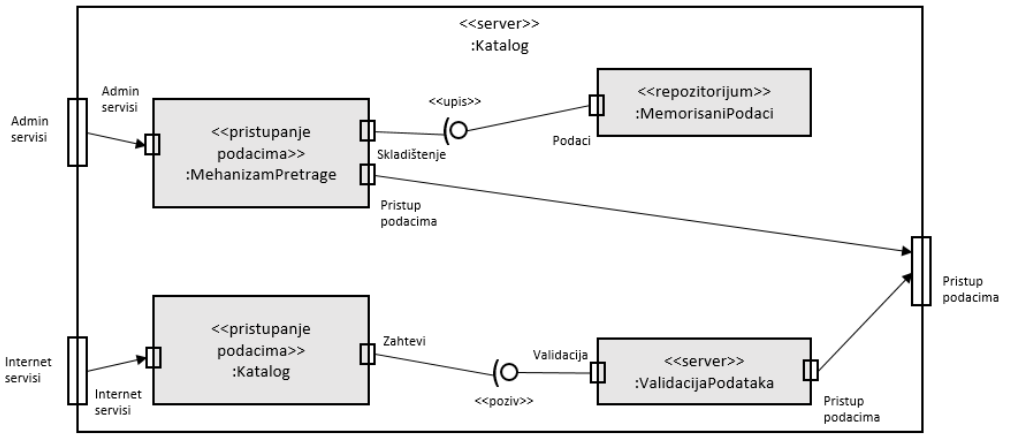
Konektori su druga vrsta elemenata u pogledu komponenta i konektor softverske arhitekture.

Primeri konektora su: poziv za servisiranje, asinhrone poruke, tokovi podataka. Konektori često predstavljaju složene oblike interakcije kao što je recimo kanal komunikacije orijentisan ka transakcijama između servera baze podataka i klijenta ili magistrala servisnih usluga koja posreduje u interakcijama između usluga korisnika i provajdera.
Konektori imaju uloge koji su njihovi interfejsi koji definišu načine na koje konektor može koristiti komponente za obavljanje interakcije. Primer može biti konektor klijent-server koji može imati ulogu poziva usluge (" invokes-services") i ulogu omogućavanja usluge (" provides-services"). Cev (u okviru " pipes and filter" stil softverske arhitekture) predstavlja uloge pisanja i čitanja. Uloge konektora se razlikuju od interfejsa modula po tome što se mogu replicirati ukazujući time koliko komponenti može biti uključeno u njihovu interakciju. Uloga obično definiše očekivanja od učesnika unutar interakcije.

RELACIJE U POGLEDU KOMPONENTE I KONEKTORA

Osnovna relacija pogleda komponenta i konektor je konekcija (attachment).

Navedena relacija prikazuje povezanost između određene komponente i određenog konektora na dijagramu stila komponenta i konektor softverske arhitekture. Konekcija se označava asocijacijom (priključivanjem) porta određene komponente na konkretnu ulogu konektora. Validna konekcija je ona u kojoj su portovi komponenti i uloge konektora međusobno kompatibilne pod ograničenjima koja su definisana unutar stila. Recimo, u okviru arhitekture "poziv-povratak" ("call-return") potrebno je potvrditi da su svi portovi "poziva" prikačeni na neki od povratnih konektora softverske arhitekture. Pored toga, potrebno je omogućiti da protokol porta bude u skladu sa ponašanjem konektora sa kojim je povezan. Druga relacija koja se koristi u pogledu komponenta konektor je "delegiranje interfejsa" ("interface delegation"). U slučaju da komponenta ili konektor ima pod arhitekturu, potrebno je detaljno dokumentovati odnos između unutrašnje strukture i spoljnih interfejsa te komponente ili konektora. Veza između pod arhitekture i spoljnih konektora može biti dokumentovana preko delegiranja interfejsa. U tom slučaju, vrši se mapiranje internih portova na spoljne portove (u slučaju da komponenta ima pod arhitekturu) ili mapiranje internih uloga na spoljne uloge (u slučaju konektora). Delegiranje interfejsa u okviru komponente prikazana je na slici 3.



Slika 1.3 Primer delegiranja interfejsa komponente softverske arhitekture [Izvor: Clements [2]]

SVOJSTVA U POGLEDU KOMPONENTE I KONEKTORA

Svojstva komponente i konektora su različita.

Svaki identifikovani element softverske arhitekture treba da ima ime i tip. Dodatna svojstva zavise od specifičnosti komponente ili konektora (projektant može dodati svojstva za detaljniji prikaz elementa). Identifikovana svojstva su potreba za vođenje implementacije softverske arhitekture i konfiguracije komponente i konektora. Ukoliko je potrebno izvršiti analizu performansi sistema na osnovu pogleda komponente i konektora, projektant može dodati svojstva koja se tiču kapaciteta, latence ili prioriteta. Najčešća svojstva koje se mogu koristiti su:

- **Pouzdanost.** Koristi se za utvrđivanje ukupne pouzdanosti sistema. Primer pitanja u okviru analize pouzdanosti može biti: "Koja je verovatnoća prestanka rada komponente ili konektora?"

- **Performanse.** Koristi se za analizu sistemskih mogućnosti, testiranje potrebnog vremena za odgovor na određeni zadatak ili za utvrđivanje propusnog opsega određenog elementa softverske arhitekture.
- **Potrebni resursi.** Koristi se za utvrđivanje potrebnog hardvera za određeni sistem. Primer analize potrebnih resursa može se bazirati na pitanju: " Koje su potrebe obrade i skladištenja podataka komponente ili konektora?".
- **Funkcionalnost.** Koristi se za razumevanje kompletnih funkcija i mogućnosti unutar sistema. Primer analize funkcionalnosti može biti: "Koje funkcije obavlja element (bilo da se radi o komponenti ili konektoru)?"
- **Bezbednost.** Koristi se za određivanje potencijalnih ranjivih tačaka sistema u okviru pogleda komponenta i konektor.
- **Konzistentnost.** Koristi se u procesu analize ili simulacije performansi komponenti i za identifikovanje mogućih blokada u sistemu.
- **Nivo.** Koristi se za definisanje procedure izgradnje sistema i raspoređivanje komponenti ili konektora u softverskoj arhitekturi.

UPOTREBA STILA KOMPONENTE I KONEKTORA

Pogled komponenta i konektor se najčešće koristi kako bi se prikazao način funkcionisanja sistema programerima koji se bave implementacijom ili drugim zainteresovanim korisnicima siste

Ovaj pogled određuje strukturu i ponašanje elemenata izvršavanja i daje odgovore na sledeća pitanja:

- Koje su glavne izvršne komponente sistema i kako komponente komuniciraju?
- Koja su glavna skladišta deljenih podataka?
- Koji delovi sistema se repliciraju i koliko puta?
- Kako podaci teku kroz sistem i kako se dobijaju?
- Koji protokoli interakcije se koriste u komunikaciji elemenata?
- Koji delovi sistema rade paralelno?
- Kako se struktura sistema menja u toku izvršavanja?

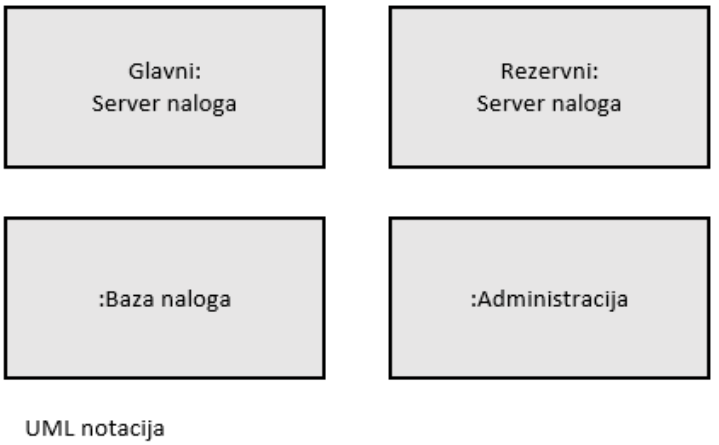
Pogled komponenta i konektora takođe se koristi za razjašnjavanje atributa kvaliteta sistema kao što su performanse, pouzdanost i dostupnost. Dobro dokumentovani prikaz omogućava projektantima softvera da predvide svojstva sistema, izvrše merenje elemenata softverske arhitekture i definišu interakciju između elemenata. Pravilno dokumentovanje pouzdanosti pojedinih elemenata softverske arhitekture omogućava da projektant softvera u toku analize ukupne pouzdanosti sistema ima relevantne podatke.

NOTACIJE POGLEDA KOMPONENTE I KONEKTORA - PRIKAZ KOMPONENTI

Upotreba poluformalnih notacija u UML jeziku omogućavaju prikaz komponenti i konektora softverske arhitekture i prikaz važnih informacija kao što su interfejsi, svojstva i opis ponašanja

Kao i za druge poglede softverske arhitekture, i za pogled komponente i konektora neformalna notacija podrazumeva upotrebu kutija i linija, gde kutije predstavljaju komponente a konektori su predstavljeni linijama. Takođe, moguće je dodatno obeležiti komponente i konektore unutar neformalne notacije i tako unaprediti prikaz softverske arhitekture. Pored nazivanja komponenti potrebno je precizirati i njihovo značenje. To je moguće izvršiti u dodatnom opisu na dijagramu (korišćenjem ključa koji objašnjava svaki nacrtani simbol na dijagramu). Prikaz konektora podrazumeva da projektant softvera definiše smer strelica konektora i tako na pravi način prikaže interakciju između povezanih komponenti. Prikaz komponenata kroz UML jezik

Moguće je razlikovati tipove komponenti i pod arhitekture komponente što je korisno prilikom definisanja specifičnih prikaza softverske arhitekture. Na slici 4 prikazano je predstavljanje komponenti kroz UML jezik. Imena komponenata koja ne sadrže dvotačku (:) su tipovi, komponente koje sadrže (:) su instance (obeležene su sa (:) na početku imena) gde je ime prikazano sa leve strane. Nepoznate instance kao što je "Baza naloga" na slici počinje sa dvotačkom (:). Kao što je već opisano, dijagram je moguće detaljno opisati kroz vodič ili ključ koji predstavlja sve simbole koji su korišćeni u dijagramu.

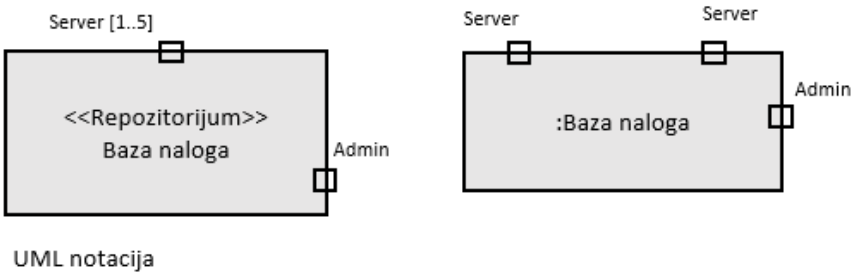


Slika 1.4 UML prikaz komponenti [Izvor: Clements [2]]

NOTACIJE POGLEDA KOMPONENTE I KONEKTORA - PRIKAZ PORTOVA I KONEKTORA

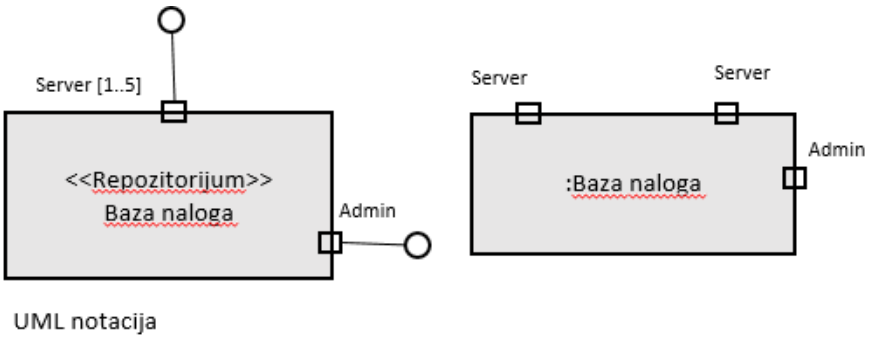
Obeležavanje portova i konektora u prikazu softverske arhitekture kroz pogled kompente i konektor zahteva upotrebu formalne notacije.

Korišćenjem UML portova kao na slici 5 moguće je predstaviti portove u okviru stila komponenta i konektor softverske arhitekture. UML port može biti obeležen i brojem (kao što je prikazano u levom delu slike 3) iako se to obično vrši samo na tipovima komponenti. Broj portova na komponenti može biti obeležen kao na desnom delu slike 5.



Slika 1.5 Prikaz korišćenja UML portova u stilu komponenta i konektor [Izvor: Clements [2]]

U okviru pogleda komponente i konektora potrebno je izvršiti prikaz interfejsa u određenim slučajevima. Kroz UML jezik moguće je predstaviti interfejs kao na slici 6. Svaki port komponente (kvadratom predstavljen na slici), može imati proizvoljan broj interfejsa. Slika 4 prikazuje iste komponente u okviru pogleda komponente i konektora kao na slici 3 ali sada svaki port tipa " Baza naloga" sadrži jedan prošireni interfejs koji se kroz UML može razraditi prikazom metoda ili atributa. Instanca "Baza naloga" sa desne strane ima tačno dva server porta dok su interfejsi izostavljeni u ovom slučaju.

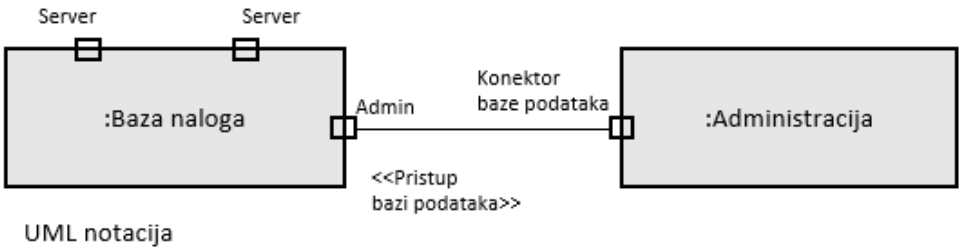


Slika 1.6 Prikaz korišćenja UML interfejsa u stilu komponenta i konektor [Izvor: Clements [2]]

NOTACIJE POGLEDA KOMPONENTE I KONEKTORA - PRIKAZ KONEKTORA KROZ UML JEZIK

Konektor između dve komponente u UML jeziku prikazuje se jednostavnom linijom.

Moguće je prikazati i tip konektora korišćenjem stereotipa. Slika 7 prikazuje konektor predstavljen korišćenjem UML jezika. Ukoliko su svi konektori istog tipa, nije potrebno za svaki konektor prikazivati tip na dijagramu već to specificirati u ključu koji opisuje dijagram.



Slika 1.7 Prikaz korišćenja UML konektora između dve komponente stila komponenta i konektor [Izvor: Clements [2]]

Uloge konektora ne mogu biti eksplicitno predstavljene korišćenjem UML konektora jer element konektora ne uključuje prikaz interfejsa (za razliku od UML simbola porta koji omogućava prikaz interfejsa). Preporučuje se da se označi kraj konektora i tako identifikuje opis uloga samog konektora.

RELACIJE POGLEDA KOMPONENTE I KONEKTORA SA DRUGIM POGLEDIMA NA SOFTVERSKU ARHITEKTURU

Pogled komponenta i konektor može biti korišćen zajedno sa pogledom na module softverske arhitekture.

Pogled komponenta i konektor razlikuje se od prikaza modula u određenim delovima. Elementi komponenta i konektor pogleda predstavljaju instance entiteta izvršavanja dok elementi pogleda modula predstavljaju entitete za implementaciju. Primer može biti sistem koji ima deset klijenata koji su povezani na jedan server. U tom slučaju, postoji jedanaest komponenti i deset konektora ali tačno dva modula.

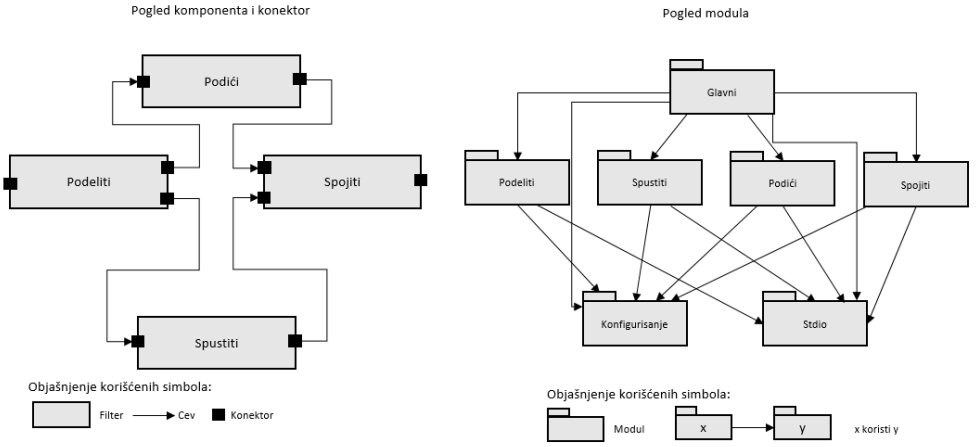
- Isti modul programskog koda može biti korišćen od više različitih elemenata pogleda komponenta i konektor
- jedna komponenta pogleda komponenta i konektor može izvršavati programski kod koji se nalazi u različitim modulima
- komponenta pogleda komponenta i konektor može imati više različitih tačaka interakcije gde je svaka tačka definisana od istog interfejsa modula
- kako svaki modul ne mora biti prikazan u svakom pogledu modula softverske arhitekture, komponenta pogleda komponenta i konektor ne mora biti mapirana u bilo kom modulu određenog pogleda modula

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

UPOREDNI PRIKAZ KOMPONENTA I KONEKTOR POGLEDA I POGLEDA MODULA SOFTVERSKE ARHITEKTURE

Dat je primer komponente i konektor pogleda u odnosu na pogled modula softverske arhitekture

Na slici 8 prikazan je pogled modula softverske arhitekture i on se koristi za implementaciju uz pomoć programskog jezika C.Ovaj pogled prikazuje odnose između modula gde glavni modul ("glavni") komunicira sa ostalim modulima koji su prikazani ("podići", "spustiti", "podeliti" i "spojiti").Glavni modul kontroliše ostale module preko modula "Konfigurisanje" i "Studio".Pogled komponenta i konektor prikazuje isti sistem opisan kroz stil cevi i filtera. Svaka komponenta je filter koja vrši transformaciju ulaznih podataka. Komunikacija između komponenata je prikazana a konektori sprovode podatke



Slika 1.8 Uporedni prikaz komponenta i konektor pogleda i pogleda modula softverske arhitekture [Izvor: Clements [2]]

Poglavlje 2

Različiti stilovi pogleda komponente i konektora

STILOVI TOKA PODATAKA SOFTVERSKE ARHITEKTURE

Primena ovih stilova je u okviru domena gde se obavlja obrada podataka kroz nekoliko transformacionih koraka.

Stilovi toka podataka omogućavaju model u kome se komponente prikazuju kao transformatori podataka a gde konektori prenose podatke iz izlaza jedne komponente na ulaz druge komponente. Svaka vrsta komponente u stilu toka podataka ima određeni broj ulaznih i izlaznih portova. Posao komponente je da obrađuje podatke dobijene na ulazne portove i prosleđuje ih na izlazne portove. Najpoznatiji stil toka podataka je stil cevi i filtera.

STIL POZIV - POVRATAK SOFTVERSKE ARHITEKTURE

Primeri stilova poziv-povratak su klijent-server, peer to peer i REST stilovi.

Stilovi softverske arhitekture poziv-povratak predstavljaju model u okviru koga komponente pružaju skup usluga koje mogu biti pozvane od strane drugih komponenti. Komponenta koja poziva uslugu pauzira sve dok ta usluga nije završena. Konektori su odgovori za dostavljanje zahteva za određenu uslugu od strane komponente koja podnosi zahtev za vraćanje rezultata. Stilovi poziv-povratak su različiti. Postoje varijante koje se razlikuju u pogledu ponašanja svojih konektora, gde konektor može omogućiti rukovanje greškama. Takođe, razlika stilova je i u ograničenjima ili nivoima deljenja komponenti u prikazu softverske arhitekture.

STIL PEER-TO-PEER SOFTVERSKE ARHITEKTURE

Stil peer-to-peer omogućava da komponente direktno komuniciraju kao komponente na istom nivou uz razmenu usluga.

Interakcija omogućava slanje zahteva određene komponente drugoj komponenti koja treba da pruži odgovor na dobijeni zahtev. U tom slučaju, bilo koja komponenta može da komunicira sa bilo kojom drugom komponentom. Konektori u ovom stilu softverske arhitekture mogu

uključivati složene dvosmerne protokole interakcije i na taj način vršiti održavanje dvosmerne komunikacije između dve ili više komponente. Primer upotrebe peer-to-peer može biti Skype.

STIL SOFTVERSKE ARHITEKTURE ZASNOVAN NA DOGAĐAJU

Stilovi zasnovani na događajima omogućavaju komunikaciju između komponenata korišćenjem asinhronih poruka

Sistemi koji koriste ovaj stil su organizovani kao skup komponenti koji pokreću ponašanje u drugim komponentama kroz događaje. U upotrebi su različiti stilovi kao i konektori koji se koriste. Konektori su od tačke to tačke ("point to point") i prenose poruke na način sličan povratnom pozivu uz omogućavanje više istovremenih poziva. Takođe, konektori mogu biti i višestranični omogućavajući da se događaj iz određene komponente šalje na više komponenti. Sistemi u okviru kojih se koriste višestranični konektori su sistemi za objavljivanje i pretplatu.

Poglavlje 3

Stil cevi i filtera

PREGLED STILA CEVI I FILTERA

Stil cevi i filtera omogućava prenos podataka konektorom sa izlaznog porta komponente na ulazni port naredne komponente.

Konektor ovog stila je tipa cev ("pipe"). Jedna komponenta (filter) može isporučiti obrađene podatke na nekoliko izlaznih portova. Cevi deluju kao jednosmerni vodovi obezbeđujući komunikacioni kanal sa baferom koji čuva informaciju o prenosu podataka od jednog do drugog filtera. Filteri mogu delovati asinhrono i istovremeno omogućavajući na taj način projektantu softvera da razume krajnje ponašanje komponenata softverske arhitekture.

ELEMENTI, RELACIJE I SVOJSTVA STILA CEVI I FILTERA

Elementi stila cevi i filtera su cevi i filteri na osnovu kojih se vrši predstavljanje softverske arhitekture.

Upotreba stila cevi i filtera najčešće je u sistemima za transformaciju podataka gde se ukupna obrada može razvrstati u niz nezavisnih koraka gde je svaki korak odgovaran za inkrementalnu transformaciju svojih ulaznih podataka. Na taj način, filteri mogu primati podatke od senzora filtrirajući ih u podatke iz kojih mogu biti generisani određeni izveštaji za korisnike. Analize koje je moguće izvršiti upotrebom stila cevi i filtera mogu pokazati rezultate koji se tiču performansi, propusnosti sistema, protoku informacija na ulazu i izlazu sistema.

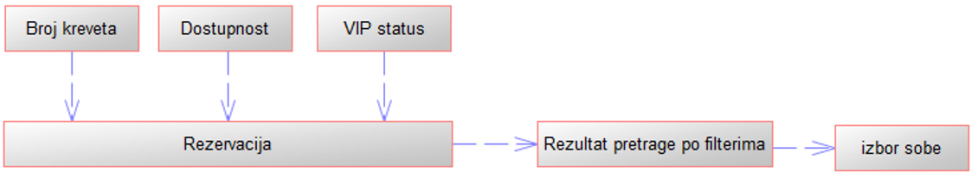
Pregled	Cevi i filteri
Elementi	Filter, komponenta koja transformiše podatke koji se nalaze na ulaznom portu (ulazni podaci) u podatke koji se nalaze na izlaznom portu (izlazni podaci). Filteri se najčešće izvršavaju istovremeno i inkrementalno. Svojstva filtera mogu odrediti format ulaznih ili izlazni podataka i izvršenu transformaciju unutar filtera. Cevi, predstavljaju konektor koji služi da podatke sa izlaznog porta filtera prebaci na ulazni port drugog filtera. Cev omogućava jedinični ulaz podataka, čuva sekvencu prolaska podataka i ne menja podatke koji kroz nju prolaze. Svojstva cevi mogu odrediti veličinu bafera, protokol interakcije i format podataka koji prolazi kroz cev.
Relacije	Veza konekcije (attachment) povezuje izlazne portove filtera sa ulazom u cev i ulazne portove filtera sa izlazom iz cevi.
Ograničenja	Cevi povezuju izlazni port filtera sa ulaznim portom filtera. Povezani filteri moraju koristiti isti tip podataka koji između njih prolazi kroz cev. Specijalizacija stila može ograničiti asocijaciju komponenti na linearnu sekvencu, ponekad nazvanu cevovod. Ostale specijalizacije mogu propisati da komponente imaju određene nazive portova kao što su stdin, stdout,stderr port UNIX filtera
Upotreba	Poboljšanje ponovne upotrebe jer su filteri nezavisni Poboljšanje propusnosti kroz paralelizaciju obrade podataka Pojednostavljenje razmišljanja o ukupnom ponašanju sistema

Slika 3.1 Elementi, relacije i svojstva stila cevi i filtera [Izvor: Clements [2]\

POKAZNI PRIMER - UPOTREBA STILA CEVI I FILTERA

Primer prikazuje pretragu i rezervaciju sobe u hotelu korišćenjem stila cevi i filtera u aplikaciji.

U stilu cevi i filtera znatno je olakšana radnja pretrage što smanjuje vreme rezervacije. Na slici ispod dat je primer cevi i filtera pri traženju sobe. Kao parametri unose se VIP status (true/ false) broj kreveta u sobi kao i dostupnost sobe (u slučaju da je rezervacija za neki dalji period, iako je soba trenutno nedostupna, željeni datum i dalje može biti nakon njenog oslobađanja).



Slika 3.2 Primer rezervacije hotelske sobe korišćenjem stila cevi i filtera

Poglavlje 4

Stil klijent-server

PREGLED STILA POZIV-POVRATAK

Kao i kod drugih stilova poziv-povratak, komponente klijent-server stila interaguju kroz zatraživanje usluga od drugih komponenti.

Komponente koje su podnosioci zahteva se nazivaju klijenti a komponente pružaoci usluga su serveri. Serveri mogu pružati skup usluga preko jednog ili više portova. Neke komponente mogu delovati kao klijenti i serveri u softverskoj arhitekturi. Takođe, može biti jedan ili više servera. Primeri klijent-server arhitekture su:

- informacioni sistemi koji se pokreću na lokalnim mrežama gde su klijenti GUI aplikacije a sever sistem za upravljanje bazom podataka
- veb aplikacije na kojima klijenti rade na veb pretraživačima a komponente se pokreću na veb serveru (primer može biti Tomcat)

Glavni tipovi komponenti su klijenti i serveri. Konektor koji se koristi je konektor za upit/odgovor. Server komponente imaju portove koji opisuju usluge koje pružaju. Klijenti imaju portove koji opisuju usluge koje zahtevaju.

ELEMENTI, RELACIJE I SVOJSTVA STILA KLIJENT-SERVER

Stil klijent-server predstavlja sistemski prikaz koji razdvaja klijentske aplikacije od usluga koje koriste.

Stil podržava ponovnu upotrebu komponenti i zajedničkih usluga. Serverima može pristupiti bilo koji broj klijenata, relativno je lako dodati nove klijente u softversku arhitekturu. Takođe, serveri mogu biti replicirani da podrže skalabilnost ili dostupnost. Klijenti i serveri su često grupisani i raspoređeni na različitim mašinama u distribuiranom okruženju u svrhu formiranja višestruke hijerarhije, ukoliko je to potrebno.

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

Pregled	Klijent-server
Elementi	Klijent, predstavlja komponentu koja poziva servise serverske komponente Server, predstavlja komponentu koja omogućava servise klijentskoj komponenti. Svojstva se razlikuju u zavisnosti od planova projektanta softvera ali najčešće uključuju informacije o prirodi portova servera (kao što je: „koliko klijenata može biti prikačeno na server“) i karakteristike koje utiču na performanse (maksimalni broj poziva). Konektor za upit/odgovor, koristi ga klijent za pozivanje servisa na serveru. Konektori za upit/odgovor imaju dve uloge: ulogu upita i ulogu odgovora. Svojstva konektora uključuju tip poziva (lokalni ili udaljeni) i tip podatka, da li su podaci šifrovani ili ne.
Relacije	Veza konekcije (attachment) povezuje klijentske portove za upit servisa sa upitom konektora I serverski port za odgovor servisa sa odgovorom konektora
Ograničenja	Klijenti su povezani sa serverima kroz upit/odgovor konektora Server komponente mogu biti klijenti drugih komponenti Specijalizacija može uključiti sledeće: • Broj konekcija na određeni port • Dozvoljene relacije na određeni server • Komponente mogu biti organizovane u redovima
Upotreba	Omogućavanje izmena i ponovnog korišćenja servisa Poboljšanje skalabilnosti i mogućnosti replikacije servera Analiziranje zavisnosti i bezbednosti komponenti

Slika 4.1 Elementi, relacije i svojstva stila klijent-server [Izvor: Clements [2]]

POKAZNI PRIMER - UPOTREBA STILA KLIJENT-SERVER

Za komponentu klijenta i servera, slanje i primanje poruka su zasebni koraci.

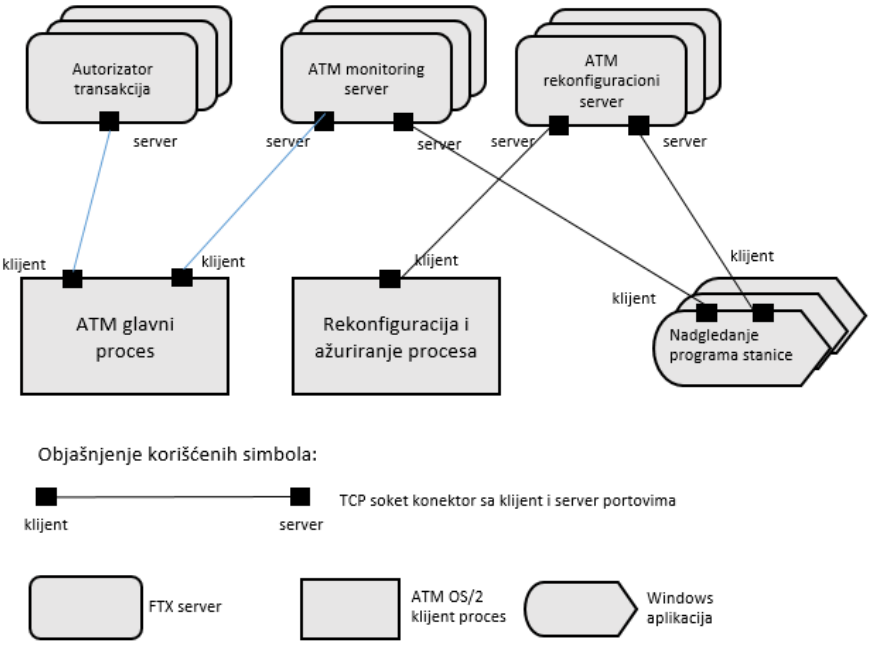
Na slici 2 je prikazan klijent-server stil softverske arhitekture za ATM uređaj. U ovoj arhitekturi postoje tri vrste komponenti:

- FTX server koji pokreće procese u pozadini i kreira portove kojima pristupaju klijenti na osnovu svojih zahteva.
- ATM OS/2 klijentski procesi koji se pokreću na bankomatima i koji zahtevaju procese od serverske komponente.
- Windows aplikacija komponenta klijenta, program koji je razvijen za praćenje grupe bankomata sa svoje radne stanice.

U ovakvom prikazu softverske arhitekture, TCP konektor se koristi za komunikaciju servera i distribuirane aplikacije. Klijent poziva operaciju kako bi primio odgovor od servera. . Implementacija konektora mora da se bavi korelacijom poruka zahteva i poruka odgovora kao i vremenskom intervencijom i mogućim greškama u komunikaciji.

Glavni proces bankovnog sistema na ATM uređaju šalje zahteve za autorizaciju bankovnih transakcija koji odgovaraju operacijama korisnika (depozit ili podizanje novca). Takođe, vrši slanje poruke monitoringu o stanju ATM uređaja (senzora, potrošnog materijala). Komponenta rekonfiguracije i ažuriranja procesa šalje zahteve serveru za rekonfiguraciju ATM-a. Rekonfiguracija predstavlja ažuriranje koje mogu vršiti osoblje zaposleno u banci kroz

komponentu za nadgledanje. Komponenta za nadgledanje šalje periodične zahteve u cilju dobijanja statusa bankomata.



Slika 4.2 Primer klijent-server stila [Izvor: Clements [2]]

CLIENT-SERVER ARCHITECTURE - SOFTWARE ENGINEERING (VIDEO)

Trajanje: 6:17 minuta.

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

Poglavlje 5

Peer-to-peer stil

ELEMENTI, RELACIJE I SVOJSTVA STILA PEER-TO-PEER

Prilikom upotrebe stila peer-to-peer potrebno je koristiti propisane elemente i relacije za pravilan prikaz softverske arhitekture.

Tipovi komponenti ovog stila su peer-ovi (identične komponente) koji su uglavnom nezavisni programi koji se pokreću na mrežnim čvorovima. Tip konektora koji se koristi je konektor poziv-povratak. Razlika između klijent server stila i ovog stila je da interakcija može biti inicirana od bilo koje komponente, jer svaka komponenta može delovati kao klijent ili server.

Pregled	Peer to peer
Elementi	Peer komponenta Poziv-povratak konektor, koji se koristi za povezivanje na peer mrežu, pretragu drugih peer komponenti i za zahtevanje servisa sa drugih peer komponenti.
Relacije	Veza konekcije (attachment) povezuje peer-ove sa poziv-povratak konektorom.
Ograničenja	Ograničenja mogu biti postavljena na dozvoljeni broj konekcija na određenom portu Peer komponente mogu omogućiti usmeravanje, indeksiranje i pomoć drugoj peer komponenti Specijalizacija može nametnuti ograničenja vidljivosti određenih komponenti u softverskoj arhitekturi
Upotreba	Omogućavanje poboljšane dostupnosti komponenti unutar sistema Poboljšanje skalabilnosti Omogućavanje distribuiranih sistema (trenutno deljenje datoteka, poruka i računarske mreže)

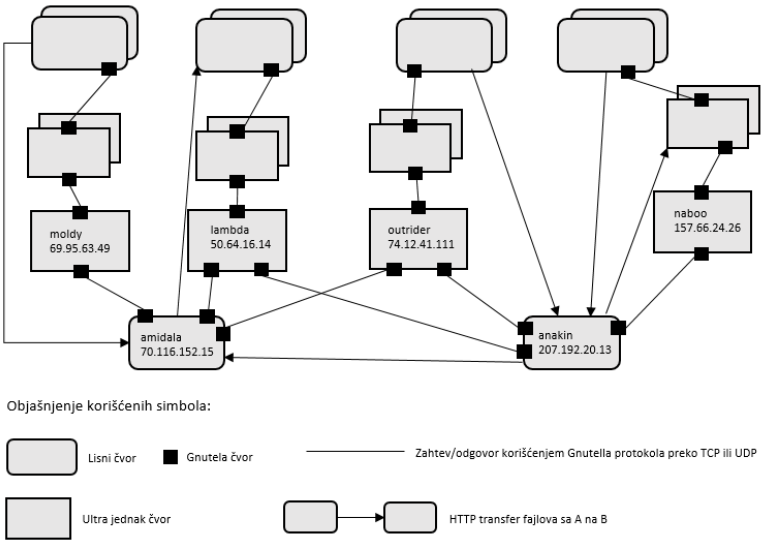
Slika 5.1 Elementi, relacije i svojstva stila peer to peer [Izvor: Clements [2]]

POKAZNI PRIMER - UPOTREBA STILA PEER-TO-PEER

Dat je primer upotrebe stila peer-to-peer na softversku arhitekturu konkretnog sistema.

Na primeru koji je dat na slici 2, prikazan je deo jednak-jednakom prikaza (engl "peer to peer") mreže Gnutela korišćenjem neformalne notacije komponente i konektora. Identifikovane su dva komponente koje su povezane sa drugim komponentama u cilju traženja određenih

datoteka. Komponente (engl. peer) se identifikuju po svojoj IP adresi a protokoli između komponenti prenose podatke preko UDP ili TCP portova. Komponenta može pretraživati druge komponente prosleđivanjem zahteva. Sve komponente koje imaju pozitivne rezultate pretrage šalju povratnu informaciju komponenti koja je inicirala pretragu uz slanje svoje IP adrese i broja porta. Tada, komponenta koja je poslala zahtev uspostavlja komunikaciju sa svim komponentama koje sadrže pozitivan rezultat i iniciraju prenos podataka kroz HTTP protokol. Komponenta ultra jednak čvor (engl. "ultrapeer") se koristi za usmeravanje zahteva za pretragu i odgovorni su za sve lisne čvorove (engl. "leaf peer") komponente koje su konektovane za njih. Na slici 2 su predstavljeni: "moldy", "lambda", "outrider" i "naboo" kao "ultrapeer" komponente i "amidala" i "anakin" kao "leaf peer" komponente u softverskoj arhitekturi.



Slika 5.2 Primer peer to peer stila [Izvor: Clements [2]]

Poglavlje 6

Servisno-orijentisani stil

PREGLED SERVISNO-ORIJENTISANOG STILA SOFTVERSKE ARHITEKTURE

Glavna prednost servisno-orijentisane arhitekture je interoperabilnost.

Primena servisno-orijentisane arhitekture omogućava sledeće infrastrukturne usluge:

- servisni poziv može usmeravati poruke između potrošača usluga i pružaoca usluga. Takođe, ESB servisna magistrala može pretvoriti poruke iz jednog protokola ili tehnologije u drugu, vršiti transformacije podataka (izmenu formata sadržaja, razdvajanje ili spajanje sadržaja).
- transparentnost lokacije pružaoca usluga, servisni registar se može koristiti u SOA arhitekturi. Registar predstavlja komponentu koja omogućava da se usluge registruju a zatim da se pretražuju u toku izvršavanja. Takođe, moguće je povećati mogućnost proširivanja tako što će lokacija pružaoca usluga biti transparentna za sve potrošače.
- server za orkestraciju (rukovođenje) predstavlja posebnu komponentu koja izvršava skripte nakon pojave određenog događaja (primljen je zahtev za porudžbinu). Zatim, server za orkestraciju pokreće interakciju između različitih potrošača i proizvođača usluga u SOA sistemu.

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

ELEMENTI, RELACIJE I SVOJSTVA SERVISNO-ORIJENTISANOG STILA

Servisno-orijentisana arhitektura nudi potrebne elemente za interakciju sa eksternim uslugama dostupnim preko interneta.

Pružaoci usluga i potrošači usluga mogu da rade na različitim platformama i da se integrišu u različite sisteme i stare sisteme. Komponente kao što su registar ili ESB takođe omogućavaju dinamičku rekonfiguraciju koja može biti korisna kada postoji potreba da se zamene ili dodaju nove komponente bez prekida rada sistema.

Pregled	Servisno-orijentisani stil
Elementi	Pružalac usluga (service providers), omogućava jednu ili više usluga kroz objavljeni interfejs. Svojstva se razlikuju od tehnologije implementacije (EJB, ASP.NET) ali takođe mogu uključiti i performanse, ograničenja, ovlašćenja, dostupnost i troškove implementacije. U nekim slučajevima, svojstva su definisana kroz sporazum na nivou usluga (service-level agreement, SLA). Korisnici usluga, zahtevaju usluge direktno ili putem posrednika ESB, posrednički element koji može da se menja i transformiše poruke između pružaoca usluga i potrošača Registar usluga (Registry of services), registar usluga koji mogu koristiti pružaoci usluga za registrovanje njihovih usluga kao i korisnici usluga za potražnju određenih usluga Server za organizaciju (Orchestration server), organizuje interakciju između korisnika usluga i pružaoca usluga uz definisana poslovna pravila. SOAP konektor, koji koristi SOAP protokol za sinhronu komunikaciju između veb servisa, najčešće preko HTTP-a. Rest konektor, koristi osnovne zahtev/odgovor operacije HTTP protokola Konektor za razmenu poruka (Messaging connector), koji koristi sistem poruka za omogućavanje razmene od tačke do tačke.
Relacije	Veza konekcije omogućava priključivanje različitih tipova portova sa odgovarajućim konektorima.
Ograničenja	Korisnici usluga su povezani sa pružaocima usluga preko posredničkih elemenata (kao što je ESB, registar ili BPEL server). ESB posrednički elementi dovode do primene topologije na čvorištu. Specifični SOA šabloni nameću dodatna ograničenja
Upotreba	Dozvoljena je interoperabilnost distribuiranih komponenti koje rade na različitim platformama ili na Vebu. Omogućena je interagicija sistema. Omogućena je dinamička rekonfiguracija sistema.

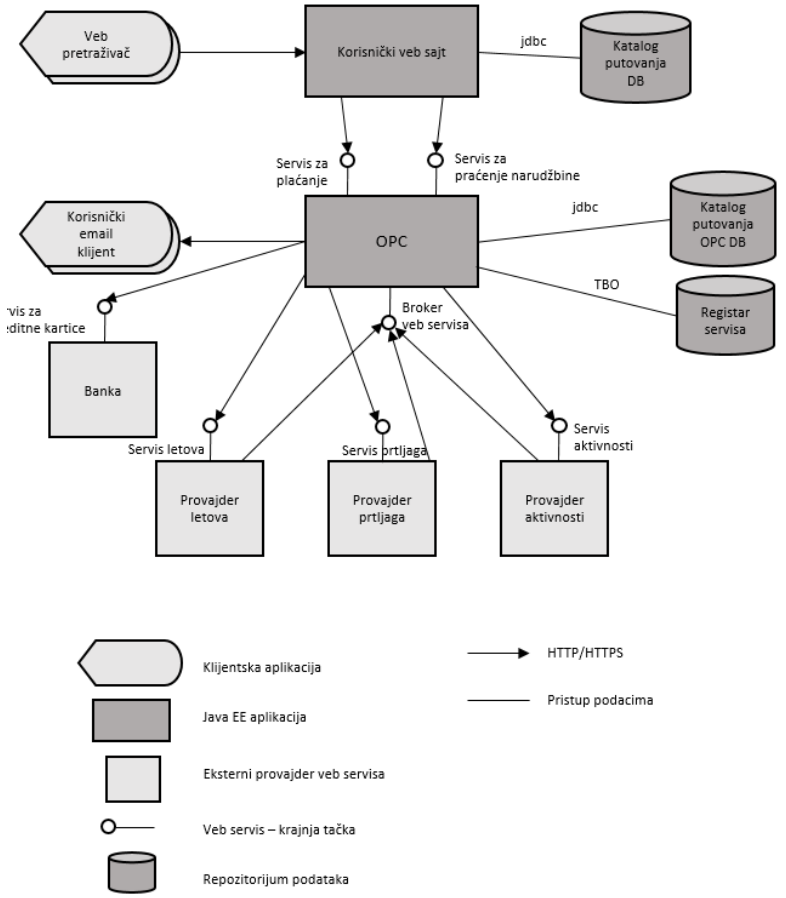
Slika 6.1 Elementi, relacije i svojstva servisno-orijentisanog stila [Izvor: Clements [2]]

POKAZNI PRIMER - UPOTREBA SERVISNO-ORIJENTISANOG STILA

Dat je primer upotrebe servisno-orijentisanog stila softverske arhitekture na konkretnom sistemu.

Primer na slici 2 prikazuje servisno-orijentisani pogled na sistem "Adventure builder". Sistem podržava rad izmišljene kompanije koja putem interneta prodaje avanturističke pakete za odmor u raznim delovima sveta. Sistem se sastoji iz četiri osnovne funkcionalnosti. Korisnik može posetiti "Korisnički veb sajt" i pregledati katalog putovanja u koje spadaju i letovi do određenih odredišta, pregled smeštaja i aktivnosti koje se mogu unapred kupiti ("Provajder aktivnosti"). U aktivnosti spadaju biciklizam, ribolov, surfovanje, turistički obilazak balonom i ronjenje. Takođe, korisnik može samostalno kreirati svoje putovanje odabirom prevoza, smeštaja i aktivnosti kroz opisani sistem. Korisnik može izvršiti naručivanje paketa za odmor. Obrada navedenog naručivanja zahteva od sistema komunikaciju sa više spoljnih entiteta koji su predstavljene na slici 2. Entiteti su banka koja obezbeđuje plaćanje kupcima, provajder letova koji omogućava let do destinacije, provajder prtljaga koji upisuju i smeštaju prtljag pre i nakon leta kao i provajder aktivnosti u okviru koga korisnik može odabrati aktivnosti na odmoru. Komunikacija sa eksternim entitetima može trajati od nekoliko sati do nekoliko dana, zavisno od ažurnosti navedenih entiteta. Interni sistem (OPC) periodično komunicira sa ostalim servisima u cilju ažuriranja kataloga sa novim ponudama.

Sistem interaguje preko SOAP veb usluga sa nekoliko drugih eksternih proizvođača usluga. Spoljne komponente mogu biti sistemi koji preko SOAP konektora komuniciraju sa drugim komponentama sistema.



Slika 6.2 Primer servisno-orijentisanog stila [Izvor: Clements [2]]

SERVICE-ORIENTED ARCHITECTURE (VIDEO)

Trajanje: 9:04 minuta.

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

Poglavlje 7

Stil objavi-pretplati se

PREGLED STILA OBJAVI-PRETPLATI SE

Unutar stila objavi-pretplati se komponente komuniciraju putem najavljenih događaja.

Takođe, komponente mogu biti "pretplaćene" na određeni skup događaja unutar sistema. Stil objavi-pretplati se koristi za slanje događaja nepoznatom skupu primaoca. Zbog toga je skup primalaca događaja nepoznat komponenti koja proizvodi događaje. Na taj način obezbeđena je ispravnost proizvođača koji ne zavisi od primalaca a novi primaoci događaja mogu se dodati bez modifikovanja proizvođača.
Stil objavi-pretplati se često koristi za razdvajanje korisničkih interfejsa iz aplikacije. Interfejsi se takođe mogu koristiti za interakciju alata u okruženju za razvoj softvera gde alati interaguju objavljivanjem događaja koji pokreću pozivanje drugih alata.

ELEMENTI, RELACIJE I SVOJSTVA STILA OBJAVI-PRETPLATI SE

Primena stila objavi-pretplati se zavisi od performansi infrastrukture sistema.

Glavni oblik konektora u ovom stilu je često magistrala i komponente postavljaju događaj na magistralu nakon objave a zatim konektor šalje događaj ostalim pretplaćenim komponentama u sistemu.

Primeri sistema koji koriste stil objavi-pretplati se su:

- grafički korisnički interfejs, gde se korisničke ulazne akcije tretiraju kao događaji koji su usmereni na odgovarajuće upravljače za unos
- aplikacije zasnovane na MVC šablonu pri čemu obaveštavaju komponente kada se stane objekta promeni
- proširiva razvojna okruženja u kojima se alati kodiraju kroz određene događaje
- mejl liste, gde se skup pretplatnika može registrovati za određene teme
- društvene mreže, gde se prijatelji obaveštavaju kada se promene pojave na određenoj stranici

Pregled	Objavi/pretplati se
Elementi	Svaka komponenta i konektor komponenta sa minimum jednim objavi ili pretplati se portom. Svojstva se razlikuju ali treba uključiti sve događaje koji se objavljuju ili na koje je komponenta pretplaćena uz uslove koji su postavljeni. Konektor objavi/pretplati se, koji će objaviti i slušati uloge za komponente koje je potrebno objaviti ili komponente koje je potrebno pretplatiti na određeni događaj.
Relacije	Veza konekcije koja vrši povezivanje komponenti sa objavi/pretplati se konektorom (kroz definisanje komponenti koje objavljuju događaje ili komponente).
Ograničenja	Sve komponente su povezane sa distributerom događaja koji može biti predstavljen kao magistrala ili kao konektor ka komponenti. Portovi za objavljivanje su povezani sa ulogama dok su portovi za mogućnost „pretplati se na događaj“ povezani sa ulogom osluškivanja. Ograničenja mogu ograničiti koje tačno komponente sistema mogu „slušati“ određeni događaj bez obzira da li komponenta može da sluša svoje događaje ili koliko komponenti za objavu postoji unutar sistema. Komponenta može biti i izdavač događaja (objaviti događaj) i pretplatnik događaja (pretplatiti se na događaj) tako što će imati portove oba tipa.
Upotreba	Slanje događaja nepoznatim primaocima, izolovanje proizvođača događaja od potrošača događaja Obezbeđivanje osnovnih funkcionalnosti za GUI šablone, liste mejlova kao i društvene mreže

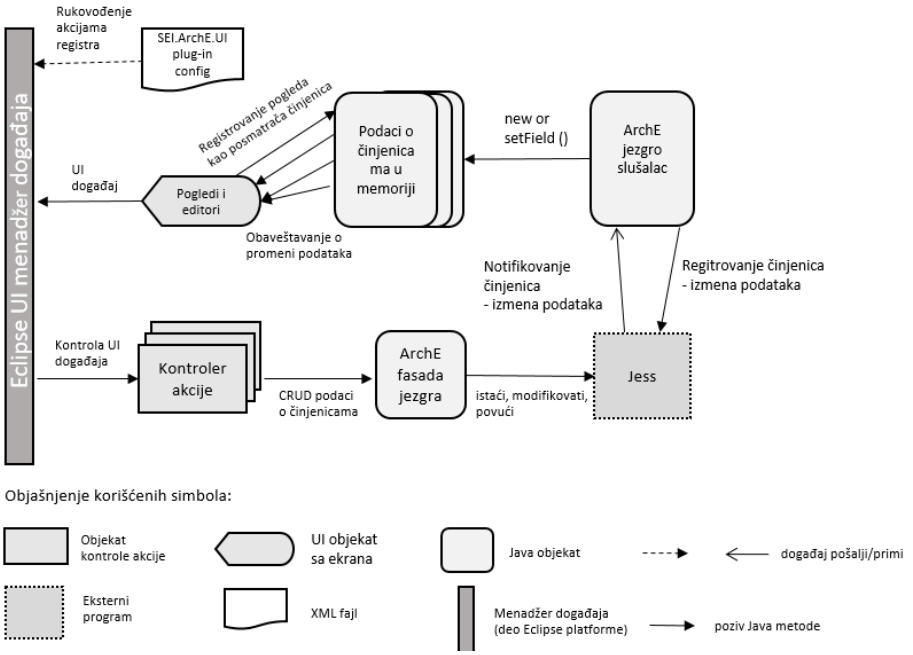
Slika 7.1 Elementi, relacije i svojstva stila objavi-pretplati se [Izvor: Clements [2]]

POKAZNI PRIMER - UPOTREBA STILA OBJAVI-PRETPLATI SE

Dat je primer upotrebe stila objavljivanje-pretplata na ArchiE alatu

Na slici 2 je prikazan primer "SEI ArchiE alata". Predstavljena arhitektura ima tri različite interakcije za objavljivanje i pretplatu:

- Eclipse UI menadžer događaja koji predstavlja magistralu događaja za događaje na korisničkom interfejsu (kao što je klik mišem). Navedeni menadžer događaja beleži akcije korisnika na korisničkom interfejsu i komponente koje pružaju odgovor akcijama korisnika.
- Podaci koji se koriste unutar "ArchE" alata se skladište u pravilo nazvano "Jess". Elementi podataka su nazvani činjenice. Kada akcija korisnika kreira, ažurira ili obriše činjenicu ta akcija pokreće sledeći događaj unutar sistema. Izmenjena činjenica se beleži unutar sistema kao događaj i takav događaj može inicirati pokretanje drugog događaja. Takođe, "Jess" se može okarakterisati i kao magistrala događaja koja pokreće rad drugih komponenti sistema.
- "ArchE" zadržava kopije svih činjenica i događaja koji su izvršeni unutar sistema. Kopije predstavljaju Java objekte. Korisnik komunicira sa objektima kroz korisnički interfejs i forme koje su mu na raspolaganju gde može izmeniti činjenice pokretanjem događaja brisanja, ažuriranja ili dodavanja.



Slika 7.2 Primer stila objavi-pretplati se [Izvor: Clements [2]]

Poglavlje 8

Stil deljenih podataka

ELEMENTI, RELACIJE I SVOJSTVA STILA DELJENIH PODATAKA

Stilovi skladišta sadrže jednu ili više komponenti koje se zovu skladišta (repozitorijumi) koji sadrže veliki broj podataka.

Ostale komponente ovog stila vrše čitanje i upis podataka u skladišta. Za pristup skladišta posreduje softver koji se zove sistem za upravljanje bazama podataka (DBMS) i koji obezbeđuje interfejs za povratni poziv za preuzimanje i manipulaciju podacima. Stil skladišta omogućava pristupačnost podataka i obaveštavanje drugih komponenti ukoliko se dešavaju izmene podataka u skladištu.
Unutar stila deljenih podataka najčešće postoji jedno skladište podataka i više komponenti koje pristupaju skladištu.

Pregled	Stil deljenih podataka
Elementi	Komponenta skladišta (Repository component), Svojstva uključuju tipove skladištenih podataka kao i mogućnosti dalje distribucije Komponenta za pristup i prenos podataka (Data accessor component), Komponenta za čitanje i pisanje podataka (Data reading and writing connector), važno svojstvo je da li je konektor transakcijski ili ne
Relacije	Veza konekcije određuje kojim priključcima je povezana komponenta za pristup i prenos podataka sa skladištem podataka.
Ograničenja	Komponenta za pristup i prenos podataka interaguje sa skladištima podataka.
Upotreba	Omogućavanje da više komponenti pristupi trenutnim podacima Omogućavanje poboljšanja modifikacije kroz razdvajanje komponenti koje proizvode podatke od potrošača podataka.

Slika 8.1 Elementi, relacije i svojstva stila deljenih podataka [Izvor: Clements [2]]

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

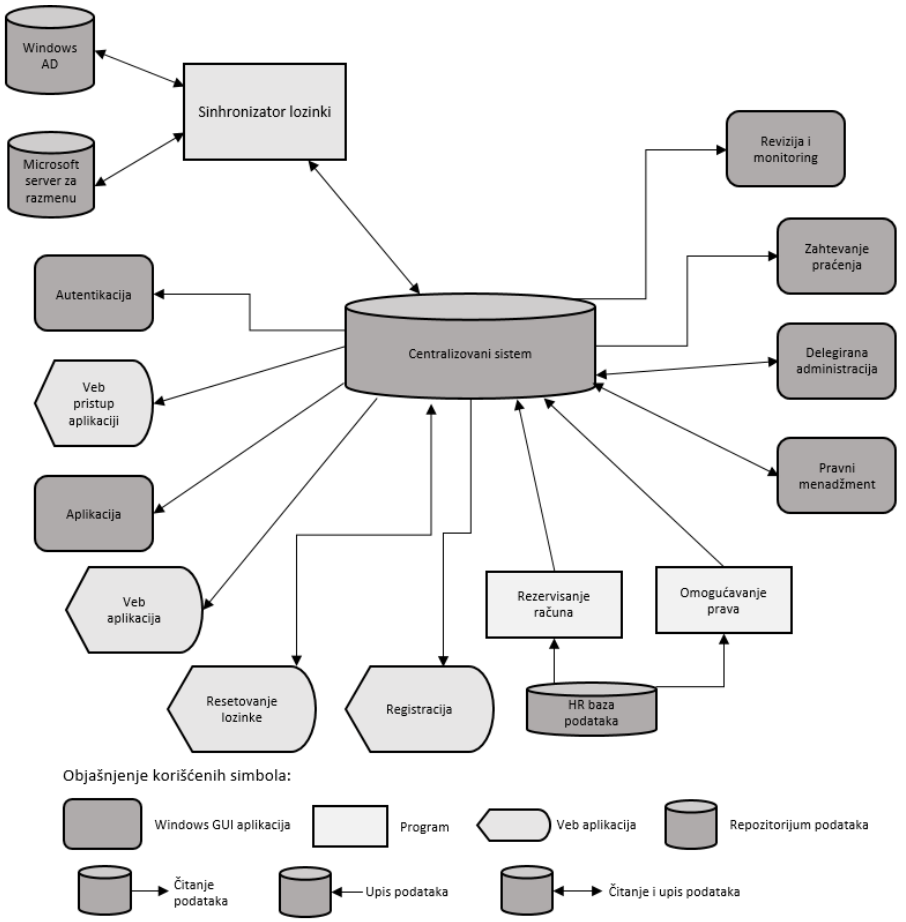
POKAZNI PRIMER - UPOTREBA STILA DELJENIH PODATAKA

Dat je primer upotrebe stila deljenih podataka na konkretnom sistemu.

Na slici 2 prikazan je dijagram stila deljenih podataka za sistem korporativnog upravljanja pristupom. Moguće je pristupiti na tri različita načina:

- Windows aplikacija
- Veb aplikacija
- Programi ili skripte koje se pokreću u pozadini bez korisničkog interfejsa

Dijagram prikazuje deljene podatke između delova sistema. Centralni deo je "centralizovani sistem", prikazan na dijagramu, koji predstavlja skladište korisničkih naloga, lozinki, definisanih grupa korisnika, uloga, dozvola za pristup i drugih relevantnih informacija. Navedene podatke centralizovani sistem dobija iz drugih delova sistema i spoljnih skladišta podataka. Korisnički ID dobija se u sinhronizaciji sa spoljnim skladištima koja se nalaze u gornjem levom uglu dijagrama. Računi zaposlenih se kreiraju ili deaktiviraju na osnovu promena u HR bazi podataka u kojoj su skladišteni. Takođe, dozvole za pristup korisnika se izdaju ili ukidaju na osnovu podataka iz HR baze podataka.



Slika 8.2 Primer stila deljenih podataka [Izvor: Clements [2]]

Poglavlje 9

Pokazna vežba - stilovi komponente i konektora

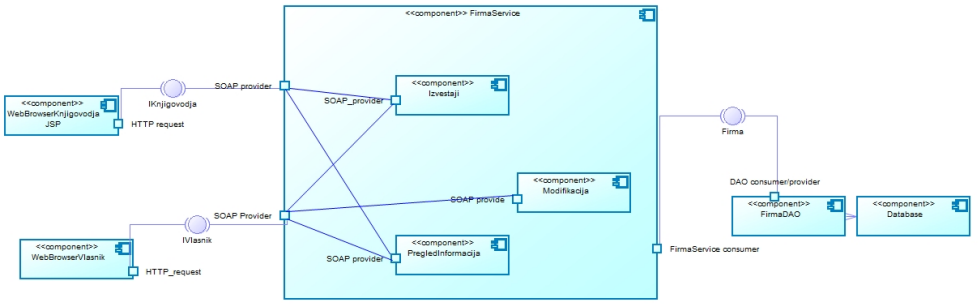
PRIKAZ SOFTVERSKE ARHITEKTURE SOFTVERA ZA VOĐENJE KNJIGOVODSTVA UPOTREBOM STILA KOMPONENATA I KONEKTORA

Dat je primer upotrebe stila komponente i konektora na softveru za vođenje knjigovodstva.

Slika 1 prikazuje upotrebu stila komponente i konektora gde su predstavljene sledeće komponente:

- Izveštaji
- Pregled informacija
- Modifikacija
- FirmaDAO
- Database
- WebBrowserKnjigovođa JSP i WebBrowserVlasnik JSP (15min)

Preporuka je da se za predstavljanje stilova povezivanja koristi Component diagram (preporučeni alati PowerDesigner ili draw.io).

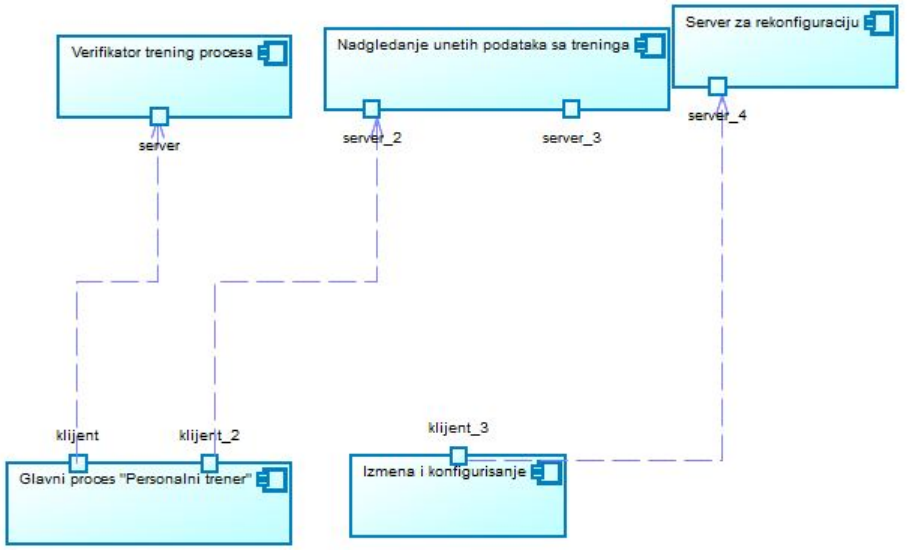


Slika 9.1 Primer upotrebe stila komponente i konektora na softveru za vođenje knjigovodstva [Izvor: Nebojša Gavrilović]

PRIKAZ SOFTVERSKE ARHITEKTURE APLIKACIJE PERSONALNI TRENER KROZ KLIJENT-SERVER STIL

Prikazana je primena stila klijent-server na softversku arhitekturu aplikacije "Personalni trener".

Na slici 2 prikazana je softverska arhitektura aplikacije "Personalni trener" prikazana kroz stil klijent server. Na donjem delu slike nalaze se komponente koje se nalaze na klijentskoj strani aplikacije "Glavni proces "Personalni trener" i "Izmena i konfigurisanje". Na serverskoj strani nalaze se tri komponente "Verifikator trening procesa", "Nadgledanje unetih podataka za trening" i "Server za rekonfiguraciju". Komponente su povezane odgovarajućim portovima a dijagram je nacrtan kroz "Component Diagram" u okviru Power Designer-a.(10min)



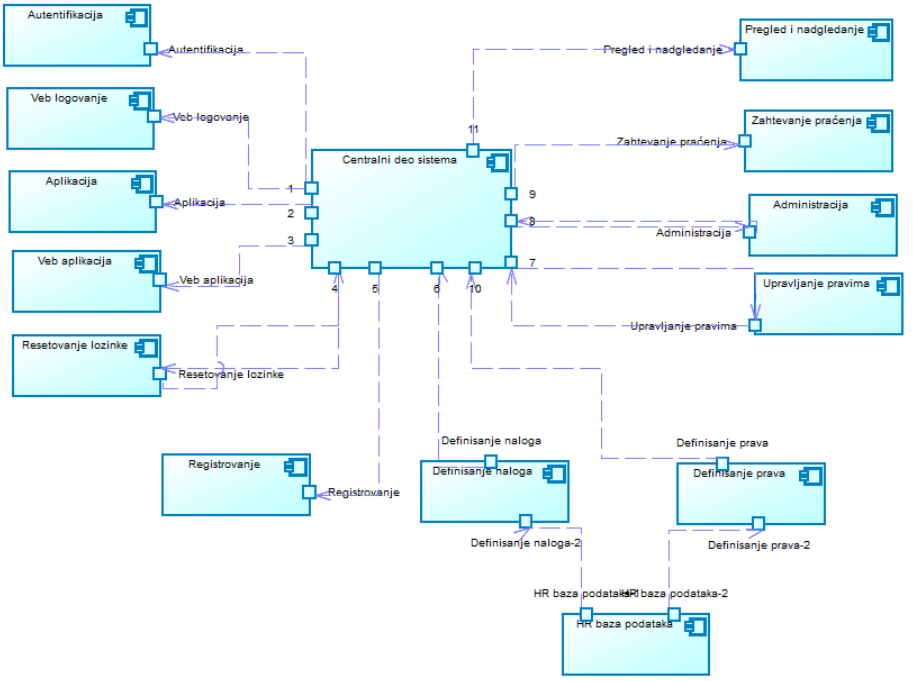
Slika 9.2 Komponente softverske arhitekture aplikacije "Personalni trener" prikazana stilom klijent-server [Izvor: Nebojša Gavrilović]

PRIKAZ SOFTVERSKE ARHITEKTURE APLIKACIJE PERSONALNI TRENER KROZ STIL DELJENIH PODATAKA

Dat je primer softverske arhitekture aplikacije "Personalni trener" kroz stil deljenih podataka

Na slici 2 prikazan je dijagram stila deljenih podataka za softversku arhitekturu aplikacije "Personalni trener". Centralni deo sistema obuhvata skladište podataka za korisničke naloge, lozinke, grupe korisnika, korisničke role, dozvole i dodatne informacije nalozima korisnika. Centralnom delu sistema (skladištu podataka) pristupaju ostale komponente sistema i čitaju ili upisuju podatke (zavisno od relacije koja je predstavljena na dijagramu). Upis podataka u skladište vrše komponente sistema od kojih strelica ide ka centralnom delu sistema a centralni deo sistema čita podatke iz komponenti ka kojima ide strelica definisane relacije.

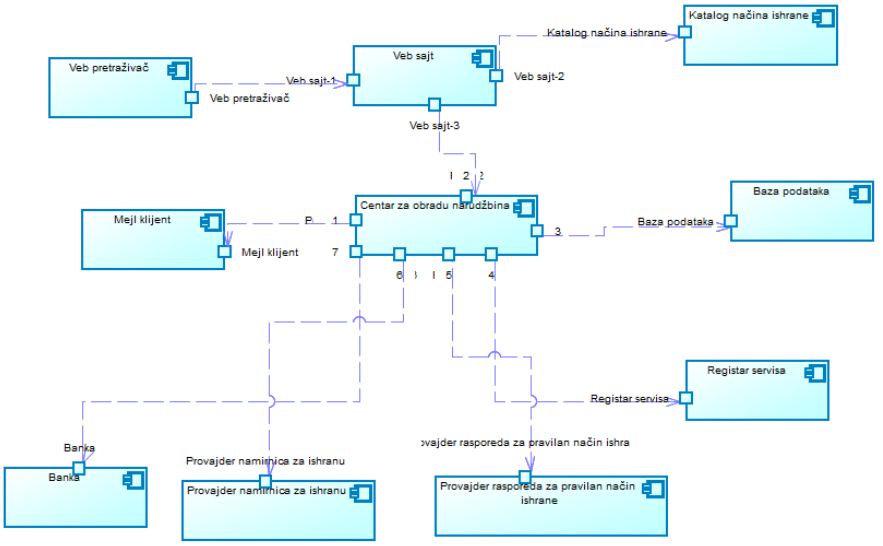
Dijagram prikazuje razmenu podataka između komponenti sistema i izmenu podataka koji se tiču korisničkog naloga. Pored centralnog skladišta tu se nalazi i "HR baza podataka" koja preko komponenti "Definisanje naloga" i "Definisanje prava" komunicira sa centralnim delom sistema. Iz navedene "HR baze podataka" prikupljaju se podaci i prava koje korisnički nalozi mogu da imaju u okviru aplikacije.(15min)



Slika 9.3 Komponente softverske arhitekture prikazane kroz stil deljenih podataka [Izvor: Nebojša Gavrilović]

PRIKAZ SOFTVERSKE ARHITEKTURE APLIKACIJE PERSONALNI TRENER KROZ SERVISNO-ORIJENTISANI STIL

Primena servisno-orijentisanog stila na softversku arhitekturu aplikacije "Personalni trener" data je u nastavku.



Slika 9.4 Prikaz komponenti softverske arhitekture kroz servisno-orijentisani stil [Izvor: Nebojša Gavrilović]

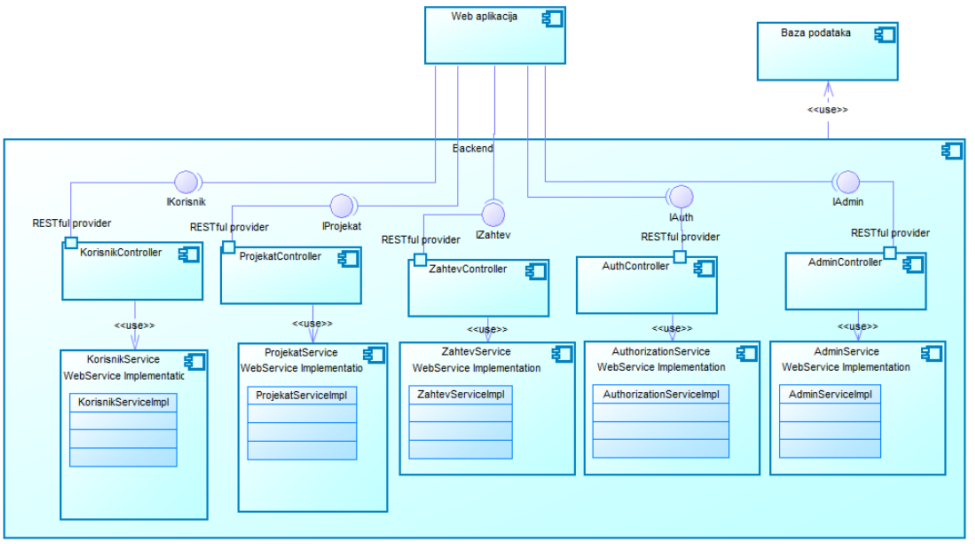
Na slici 3 prikazana je softverska arhitektura aplikacije "Personalni trener" kroz servisno-orijentisani stil. Centralna komponenta predstavlja "Centar za obradu narudžbina" kojoj pristupa korisnik putem komponente "Veb sajt". Korisnik zatim može vršiti odabir rasporeda i saveta za pravilan način ishrane a centralna komponenta komunicira sa eksternim komponentama "Banka", "Provajder namirnica za ishranu" i "Provajder rasporeda za pravilan način ishrane" od kojih dobija servise potrebne za kompletiranje narudžbine korisnika. Korisnik može platiti svojom karticom korišćenjem servisa za plaćanje iz komponente "Banka", može dobiti listu namirnica za raspored pravilnog načina ishrane iz komponente "Provajder namirnica za ishranu" a raspored može dobiti iz servisa komponente "Provajder rasporeda za pravilan način ishrane".

Podaci se skladište unutar baze podataka pored koje postoji i registar servisa koji beleži komunikaciju sa svim servisima. (15min)

PRIKAZ SOFTVERSKE ARHITEKTURE SISTEMA ZA MENTORISANJE KROZ SERVISNO-ORIJENTISANI STIL

Predstavljena je softverska arhitektura sistema kroz servisno-orijentisani stil.

Web aplikacija predstavlja komunicira sa serverom preko odgovarajućih REST servisa. U zavisnosti od podataka koje zahteva korisnik, aplikacije će pozivati odgovarajući servis. Server koristi bazu podataka, preko JDBC konektora, radi ostvarivanja servisa koje nudi.(15min)



Slika 9.5 Softverska arhitektura sistema za mentorisanje prikazana kroz servisno-orijentisani stil [Izvor: Nebojša Gavrilović]

Poglavlje 10

Individualna vežba - Zadaci

ZADATAK ZA SAMOSTALAN RAD

Na osnovu definisanih korisničkih zahteva primeniti različite stilove softverske arhitekture.

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

Korišćenjem opisanog scenarija potrebno je:

- Prikazati softversku arhitekturu sistema kroz stil skladišta (25min)
- Prikazati softversku arhitekturu sistema kroz stil publish-subscribe (20min)
- Prikazati softversku arhitekturu sistema kroz servisno-orijentisani stil (25min)
- Prikazati softversku arhitekturu sistema kroz stil peer-to-peer (15min)

Uporediti modelovane softverske arhitekture i dokumentaciju za svaki modelovani stil. Utvrdite i objasnite stil arhitekture koji najviše odgovara opisanom scenariju. (20min)

ZADACI ZA SAMOSTALAN RAD - STIL CEVI I FILTERA

Primeniti stil cevi i filtera na odabranu softversku arhitekturu.

Zadatak 1: Primeniti stil cevi i filtera na softverskoj arhitekturi proizvoljnog sistema. Modelovati odgovarajući dijagram i pridržavati se definisane notacije za predstavljanje stila cevi i filtera. (30min)

Zadatak 2: Detaljno dokumentovati svaku identifikovanu komponentu u softverskoj arhitekturi i opisati relacije između komponenti. (15min)

ZADACI ZA SAMOSTALAN RAD - STIL KLIJENT-SERVER

Primeniti klijent-server stil na odabranu softversku arhitekturu.

Zadatak 1: Primeniti klijent-server stil na softverskoj arhitekturi proizvoljnog sistema. Modelovati odgovarajući dijagram i pridržavati se definisane notacije za predstavljanje stila klijent-server. (30min)

Zadatak 2: Detaljno dokumentovati svaku identifikovanu komponentu u softverskoj arhitekturi i opisati relacije između komponenti. (15min)
Zadatak 3: Nacrtati ključ u okviru dijagrama softverske arhitekture koji opisuje korišćene simbole unutar dijagrama. (pogledati pokazni primer klijent-server stila)(15min)

ZADACI ZA SAMOSTALAN RAD - PEER TO PEER STIL

Primeniti peer-to-peer stil na odabranu softversku arhitekturu.

Zadatak 1: Primeniti peer-to-peer stil na softverskoj arhitekturi proizvoljnog sistema. Modelovati odgovarajući dijagram i pridržavati se definisane notacije za predstavljanje stila peer-to-peer. (30min)
Zadatak 2: Detaljno dokumentovati svaku identifikovanu komponentu u softverskoj arhitekturi i opisati relacije između komponenti. (15min)
Zadatak 3: Nacrtati ključ u okviru dijagrama softverske arhitekture koji opisuje korišćene simbole unutar dijagrama. (pogledati pokazni primer peer-to-peer stila)(15min)

ZADACI ZA SAMOSTALAN RAD - SERVISNO-ORIJENTISANI STIL

Primeniti servisno-orijentisani stil na odabranu softversku arhitekturu.

Zadatak 1: Primeniti servisno-orijentisani stil na softverskoj arhitekturi proizvoljnog sistema. Modelovati odgovarajući dijagram i pridržavati se definisane notacije za predstavljanje servisno-orijentisanog stila . (30min)

Zadatak 2: Detaljno dokumentovati svaku identifikovanu komponentu u softverskoj arhitekturi i opisati relacije između komponenti. (15min)

Zadatak 3: Nacrtati ključ u okviru dijagrama softverske arhitekture koji opisuje korišćene simbole unutar dijagrama. (pogledati pokazni primer servisno-orijentisanog stila)(15min)

ZADACI ZA SAMOSTALAN RAD - STIL OBJAVI-PRETPLATI SE

Primeniti objavi-pretplati se stil na odabranu softversku arhitekturu.

Zadatak 1: Primeniti objavi-pretplati se stil na softverskoj arhitekturi proizvoljnog sistema. Modelovati odgovarajući dijagram i pridržavati se definisane notacije za predstavljanje servisno-orijentisanog stila. (30min)
Zadatak 2: Detaljno dokumentovati svaku identifikovanu komponentu u softverskoj arhitekturi i opisati relacije između komponenti. (15min)
Zadatak 3: Nacrtati ključ u okviru dijagrama softverske arhitekture koji opisuje korišćene simbole unutar dijagrama. (pogledati pokazni primer stila objavi-pretplati se)(15min)

ZADACI ZA SAMOSTALAN RAD - STIL DELJENIH PODATAKA

Primeniti stil deljenih podataka na odabranu softversku arhitekturu.

Zadatak 1: Primeniti stil deljenih podataka na softverskoj arhitekturi proizvoljnog sistema. Modelovati odgovarajući dijagram i pridržavati se definisane notacije za predstavljanje servisno-orijentisanog stila. (30min)

Zadatak 2: Detaljno dokumentovati svaku identifikovanu komponentu u softverskoj arhitekturi i opisati relacije između komponenti. (15min)
Zadatak 3: Nacrtati ključ u okviru dijagrama softverske arhitekture koji opisuje korišćene simbole unutar dijagrama. (pogledati pokazni primer stila deljenih podataka) (15min)

ZADACI ZA DISKUSIJU NA VEŽBI

Otvorite diskusiju po svakom od dole postavljenih pitanja.

1. Primenom pogleda komponente i konektora moguće je prikazati izvršavanje programa. Koji su elementi softverske arhitekture koje koristi pogled komponente i konektora?(10min)

2. Navesti primer komponente "filter" u okviru stila cevi i filtera koja može biti korišćena u informacionom sistemu univerziteta. (10min)

3. Navedite primer Interoperabilnosti servisno-orijentisane softverske arhitekture.(10min)

4. Da li primena stila deljenih podataka zahteva primenu centralizovanog sistema? Objasniti na primeru.(10min)

5. Da li je moguće klijent-server stil kombinovati sa peer-to-peer stilom? (10min)

Poglavlje 11

Domaći zadatak br.4

PRAVILA ZA IZRADU DOMAĆEG ZADATKA

Student zadatake treba samostalno da uradi i da dostavi asistentu.

Domaći zadatak br.4: Primeniti objavi-pretplati se stil na softverskoj arhitekturi odabranog sistema.
Pri radu, koristite Power Designer.

Domaći zadaci su dobijaju se po definisanim instrukcijama. Domaći zadatak se imenuje: SE311-DZ04-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-DZ04. 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 12

Zaključak

ZAKLJUČAK

Stilovi komponente i konektora mogu se grupisati u više kategorija na osnovu definisanog modela. Svaka od ovih kategorija sadrži niz specifičnih stilova komponente i konektora:

- stil cevi i filtera, vrši se filtriranje procesa serijskog unosa podataka na ulazu u filter i prosleđivanje obrađenih podataka na izlaz
- stil klijent-server omogućava slanje sinhronih zahteva klijentskih komponenti ka komponentama servera
- stil peer-to-peer, komponente zajedno rade u cilju postizanja željenog cilja razmenom sinhronih zahteva/odgovora
- stil servisno-orijentisane arhitekture podrazumeva distribuirane komponente koje deluju kao proizvođači ili korisnici usluga.
- stil objavljivanje-pretplata komponente šalju događaje komponentama koje su pretplaćene na taj događaj
- stil deljenih podataka prikazuje način pristupanja skladištu podataka i funkcije čitanja ili pisanja.

LITERATURA

U ovoj lekciji korišćena je kao obavezna literatura, referenca 2, poglavlje 3 i poglavlje 4(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, poglavlje 3 (strane od 123 do 152), poglavlje 4 (strane od 155 do 189)

Dopunska literatura:

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/