UNIVERZITET METROPOLITAN BEOGRAD

SE311 - PROJEKTOVANJE I ARHITEKTURA SOFTVERA

Stilovi alokacije i hibridni stilovi

Lekcija 05

PRIRUČNIK ZA STUDENTE

SE311 - PROJEKTOVANJE I ARHITEKTURA SOFTVERA

Lekcija 05

STILOVI ALOKACIJE I HIBRIDNI STILOVI

UVOD

Elementi softvera u softverskoj arhitekturi komuniciraju sa ne-softverskim elementima u okruženju u kome se softver razvija, raspoređuje i izvršava. Računarski i komunikacioni hardver, sistemi za upravljanje datotekama i razvojni timovi komuniciraju sa softverskom arhitekturom. Skup struktura koje su potrebne za "razumevanje sistema" uključuje strukture koje prikazuju odnose između komponenata softverske arhitekture i ne-softverskih elemenata. Moguće je:

- izvršiti povezivanje arhitekture softvera i hardvera u cilju analize performansi sistema
- izvršiti povezivanje arhitekture softvera i strukture datoteka koje mogu upravljati sistemom u produkciji
- izvršiti povezivanje arhitekture softvera i razvojnog tima (povezivanje sa delovima tima ili članovima)

Ova lekcija vam obezbeđuje sledeće ishode učenja:

- Razumevanje stilova alokacije i hibridnih stilova, primena stilova alokacije (stila raspoređivanja, stila instalacije i stila dodeljivanja radnih zadataka)
- Razumevanje specijalizacije stilova alokacije i primena drugih stilova alokacije koji se mogu koristiti za prikazivanje softverske arhitekture
- Razumevanje prednosti kombinovanja različitih stilova u cilju detaljnog predstavljanja softverske arhitekture
- Razumevanje i primena 4+1 Kručtenovih pogleda

UVODNI VIDEO LEKCIJE

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

Poglavlje 1

Stilovi alokacije

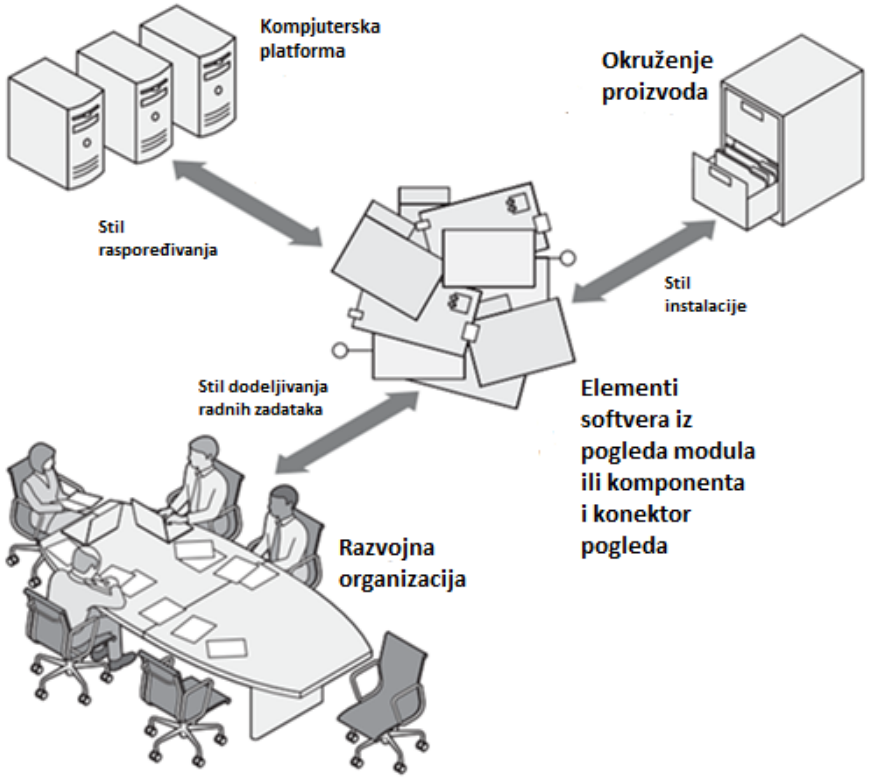
TRI RAZLIČITA STILA ALOKACIJE

Za predstavljanje softverske arhitekture koriste se tri stila alokacije, stil raspoređivanja, stil instalacije i stil dodeljivanja radnih zadataka

Povezivanje softverske arhitekture sa okruženjem je kroz tri različita stila:

1. **Stil raspoređivanja (**deployment style**):** opisuje povezanost softverskih komponenti i konektora sa hardverom kompjuterske platforme na kojoj se softver izvršava
2. **Stil instalacije (**install style**):** opisuje povezanost između softverskih komponenti i struktura fajl sistema u produkcionom okruženju softvera
3. **Stil dodeljivanja radnih zadataka (**work assignement style**):** opisuje povezanost softverskih modula sa ljudima, timovima ili organizacijom posla u toku razvojnog procesa

Na slici 1 prikazana su tri stila alokacije i način na koji su elementi softvera povezani sa svojim okruženjem.



Slika 1.1 Tri stila alokacije [Izvor: Clements [2]]

ELEMENTI, RELACIJE I SVOJSTVA STILOVA ALOKACIJE

Primena stilova alokacije zahteva korišćenje definisanih elemenata, relacija i svojstava.

Relacija koja se koristi u stilu alokacije je dodeljen prema ("allocated to"). Stilove alokacije moguće je primeniti i obrnutim tokom, tako što se elementi okruženja softvera povezuju sa komponentama softverske arhitekture. Element softverske arhitekture može biti dodeljen ka više elemenata okruženja softvera a više elemenata softverske arhitekture može biti dodeljeno jednom elementu okruženja softvera.
Elementi softvera i elementi okruženja softvera imaju jasno definisana svojstva po kojim se vrši raspodela. Najčešće je potrebno izvršiti upoređivanje osobina koje zahteva softverski element sa osobinama koje omogućava element okruženja softvera u cilju provere da li će alokacija biti uspešna ili ne. Primer može biti: " *Ukoliko je potrebno ubrzati odziv određene softverske komponente, potrebno je proveriti da li procesor računara na kome se softver izvršava može da izvrši planirano ubrzanje*". Specifične notacije za stilove alokacije definisane su shodno različitim stilovima.

Pregled	Stilovi alokacije opisuju mapiranje između softverske arhitekture i okruženja softverske arhitekture.
Elementi	Element softvera i element okruženja. Element softvera ima svojstva koja su zahtevana od okruženja. Element okruženja ima svojstva koja se pružaju softveru.
Relacije	Dodeljeno („alocated to“). Element softvera se dodeljuje elementu okruženja. Svojstva zavise od određenog stila koji se primenjuje.
Ograničenja	Zavise od stila koji se primenjuje.

Slika 1.2 Prikaz karakteristika stilova alokacije [Izvor: Clements [2]]

Poglavlje 2

Stil raspoređivanja

PREGLED STILA RASPOREĐIVANJA

Stil raspoređivanja, pored osnovnih prikaza softverske arhitekture, omogućava i detaljno analiziranje sistema u svrhu dobijanja potrebnih informacija o njegovom načinu rada.

Elementi softverske arhitekture, primenom stila alokacije, predstavljeni su kao elementi stila komponenta i konektor. Stil raspoređivanja (deployment style) omogućava dodeljivanje određenih softverskih komponenti hardveru računarske platforme na kojoj se softver izvršava. Cilj primene stila raspoređivanja je da zahtevi izraženi softverskim elementima zadovoljavaju karakteristike hardverskih elemenata.

ELEMENTI STILA RASPOREĐIVANJA

Elementi stila raspoređivanja dele se na elemente softverske arhitekture i elemente softverskog okruženja.

Elementi okruženja softvera su entiteti koji odgovaraju fizičkim jedinicama koje čuvaju, prenose ili računaju podatke. Fizičke jedinice uključuju čvorove procesora (CPU), kanale komunikacije, skladišta memorije i skladišta podataka. Elementi stila raspoređivanja su tipični elementi koji se prikazuju u okviru pogleda komponenta i konektor.
Elementi prikazani kroz stil raspoređivanja pretpostavlja se da se pokreću na računarskoj platformi. Slika 1 prikazuje karakteristike stila raspoređivanja.

Pregled	Stil raspoređivanja opisuje mapiranje između komponenata i konektora unutar softverske arhitekture sa hardverom kompjuterske platforme.
Elementi	Softverski elementi: elementi iz komponenata i konektor pogleda, detaljna dokumentacija o svakom identifikovanom elementu o očekivanim karakteristima zahtevanim od hardvera (kao što je obrada memorije, kapacitet memorije i tolerancija grešaka). Elementi okruženja: hardver računara, procesor, memorija, disk, mreža (ruter, propusni opseg itd.).
Relacije	Dodeljeno („alocated to“). Fizički jedinice na kojima se nalaze softversi elementi u toku izvršenja. Svojstva uključuje da li alokacija može promeniti vreme izvršenja ili ne. Migrira do, kopira-migrira do ili izvršava-menja u slučaju da je alokacija dinamična. Svojstva uključuju okidač koji izaziva migraciju.
Ograničenja	Topologija alokacije je neograničena. Potrebna svojstva softvera moraju biti obezbeđena svojstvima hardvera.

Slika 2.1 Prikaz karakteristika stila raspoređivanja [Izvor: Clements [2]]

RELACIJE STILA RASPOREĐIVANJA

Relacije stila raspoređivanja omogućavaju povezivanje elemenata softvera sa elementima okruženja softvera.

Dodatne relacije koje se mogu koristiti kroz stil raspoređivanja su:

- **migrira ka ("migrates to"):** relacija od softverskog elementa na jednom procesoru do istog softverskog elementa na drugom procesoru. Ovom relacijom ukazuje se na to da softverski element može preći sa procesora na procesor ali da ne postoji istovremeno na oba procesora.
- **kopira migraciju ka ("copy migrates to"):** relacija koja je slična prethodnoj relaciji, samo u ovom slučaju softverski element šalje svoju kopiju novom procesoru dok zadržava kopiju na prvobitnom elementu obrade.
- **izvršava migraciju ka ("execution migrates to"):** relacija koja ukazuje na to da izvršenje kreće od procesora do procesora ali se lokacija koda ne menja.

Kopija procesa postoji na više od jednog procesora ali je samo jedan aktivan u bilo koje određeno vreme.
Takođe je moguće menjati alokaciju tokom vremena kroz ručnu rekonfiguraciju. Ukoliko softverski element zahteva minimalni kapacitet skladištenja, bilo koji element softverskog okruženja koji ima taj kapacitet predstavlja kandidata za validnu alokaciju.
Analizom softverskih elemenata utvrđuju se posebna svojstva koja elementi moraju posedovati. Ako je potrebna analiza kapaciteta memorije, potrebno je da svojstva softverskih elemenata opisuju aspekte potrošnje memorije a da relevantna svojstva elemenata softverskog okruženja prikazuju kapacitete memorije različitih hardverskih entiteta.

SVOJSTVA STILA RASPOREĐIVANJA

Svojstva stila raspoređivanja opisuju elemente softvera i elemente softverskog okruženja.

Svojstva elemenata softverskog okruženja mogu biti:

- **svojstva CPU**. Osobine relevantne za različita svojstva procesora (brzina procesora, broj procesora, kapacitet memorije, veličina keš memorije, brzina izvršavanja instrukcija)
- **svojstva memorije**. Karakteristike koje su relevantne za memorije (veličina memorije, karakteristike brzine)
- **kapacitet diska ili druge memorijske jedinice**. Kapacitet skladištenja i brzina pristupa jedinicama za skladištenje.
- **protok.** Kapacitet prenosa podataka kroz komunikacione kanale
- **tolerancija grešaka.** Višestruke hardverske jedinice mogu obavljati istu funkciju a mogu imati i mehanizam za kontrole prekida.

Svojstsva softverskih elemenata mogu biti:

- **potrošnja resursa.** Primer može biti da proces računanja uzima 32123 instrukcije uvek ili najviše ili u proseku.
- **potrebe za resursima i ograničenjima koja moraju biti zadovoljena**. Primer može biti da softverski element mora biti izvršen za ne više od 0.1 sekunde.
- **kritična sigurnost.** Primer može biti da softverski element mora biti pokrenut u svakom trenutku.
- **okidač migracije.** Primer može biti prebacivanje softverskog elementa iz jednog elementa obrade u drugi

KADA SE KORISTI STIL RASPOREĐIVANJA

Stil raspoređivanja softverske arhitekture omogućava detaljnu analizu performansi, dostupnosti, pouzdanost i i sigurnosti.

U procesu testiranja, stil raspoređivanja služi kako bi se stekao uvid u zavisnosti softverskih elemenata u toku izvršavanja programa. Takođe, moguće je koristiti za procenu troškova u toku procesa nabavke hardvera.
Performanse se, primenom stila raspoređivanja, podešavaju primenom alokacije softverskih elemenata na elemente okruženja softvera (hardver). Na taj način moguće je eliminisati usko grlo na procesorima koji se koriste od različitih softverskih elemenata. Projektant softvera može koristiti dodatni hardver ili zameniti hardverske elemente sa naprednim verzijama ukoliko zahtevi nisu ispunjeni na pravi način.
Dostupnost i pouzdanost direktno utiču na ponašanje sistema u slučaju neispravnih ili neuspešnih elemenata obrade ili komunikacionih kanala. Sigurnost i otpornost sistema određena je od strane konfiguracije hardvera i alokacije softvera na hardver. Ograničenjem dostupnosti servisa može se poboljšati sigurnost samog sistema.

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

POKAZNI PRIMER: STIL RASPOREĐIVANJA - NEFORMALNA NOTACIJA

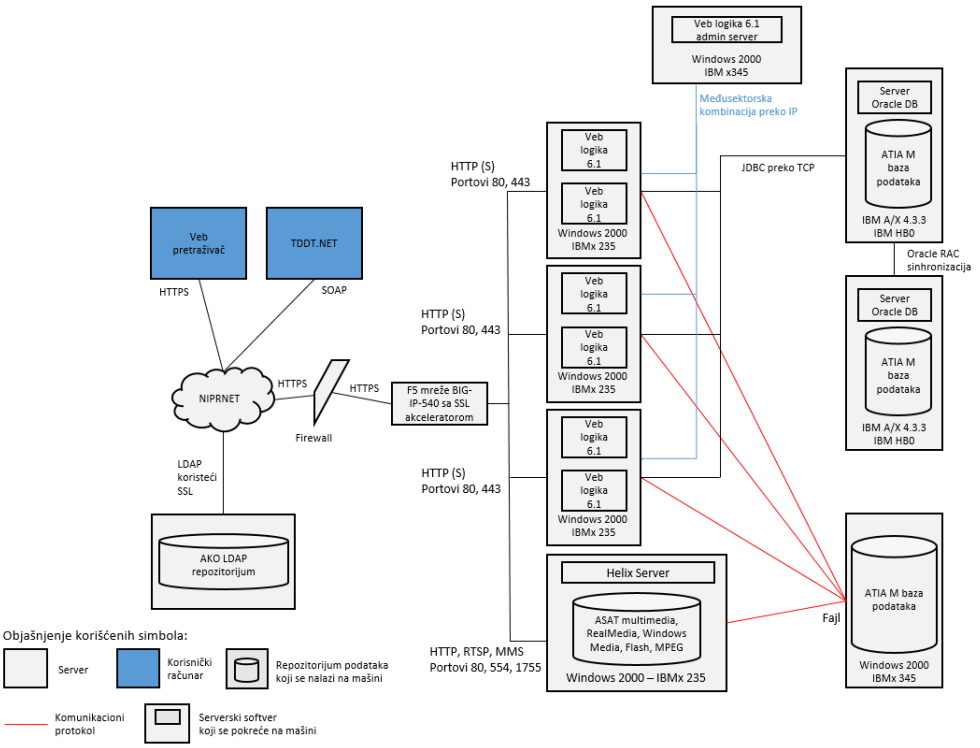
Neformalna grafička notacija sadrži kutije, krugove, linije koje predstavljaju softverske elemente ili relacije između njih.

Često se različitim simbolima predstavljaju i elementi softverskog okruženja. Simboli predstavljaju slike ili oblike hardverskih komponenti. Softverski elementi mogu biti navedeni unutar ili pored hardvera na koji su alocirani kako bi se prikazao alocirani odnos. Ukoliko je struktura primene jednostavna, potrebno je tabelom opisati softverske jedinice i hardverske elemente na kome se softverske jedinice izvršavaju. Na slici 1 prikazan je primer osnovnog prikaza stila raspoređivanja upotrebom neformalne notacije.
Na slici 2 (prikazana na narednoj sekciji) prikazan je stil raspoređivanja u neformalnoj notaciji. Primer se odnosi na američki vojni informacioni sistem (ATIA-M) i koristi prepoznatljive simbole za različite vrste hardvera. Linije za povezivanje su fizički kanali komunikacije koji

omogućavaju komponentama da komuniciraju jedni sa drugima. Raspodela komponenti se prikazuje preklapanjem imena na jednom simbolu koji ih predstavlja. Dodeljivanje konektora vrši se upisivanjem njihovih imena pored kanala koji označava komunikaciju dva elementa na slici. Sistem je razvijen korišćenjem Java EE platforme koja sadrži stotine komponenti (servlete i EJB ("Enterpise Java Beans")) i ima različit prikaz klijenata i servera sa određenim nivoom GUI-a i veb servisnim slojem. Komponente unutar tih slojeva su raspoređene na "WebLogic" što ukazuje predviđena anotacija. Mreža NIPRNET predstavlja mrežu sličnu internetu koja je u vlasništvu američkog ministarstva odbrane.

POKAZNI PRIMER: STIL RASPOREĐIVANJA - NEFORMALNA NOTACIJA - SLIKA

Na slici 2 dat je primer softverske arhitekture upotrebom stila raspoređivanja.



Slika 2.2 Prikaz softverske arhitekture stilom raspoređivanja korišćenjem neformalne notacije [Izvor: Clements [2]]

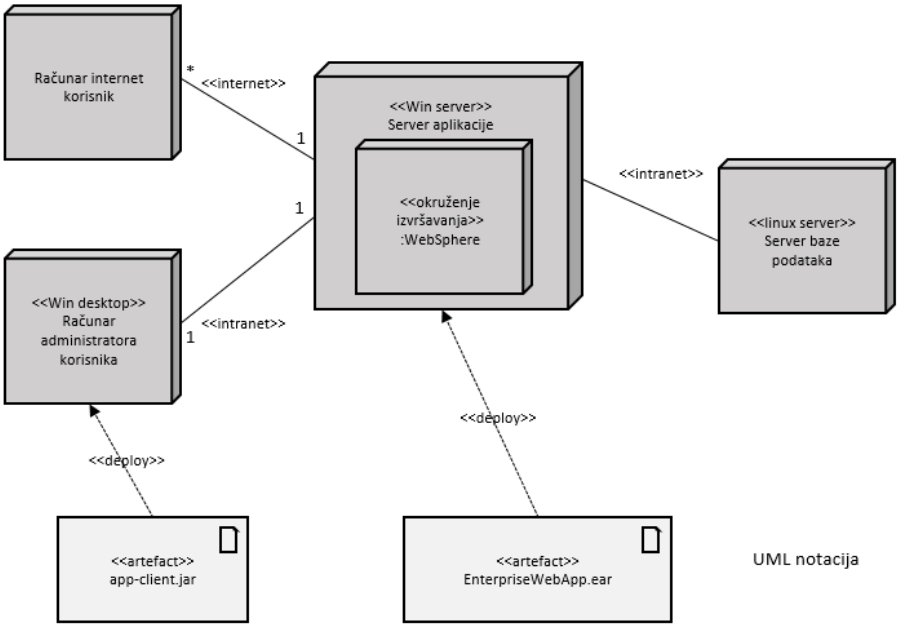
POKAZNI PRIMER: STIL RASPOREĐIVANJA - UML NOTACIJA

Prikaz stila raspoređivanja kroz UML jezik omogućava prikaz čvorova koji su povezani određenim komunikacionim relacijama.

Čvorovi odgovaraju elementima obrade i najčešće imaju memoriju i mogućnost obrade. Komponente mogu biti povezane sa drugim komponentama pomoću relacija zavisnosti. U UML dijagramu za sloj raspoređivanja, komponente mogu sadržati objekte kao i delove komponenti. Migracija komponenti iz čvora u čvor (ili objekta iz komponente u komponentu) prikazuje se kroz stereotip veze <<becomes>>. Čvor je u tom slučaju prikazan kao trodimenzionalna kutija sa opcionim imenom unutar kutije. Moguće je precizno označiti komunikacioni kanal stereotipom asocijacije kao na primer: "<<10-T Ethernet>>", "<<RS-232>>".
Svojstva su predstavljena sa imenima atributa u parovima, kao na primer: "brzinaProcesora=300 mHz", "memorija=128 MB"). Na slici 2 prikazana je UML notacija za stil raspoređivanja softverske arhitekture.

Na slici 3 prikazan je stil raspoređivanja primenjen na softversku arhitekturu koji prikazuje hardversku platformu koja služi za pokretanje Java EE sistema. Relacija "<<deploy>>" prikazuje koji objekti su raspoređeni na koje čvorove. Čvor koji omogućava pokretanje određenih komponenti softvera na hardveru je "<<execution environment>>". Da bi čitalac softverske dokumentacije znao koja komponenta je raspoređena na koji specifični čvor potrebno je da koristi stil instalacije softverske arhitekture.

Stil raspoređivanja je povezan sa drugim komponenta i konektor stilovima koji omogućavaju softverske elemente dodeljene hardveru određene računarske platforme. Takođe, stil raspoređivanja je povezan sa stilom instalacije koji prikazuje sadržaje datoteka koje su raspoređene na hardverske čvorove.



Slika 2.3 Prikaz stila raspoređivanja softverske arhitekture korišćenjem UML jezika [Izvor: Clements [2]]

DEPLOYMENT ARCHITECTURAL PERSPECTIVE (VIDEO)

Trajanje 1:23.

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

Poglavlje 3

Stil instalacije

PREGLED STILA INSTALACIJE

Stil instalacije vrši alociranje komponenata stila komponenta i konektor u datoteke produkcionog okruženja.

Kada se softverski sistem implementira, rezultujuće datoteke moraju biti spakovane tako da mogu biti instalirane na osnovnoj proizvodnoj platformi (kao što je desktop računar ili server mašina koja pokreće aplikacioni server). Datoteke za instalaciju najčešće mogu uključiti biblioteke, izvršne datoteke, log fajlove, opise raspoređivanja, skripte i statički sadržaj (na primer HTML fajlove i slike). Za velike softverske sisteme broj fajlova koji se instalira u produkcionoj fazi može brojati i stotine fajlova. Ovi fajlovi moraju biti organizovani kako bi održali kontrolu i integritet procesa izgradnje i pakovanja sistema. Tehnike za upravljanje konfiguracijom, alati za izgradnju i alati za instalaciju najčešće pomažu da se posao instalacije završi do kraja. Opis arhitekture prikazuje kako je instalirani sistem organizovan u vidu strukture fajlova i foldera i uz opis kako su softverski elementi mapirani u strukturu koja odgovara članovima razvojnog tima.
Stil instalacije pomaže prilikom opisa koji specifični fajlovi treba da budu korišćeni i na koji način treba da budu konfigurisani i zapakovani da omoguće različite verzije sistema. Održavanje više različitih verzija je uobičajena praksa za mnoge sisteme. Različite verzije istog sistema mogu:

- podržavati internacionalizaciju
- ponuditi različite cene (besplatna i komercijalna verzija)
- omogućiti prilagođavanje različitim klijentima
- podržavati klijente u distribuiranom sistemu koji još uvek šalje zahteve u vidu poruka ka starim verzijama sistema

Kada je proces implementacije u toku, alati za upravljanje konfiguracijom i skripte za izgradnju aplikacije pomažu u automatizaciji procesa selektovanja, konfigurisanja i pakovanja određenih konfiguracionih elemenata za različite verzije. Arhitektura takođe opisuje i strukturu fajlova koji bi trebali biti u sklopu stila instalacije.

ELEMENTI, RELACIJE I SVOJSTVA STILA INSTALACIJE

Elementi stila instalacije su elementi softvera i elementi okruženja dok se kao zvanična relacija koristi "dodeljen na".

Tabela prikazuje osnovne karakteristike stila instalacije. Elementi okruženja softvera u pogledu instalacije su datoteke i direktorijumi u sistemu datoteka koji su organizovani u strukturu drveta. Softverski elementi su elementi pogleda komponenta i konektor kao što su procesi, niti, servleti ili skladišta podataka. Dve relacije stila instalacije su:

- **dodeljen na ("allocated to").** Relacija između komponenti i elemenata konfiguracije. Ova relacija povezuje komponentu sa datotekom ili folderom koji čuva tu komponentu u sistemu datoteka.
- **zadržavanje.** Folder u datoteci sadrži druge foldere i/ili datoteke. Takođe, datoteka (kao što je zip datoteka) može sadržati i druge datoteke i foldere.

Određeni fajl ili folder može biti sadržan u više datoteka ili fascikli ili za više instaliranih verzija.

Pregled	Stil instalacije opisuje mapiranje između komponenti unutar softverske arhitekture sa fajl sistemom u produkcionom okruženju.
Elementi	Softverski elementi: komponenta iz komponenta i konektor pogleda. Svojstva softverskog elementa uglavnom uključuje zahteve od strane produkcionog okruženja kao što je recimo zahtev za podržavanje Java programskog jezika ili baze podataka ili pristup određenom delu fajl sistema. Elementi okruženja: konfiguracioni predmet, kao što je fajl ii folder. Svojstva elemenata okruženja uključuju indikatore karakteristika omogućene od produkcionog okruženja.
Relacije	Dodeljeno („alocated to“). Komponenta je dodeljena određenom predmetu konfiguracije. Sadržan u. Jedan predmet konfiguracije se nalazi u drugom.
Ograničenja	Fajlovi i folderi su organizovani u strukturi drveta gde je u upotrebi „je sadržan u“ relacija.

Slika 3.1 Prikaz karakteristika stila instalacije [Izvor: Clements [2]]

KADA SE KORISTI STIL INSTALACIJE

Pogled stila instalacije može biti projektovan da bi koristio različite varijacije jer se često može desiti da zahtevi instalacije podrazumevaju instalaciju na različitim hardverskim platformama.

Kao i kod stila raspoređivanja, važni elementi softvera i elementi okruženja softvera utiču na raspodelu softverskih komponenti. Razumevanje organizacije fajlova i foldera u procesu instalacije softvera može pomoći razvojnom timu da izvrši sledeće zadatke:

- kreiranje procedura za izgradnju i raspoređivanje
- prolazak kroz veliki broj fajlova i foldera koje su deo instaliranog sistema u cilju lociranja specifičnog fajla koji zahteva pregled (primer može biti log fajl ili konfiguracioni fajl)
- selektovanje i konfigurisanje fajlova u paket određene verzije softvera
- ažuriranje i konfigurisanje fajlova višestrukih instaliranih verzija istog sistema
- identifikovanje svrhe ili sadržaja fajlova koji su oštećeni ili nedostaju što može uzrokovati problem u produkciji
- projektovanje i implementacija opcije automatskog ažuriranja

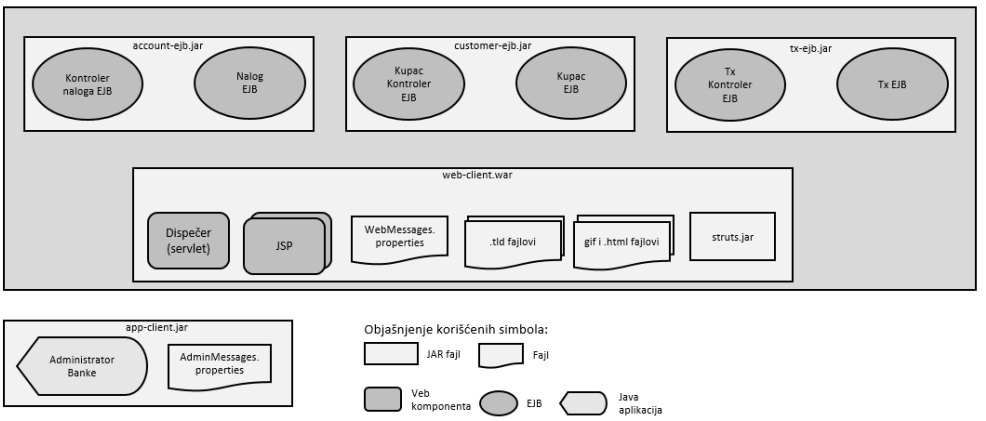
Zahtevana svojstva softverskih elemenata u stilu instalacije softverske arhitekture takođe mogu biti korišćena da podrže analizu kupovine mogućih produkcionih okruženja.

Stil instalacije je najčešće usko povezan sa stilom raspoređivanja i može se smatrati nastavkom prikaza softverske arhitekture nakon stila raspoređivanja.
Organizacija strukture drveta u okviru fajlova i foldera takođe mora biti prikazana. UML jezik nudi veliki broj ugrađenih simbola koji se mogu koristiti u okviru prikaza stila instalacije uključujući i stereotipe "<<artifact>>" za označavanje datoteke (konfigurisanje) i "<<manifest>>" objekta za označavanje zadržavanja. Na slici 2 je prikazan primer stila instalacije korišćenjem neformalne notacije.

POKAZNI PRIMER: STIL INSTALACIJE - NEFORMALNA NOTACIJA

Svaka oznaka unutar prikaza stila instalacije softverske arhitekture mora prikazivati komponente, datoteke, foldere i mapiranje između njih.

Prikaz stila instalacije sistema banke ("DukesBank") predstavlja Java EE aplikaciju koja se obično nalazi u datotekama Java arhive (JAR). Kao zip datoteka, JAR datoteka može sadržati i druge datoteke. U tom slučaju Enterpise Java Bean JAR datoteka sadrži EJB klase i druge datoteke koje EJB klase možda treba da koriste. Datoteke veb arhive (WAR) sadrže veb komponente (servlete i JSP) i vrlo često sadrže i HTML, JPEG i druge datoteke koje se koriste na veb stranicama za "statički sadržaj". Datoteke koje se koriste za arhiviranje (EAR) predstavljaju pakete od nula ili više JAR ili nula ili više WAR datoteka. Sve komponente na serveru se nalaze u DukesBankApp.ear koja se nalazi na serveru. Na dijagramu je prikazano da je aplikacija "Administrator banke" Java aplikacija koja se nalazi na klijentu, raspoređena u "app-client.jar" koji se nalazi na administratorsku korisničku mašinu.

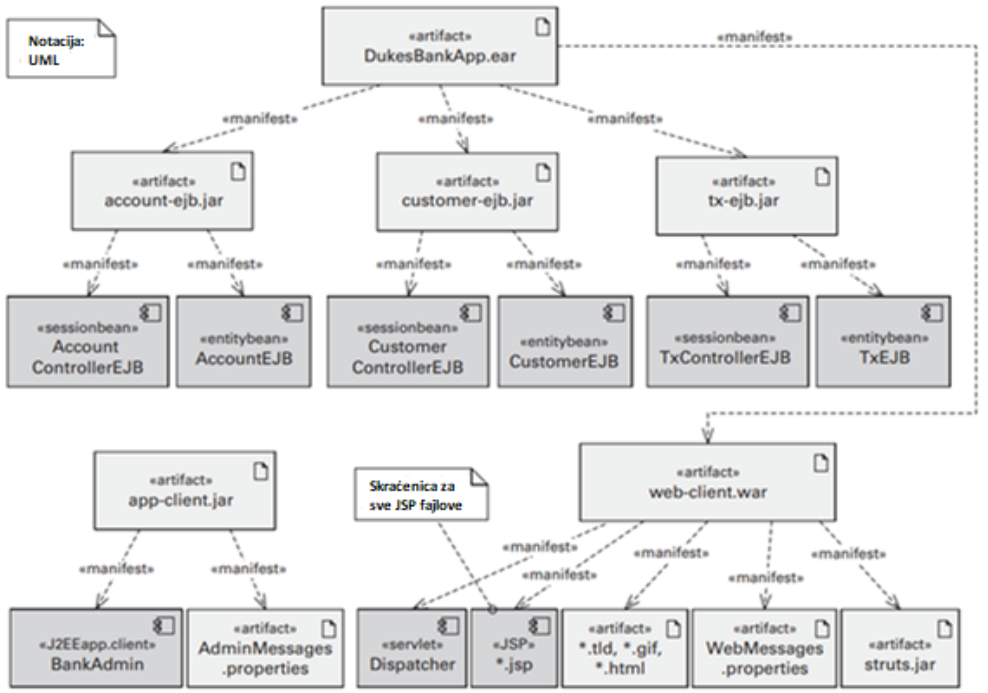


Slika 3.2 Prikaz stila instalacije korišćenjem neformalne notacije [Izvor: Clements [2]]

POKAZNI PRIMER: STIL INSTALACIJE - UML NOTACIJA

Prikaz stila instalacije kroz UML jezik zahteva korišćenje specifičnih grafičkih simbola.

Na slici 3 prikazan je stil instalacije aplikacije DukesBank predstavljen korišćenjem UML jezika. Stereotip "<<artifact>>" označava datoteku bilo koje vrste a "<<manifest>>" stereotip označava da je određena komponenta, klasa ili drugi objekat unutar datog objekta. Predstavljeni su artefakti : DukesBankApp koji se nalazi u .ear fajlu (engl. Enterpise Application archive) koji se koristi u Java EE za pakovanje jednog ili više modula u jednu arhivu kako bi se raspoređivanje na server aplikacije izvršilo istovremeno. U okviru navedenog fajla nalaze se ostale jar datoteke koje su predstavljene na slici (account-ejb.jar, customer-ejb.jar i tx-ejb.jar) i web-client.war korišćenjem <<manifest>>" stereotipa. Opisani web-client.war sadrži svoje fajlove koje koristi pri izvršavanju definisanih funkcionalnosti u okviru aplikacije. Na dijagramu su predstavljeni i moduli web-client.war modula povezani stereotipom "<<manifest>>" u donjem delu slike 3. Prilikom instalacije aplikacije vrši se raspakivanje identifikovanih modula po navedenom redosledu shodno definisanim stereotipima.



Slika 3.3 Prikaz stila instalacije korišćenjem UML jezika [Izvor: Clements [2]]

Poglavlje 4

Stil dodeljivanja radnih zadataka

PREGLED STILA DODELJIVANJA RADNIH ZADATAKA

Stil dodeljivanja radnog zadatka se koristi za podelu sistema na module i dodeljivanje određenih modula timovima ili članovima tima koji je odgovoran za realizaciju sistema.

Stil definiše odgovornost za implementaciju i integraciju modula određenom razvojnom timu. Navedeni stil se uglavnom koristi za povezivanje aktivnosti sa resursima kako bi projektant softvera osiguran da svaki modul bude dodeljen pojedincu ili timu. Arhitektura u kombinaciji sa procesom određuje alokaciju.
Upravljački alat koji može biti korišćen je "Work Breakdown Structure (WBS)". Alat definiše projekat ili grupu projekata na način koji olakšava proces organizovanja i definisanja ukupnog obima rada na projektu.
Radni zadaci predstavljaju mapiranje softverske arhitekture na grupe ljudi kroz stil dodeljivanja radnog zadatka. Timovi a samim tim i radni zadaci nisu povezani sa pisanjem koda koji će se pokrenuti u finalnom sistemu. Postoji mnogo više zadataka koji ljudi moraju da obavljaju: upravljanje konfiguracijom, testiranje, evaluaciju potencijalnih komercijalnih proizvoda ili kontinuirano održavanje proizvoda.

Čak i u slučaju da se modul softvera kupi u celini kao komercijalni proizvod bez potrebe za bilo kakvim radom na implementaciji i dalje je potrebno da neko od članova tima vrši testiranje, razumevanje načina funkcionisanja modula i omogući integraciju sa postojećim sistemom. Tim koji je odgovoran za navedene stavke ima definisan radni zadatak.

Stil dodeljivanja radnog zadatka vezan je za stil dekompozicije jer se na taj način najlakše vrši mapiranje alokacija. Prikaz stila dodeljivanja radnog zadatka može proširiti dekompoziciju modula na module koji odgovaraju razvojnim alatima, alatima za testiranje, sistemima za upravljanje konfiguracijama i slično čija nabavka ili svakodnevno korišćenje takođe može biti dodeljeno određenom članu ili timu. Primena stila dodeljivanja radnog zadatka omogućava projektantu da razmisli na koji način je moguće podeliti posao u delove kojima može upravljati.

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

ELEMENTI, RELACIJE I SVOJSTVA STILA DODELJIVANJA RADNIH ZADATAKA

Elementi ovog stila su softverski moduli i grupe ljudi u razvojnoj organizaciji.

U ovom stilu relacija "dodeljen ka" ide od softverskih elemenata ka organizacionim jedinicama.
Na slici 1 prikazane su karakteristike stila dodeljivanja radnog zadatka.

Pregled	Stil dodeljivanja posla opisuje mapiranje arhitekture softvera članovima tima ili timovima u okviru organizacije
Elementi	Softverski elementi: modul. Svojstva uključuju zahtevani skup veština i potreban raspoloživi kapacitet (napor, vreme). Elementi okruženja: Organizaciona jedinica, osoba, tim ili odeljenje. Svojstva zahtevaju veštine i kapacitete u smislu rada i raspoloživog vremena
Relacije	Dodeljeno („alocated to“) softverski element se dodeljuje ogranizacionoj jedinici
Ograničenja	Alokacija je neograničena ali se u praksi koristi ograničenje da je jedan modul dodeljen jednoj ogranizacionoj jedinici.

Slika 4.1 Prikaz karakteristika stila dodeljivanja radnog zadatka [Izvor: Clements [2]]

Stil dodeljivanja radnog zadatka pokazuje glavne elemente softvera koji moraju biti prisutni kako bi se formirao radni sistem kao i okruženja u kojima se softver razvija. Stil dodeljivanja radnog zadatka pomaže u planiranju i upravljanju resursima timova, u dodeljivanju odgovornosti za izgradnju i za objašnjavanje strukture projekta.

POKAZNI PRIMER: STIL DODELJIVANJA RADNIH ZADATAKA - NEFORMALNA NOTACIJA

Ne postoje specijalne notacije za stil dodeljvainja radnih zadataka, uglavnom je na projektantu softvera da odluči na koji način će prikazati stil dodeljivanja radnih zadataka.

Kroz neformalnu notaciju, tabela prikazuje softverske elemente i odgovorne timove. Tabelarni prikaz je vrlo jednostavan i služi kao opis stila dodeljivanja radnih zadataka. Projektant softvera ne mora da bira tim već može da prosledi informaciju ka menadžmentu.

Slika 2 predstavlja primarni prikaz NASA sistema koji je podeljen na delove i podsisteme a zatim su podsistemi dodeljeni određenim delovima razvojnog tima. Moduli najvišeg nivoa nazivaju se segmentima a zatim se oni razgrađuju u jedinice podsistema.

ECS elementi (moduli)		
Segment za obradu naučnih podataka	Podsistem	Organizaciona jedinica
	Klijent	Naučni tim
	Interoperabilnost	Glavni kontaktor tim
	Menadžment podataka	Tim za kontrolu podataka
	Obrada podataka	Tim za kontrolu podataka
	Server podataka	Tim za kontrolu podataka
	Planiranje	Orbitalni tim vozila
Segment operacije leta	Planiranje i raspoređivanje	Orbitalni tim vozila
	Menadžment podataka	Tim za kontrolu baze podataka
	Korisnički interfejs	Tim za kontrolu korisničkog interfejsa

Slika 4.2 Tabelarni prikaz stila dodeljivanja radnih zadataka kroz neformalnu notaciju [Izvor: Clements [2]]

Poglavlje 5

Ostali stilovi alokacije

UPOTREBA DRUGIH STILOVA ALOKACIJE

Pored tri navedena stila alokacije, postoje i dodatni stilovi alokacije koji na drugi način predstavljaju softversku arhitekturu.

Pored tri opisana stila, postoje i:

1. **Stil implementacije:** opisuje kako je organizovano razvojno okruženje kroz strukturu drveta primenjenu na fajlovima i folderima i na koji način se moduli iz pogleda modula uklapaju u tu strukturu. Primenom stila implementacije pogled koji proizilazi pokazuje na koji način je potrebno organizovati foldere i datoteke kako bi ih jedinice za implementaciju pronašle. Jedinice su: klase, programi, skripte, predmeti testiranja, datoteke ili bilo koji objekti u okviru sistema koji se razvija. Kroz prikaz implementacije omogućeno je da programeri jednostavnije pronalaze činjenice o razvojnim objektima i da ih postavljaju na odgovarajuće mesto. Stil implementacije je sličan stilu instalacije ali umesto prikazivanja fajlova i foldera u produkcionom okruženju , prikazuje organizaciju fajlova i foldera u razvojnom okruženju.
2. **Stil skladištenja podataka:** Stil skladištenja podataka opisuje mapiranje između softverskih entiteta podataka i hardvera na serverima podataka sa kojima softver komunicira. Kada se izvrši primena stila skladištenja podataka na sistem, rezultat je prikaz podataka u tabelama i način na koji se ti podaci distribuiraju na server. Takođe, pogled može prikazivati na koji server odlaze podaci uz detaljne specifikacije eventualne replikacije baze podataka na serverima. Na ovaj način omogućeno je praćenje i dostupnost podataka i sprečavanje mogućih napada koji mogu uticati na performanse sistema

SPECIJALIZACIJA STILOVA ALOKACIJE

Specijalizacija stila alokacije se vrši u slučaju potrebe za ponovnim korišćenjem stila u različitim delovima određenog sistema.

Drugi stilovi alokacije su takođe mogući, moguće je definisati zahteve za stilove alokacije koji povezuju sistemske zahteve sa softverskim elementima arhitekture. Primer za tako nešto može biti i specijalizacija određenog stila alokacije:

- **Specijalizacija stila raspoređivanja:** Microsoft je razvio šablon nazvan "Tiered Distribution" koji propisuje raspodelu softverskih komponenti u višeslojnoj arhitekturi na hardverske elemente koji se koriste za pokretanje. Opisani šablon omogućava generički

stil primene. Takođe, kompanija IBM ima svoju verzije šablona kao što je: topologija jedne mašine (stand- alone server), topologija vertikalnog skaliranja (vertical scaling topology), topologija horizontalnog skaliranja (horizontal scaling topology) itd.

- **Specijalizacija stila dodeljivanja radnih zadataka:** Moguće je dokumentovati korišćene šeme dodele radnih zadataka u timu kroz specijalizaciju stila dodele radnih zadataka. Specijalizacija stila može biti izvršena kroz:
  -Stil platforme. Postavljanje procesa razvoja softvera u obliku proizvodne linije gde određeni deo razvojnog tima proizvodi objekte koji se mogu ponovo koristiti a drugi deo razvojnog tima ponovo koristi tako razvijene objekte u procesu razvoja softvera.
  -Stil kompentencije. Radni zadatak se dodeljuje određenim delovima tima u zavisnosti od tehničkog znanja. Na primer, projektovanje korisničkog interfejsa je izvršeno u onom delu tima gde se nalaze stručnjaci za projektovanje korisničkog interfejsa.
  -Stil otvorenog koda. Većina nezavisnih saradnika doprinosi razvijanju softverskog proizvoda u skladu sa strategijom za tehničku integraciju. Centralizovana kontrola je minimalna osim u slučaju kada nezavisni saradnik integriše svoj programski kod u softverski proizvod.

Poglavlje 6

Kombinovanje različitih pogleda

KOMBINOVANJE POGLEDA NA SOFTVERSKU ARHITEKTURU

Osnovni principi dokumentovanja softverske arhitekture predstavljaju grupu različitih pogleda koji omogućavaju detaljan prikaz softverske arhitekture.

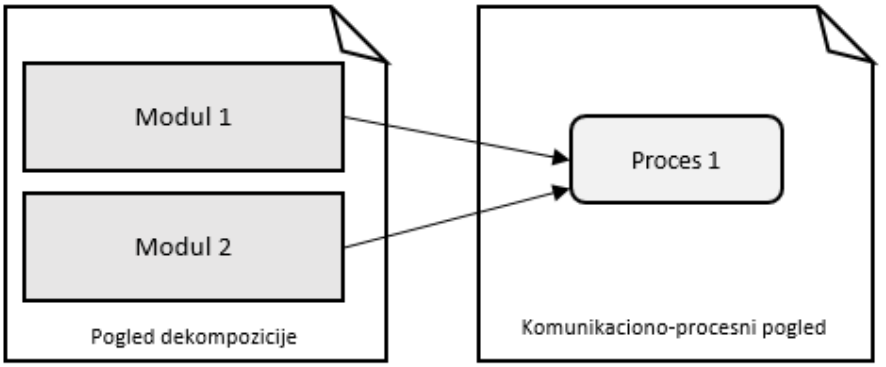
Često se dešava da tako odabrani pogledi nemaju zajedničke elemente ili relacije sa drugim pogledima pa tako čitaoci projektne dokumentacije nemaju uvid u ono što je projektant kroz poglede hteo da prikaže.
Kako su pogledi na softversku arhitekturu delovi iste softverske arhitekture i postoje kako bi čitaocu detaljno prikazali softver većina pogleda ima međusobnu povezanost sa drugim pogledima.

TIPOVI ASOCIJACIJA IZMEĐU RAZLIČITIH POGLEDA

Povezivanje različitih pogleda na softversku arhitekturu moguće je korišćenjem asocijacija.

Pogledi su povezani jedni sa drugima na različite načine i korišćenjem različitih veza:

- **više na jedan asocijacija:** Prikaz više elemenata u jednom pogledu je povezan sa jednim elementom u drugom pogledu. Implementacione jedinice su najčešće povezane sa izvršnim komponentama koje postaju. Asocijacija treba da detaljno prikaže koji moduli su mapirani sa kojom komponentom. (slika 1)
- **jedan na više asocijacija:** Prikaz jednog elementa koji je povezan sa jedinom pogledom na više elemenata drugog pogleda. Primer može biti: "Modul korisnička korpa za kupovinu ("shopping cart") je povezana sa više komponenti aplikacije za veb kupovinu." Takav primer dat je na slici 2.
- **više na više asocijacija:** Prikazuje asocijaciju između grupe elemenata jednog pogleda sa grupom elemenata drugog pogleda. Ovaj tip asocijacije reflektuje kompleksnost na dva pogleda, u cilju prikaza glavnih aspekata odabranih pogleda.



Slika 6.1 Više na jedan asocijacija različih pogleda na softversku arhitekturu [Izvor: Clements [2]]

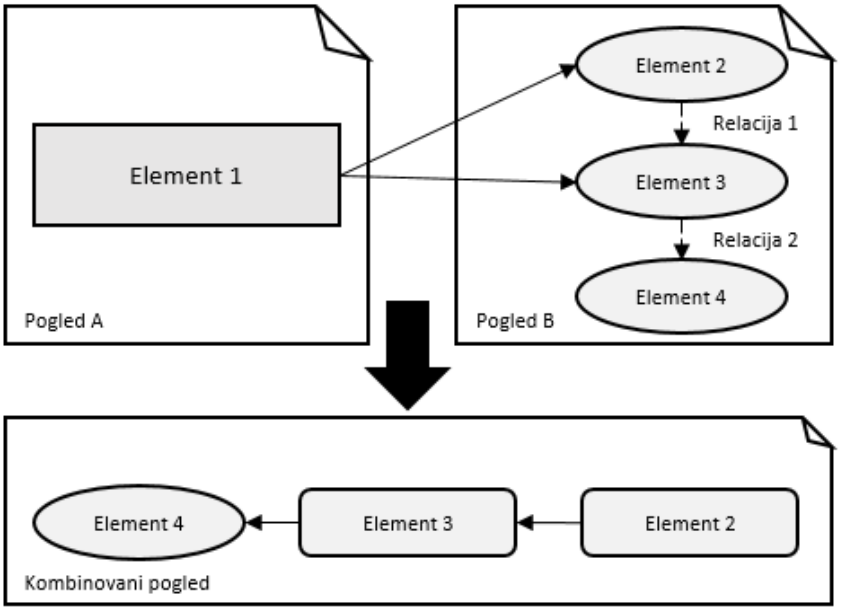
Slika 6.2 Jedan na više asocijacija različih pogleda na softversku arhitekturu [Izvor: Clements [2]]

KOMBINOVANI POGLEDI

Najjednostavniji način za prikaz asocijacije između dva pogleda je da se pogledi grupišu u jedan kombinovani pogled.

Kombinovani pogled najčešće smanjuje broj pogleda na softversku arhitekturu u dokumentu jer zamenjuje više različitih pogleda.

Na slici 3 prikazano je kako se vrši dokumentovanje mapiranja kroz kombinovane poglede.



Slika 6.3 Smeštanje više različitih elemenata jednog pogleda u jedan element drugog pogleda [Izvor: Clements [2]]

Postoje dva načina za kreiranje kombinovanog pogleda:
- **razvoj dodatnog sloja (overlay)** koji kombinuje informacije iz dva različita pogleda na softversku arhitekturu. Ovakav pristup dobro funkcioniše ako je asocijacija između dva pogleda jaka u smislu da postoji između dva softverska elemenata dva različita pogleda.
- **kreiranjem hibridnog stila**, koji kombinuje dva postojeća stila u svrhu kreiranja vodiča za stil koji ukazuje na kombinovane stilove i opisuje sve nove dobijene elemente i tipove odnosa, osobine i ograničenja. Hibridni stilovi zahtevaju definisanje novih elemenata i novih tipova odnosa shodno kombinovanju dva različita stila. Hibridni stil je korisno razviti ako se stil koristi po nekoliko puta u istom sistemu ili na istim vrstama sistema razvijenim u jednoj organizaciji i ako postoji mnogo poznatih korisnika.
Dešava se da ponekad kombinovani pogled omogućava samo kratkoročnu upotrebu u procesu analize ili komunikacije.
U tom slučaju, projektant softvera, ima tri načina za uspostavljanje asocijacije između samostalnih pogleda na softversku arhitekturu:
- dokumentovanjem mapiranja između zasebnih pogleda na softversku arhitekturu
- kreiranjem hibridnog stila i opisivanjem pogleda na softversku arhitekturu upotrebom tog stila
- kreiranjem preklapanja dva različita pogleda

KADA KOMBINOVATI POGLEDE NA SOFTVERSKU ARHITEKTURU

Prilikom razmatranja kombinovanog pogleda, potrebno je proveriti da li je asocijacija između elemenata jasna.

U suprotnom, pogledi verovatno nisu dobri za kombinovanje jer će krajnji rezultat biti kompleksan i zbunjujući pogled. U tom slučaju bi bilo bolje upravljati asocijacijom pojedinačno kroz poglede.

Za različite članove razvojnog tima potrebne su različite vrste informacija. Izbor pogleda na softversku arhitekturu direktno zavisi od potreba članova tima koji se bavi razvojem softvera. Često se dešava da kombinovanje pogleda obuhvata sledeće:

- različite komponenta i konektor poglede. Iz razloga što komponenta i konektori pogledi prikazuju relacije u toku izvršavanja softvera između komponenti potrebno je izvršiti detaljnu specifikaciju načina rada sistema. Različiti komponenta i konektor pogledi teže da prikažu različite delove sistema ili da prikažu razlaganje komponenti u drugačijim pogledima. Rezultat je često skup pogleda koji mogu biti jednostavno kombinovani.

- pogled raspoređivanja sa servisno-orijentisanim pogledom ili komunikaciono-procesnim pogledima. Servisno-orijentisani pogled opisuje servise a komunikaciono-procesni pogled prikazuje procese. U oba slučaja, tu su komponente koje su alocirane na procesore. Iz tog razloga postoji jaka veza između elemenata tih pogleda.
- pogled raspoređivanja i pogled instalacije. Kombinovanje pogleda prikazuje instalacione fajlove i hardverske elemente na koje se postavljaju instalacioni fajlovi
- pogled razlaganja i dodeljivanje radnih zadataka, implementacije. Razloženi moduli sa jedinicama na kojim je moguće raditi ili ih koristiti u druge svrhe.

- generalizacija i aspekti. Pogledi koji se oslanjaju na klase i objekte i relacije između njih, to su dva pogleda sa jakom vezom asocijacije.

POKAZNI PRIMER: KOMBINOVANJE POGLEDA NA SOFTVERSKU ARHITEKTURU

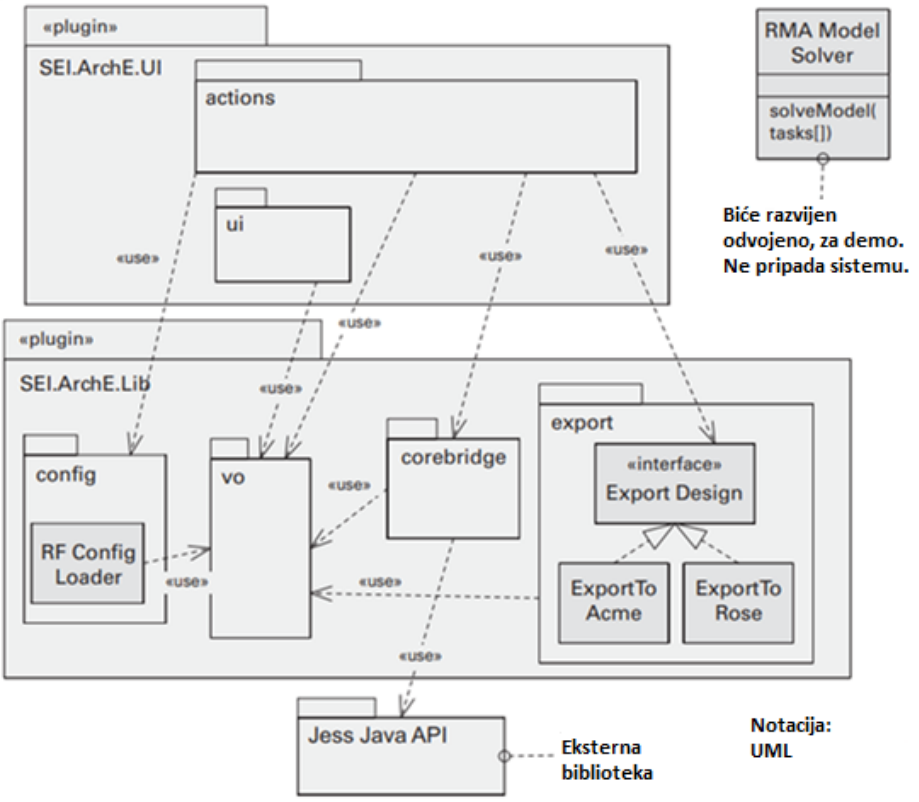
Primer kombinovanja pogleda na softversku arhitekturu ArchE sistema.

Na slici 1 prikazan je osnovni pogled na softversku arhitekturu ArchE alata (SEI Architecture Expert). Ovaj alat omogućava projektantu softvera da kreira arhitekturu sistema na osnovu tri tipa ulaznih podataka:

- atributi kvaliteta
- funkcionalnosti sistema koji se projektuje
- postojeći projektovani delovi

Ovaj alat konstruiše prikaz odgovornosti sistema i određenih uočenih zavisnosti između njih. Pokreće se pomoću dodataka koji omogućavaju modele atributa kvaliteta i koji mogu biti korišćeni za analizu performansi ili mogućih modifikacija. Na osnovu ulaznih podataka, rezultati analiza mogu odrediti projektovanje softverske arhitekture.

Slika 4 predstavlja razlaganje sistema na module, upotrebu i generalizaciju. Stereotip <<plugin>> označava da su sadržani moduli upakovani kao Eclipse dodaci (plugins).



Slika 6.4 Dekompozicija-koristi-generalizaciju kombinovani pogled na ArchE sistem [Izvor: Clements [2]]

Poglavlje 7

4+1 Kručtenovi pogledi

PREGLED 4+1 KRUČTENOVIH POGLEDA

Softverska arhitektura se najčešće koristi u procesu projektovanja i implementacije kao detaljan prikaz strukture softvera.

Elementi softverske arhitekture su definisani kako bi rešili osnovne funkcionalnosti i korisničke zahteve. Takođe, elementi softverske arhitekture treba da omoguće i ispunjavanje zahteva za performansama i ostale nefunkcionalne zahteve ko što su skalabilnost, portabilnost i dostupnost. Upotrebom Kručtenovih pogleda softversku arhitekturu je potrebno predstaviti kroz:

- Logički pogled, koji predstavlja projektovanje objektnog modela (u slučaju korišćenja objektno-orijentisane metode projektovanja)
- Procesni pogled, koji beleži procese, način komunikacije i ponašanja sistema tokom izvršavanja određene funkcionalnosti
- Fizički pogled, koji opisuje komponente softverske arhitekture iz ugla projektanta softvera
- Pogled raspoređivanja, koji opisuje povezivanje softverskih komponenti sa hardverskim elementima softverskog okruženja

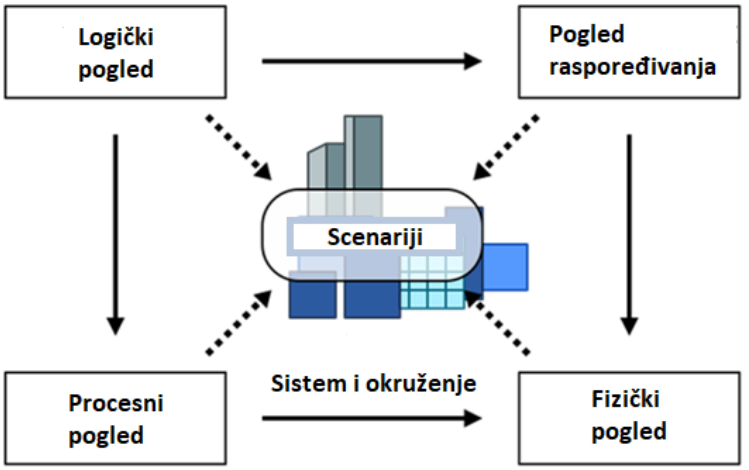
Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

PRIMENA 4+1 KRUČTENOVIH POGLEDA

Primenom Kručtenovih pogleda omogućeno je sagledavanje arhitekture softvera iz logičkog, procesnog, fizičkog i pogleda raspoređivanja softverskih komponenti.

Opisani pogledi na softversku arhitekturu zahtevaju modelovanje sledećih dijagrama:
• Logički pogled: modelovanje klasnog dijagrama i dijagrama stanja
• Procesni pogled: dijagram aktivnosti
• Fizički pogled: dijagram komponenti
• Pogled raspoređivanja: dijagram komponenti

Navedene dijagrame moguće je modelovati upotrebom alata koji podržava UML jezik i predviđenu grafičku notaciju. Pored modelovanja potrebno je detaljno dokumentovati dijagrame i opisati način predstavljanja softverske arhitekture kroz odabrani pogled.



Slika 7.1 Prikaz pogleda na softversku arhitekturu iz 4+1 Kručtenovih pogleda [Izvor: P.B. Kruchten [3]]

4+1 VIEW INTO SOFTWARE ARCHITETURE (VIDEO)

Trajanje: 7:30 minuta.

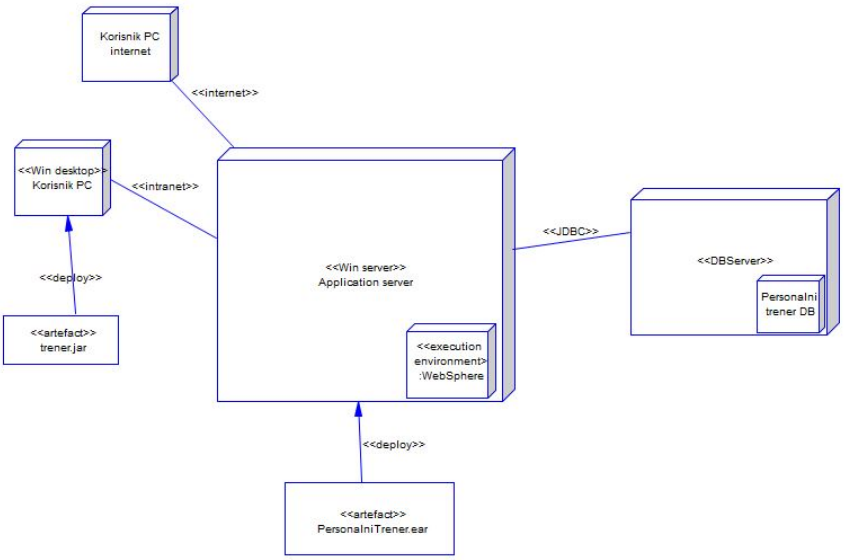
Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Poglavlje 8

Pokazna vežba - stilovi alokacije

STIL RASPOREĐIVANJA - UML FORMALNA NOTACIJA - PRIMER 1

Stil raspoređivanja omogućava prikaz povezanosti elemenata softverskog okruženja i elemenata softverske arhitekture.



Slika 8.1 Primer upotrebe stila raspoređivanja na softversku arhitekturu aplikacije "Personalni trener" [Izvor: Nebojša Gavrilović]

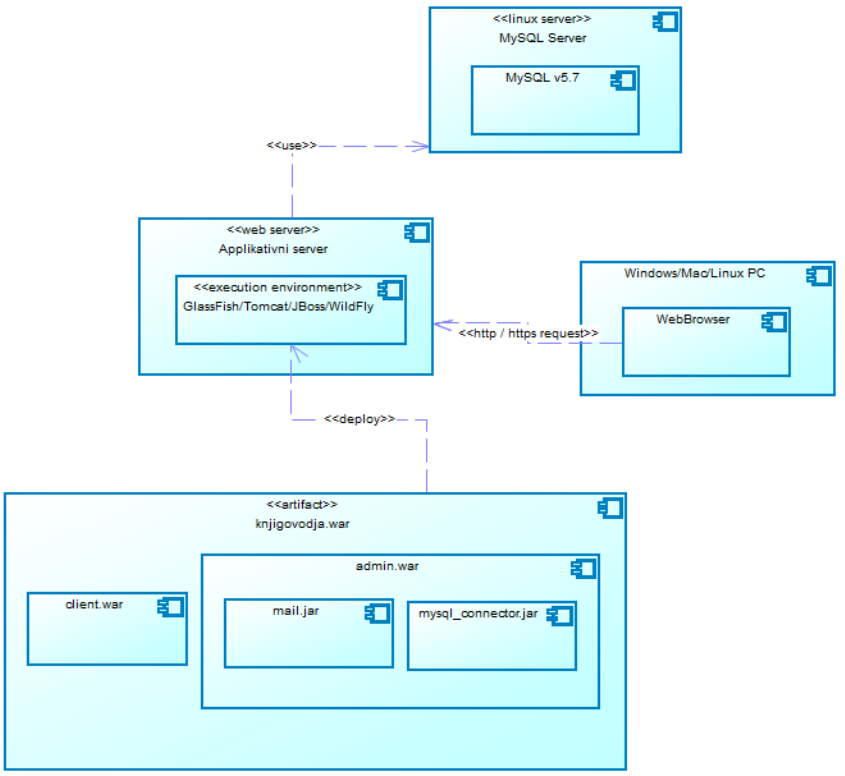
Na slici 1 prikazan je stil raspoređivanja softverske arhitekture stila raspoređivanja na aplikaciji "Personalni Trener"). Identifikovan je veb server na kome se nalazi okruženje "WebSphere" koje se koristi za pokretanje aplikacije na serveru, server baze podataka na kome se nalazi baza podataka "Personalni trener", komponenta koja predstavlja korisnički računar i komponenta koja predstavlja računar preko koga putem interneta korisnik komunicira sa aplikacijom. Prikazan je i objekat "PersonalniTrener.ear" koji se koristi za izvršavanje aplikacije na serveru kao i "trener.jar" koji se koristi na klijentskom računaru. (25min)

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

STIL RASPOREĐIVANJA - UML FORMALNA NOTACIJA - PRIMER 2

Dat je primer upotrebe stila raspoređivanja na primeru softvera za vođenje knjigovodstva.

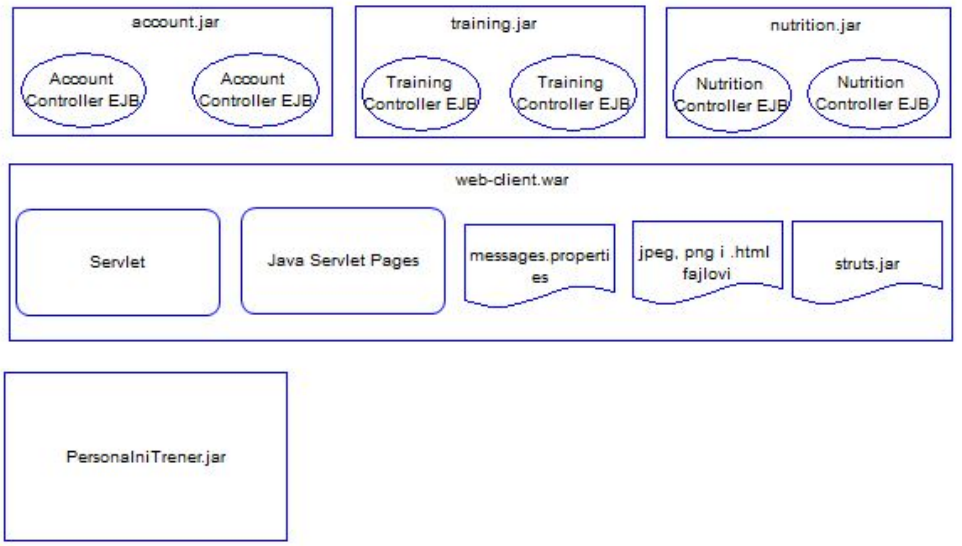
Na slici 2 dat je primer stila raspoređivanja. Predstavljen je aplikativni server na kome se pokreću GlassFish ili Tomcat ili Jboss ili WildFly zavisno od potreba aplikacije. Server na kome se nalazi baza podataka ("My SQL server") povezan je sa aplikativnim serverom dok klijent preko veb pretraživača pristupa aplikativnom serveru. Sa veb serverom komunicira i artefakt "Knjigovođa.war" na kome se nalaze "client.war" i "admin.war" (koji sadrži "mail.jar" i "mysql_connector.jar").(20min)



Slika 8.2 Upotreba stila raspoređivanja na softveru za vođenje knjigovodstva [Izvor: Nebojša Gavrilović]

STIL INSTALACIJE - NEFORMALNA NOTACIJA

Stil instalacije prikazuje komponente softverske arhitekture koje je potrebno alocirati na određene hardverske komponente platforme na kojoj se sistem izvršava.



Slika 8.3 Primer upotrebe stila instalacije na softversku arhitekturu aplikacije "Personalni trener" [Izvor: Nebojša Gavrilović]

Na slici 3 dat je primer primene stila instalacije na softversku arhitekturu aplikacije "Personalni trener". Prikazan je sadržaj identifikovanih komponenti (account.jar, training.jar, nutrition.jar i web-client.war) Pored opisanih komponenti tu se nalazi i komponenta "PersonalniTrener.jar" koja se koristi za pristup aplikaciji od strane korisnika.(20 min)

STIL DODELJIVANJA RADNIH ZADATAKA - NEFORMALNA NOTACIJA

Dat je primer stila dodeljivanja radnih zadataka po modulima (segmentima) i podmodulima (podsistemima) aplikacije "Personalni trener".

Na slici 4 prikazan je raspored dodeljenih zadataka članovima razvojnog tima (koji su podeljeni na razvojni tim, tim za dizajn interfejsa, tim za testiranje, tim za održavanje i tim za analizu podataka). Svaki podsistem (podmodul) ima po dva tima koji su angažovani na procesu razvoja ili održavanja određenog modula.(20min)

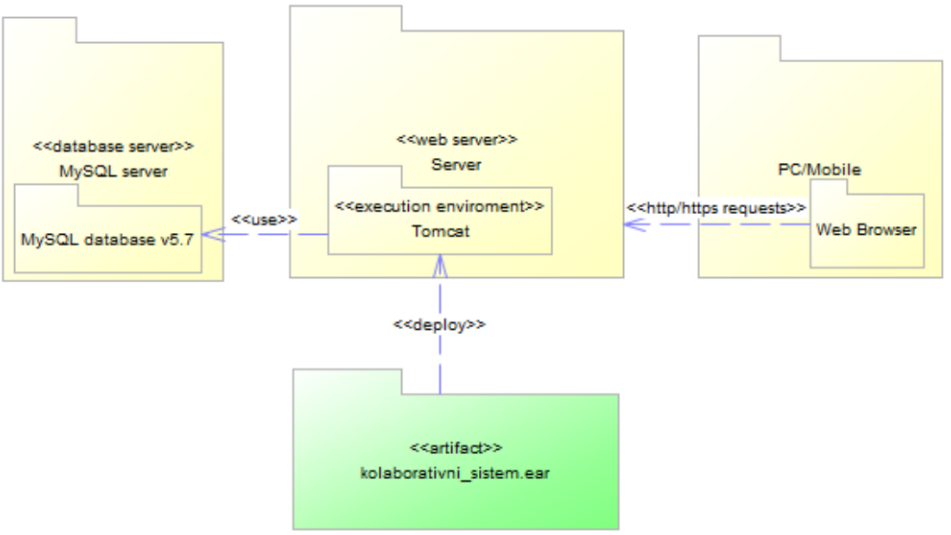
ECS elementi (moduli)		
Generisanje treninga	Podsistem	Organizaciona jedinica
	Trening	Razvojni tim, tim za dizajn interfejsa
	Ishrana	Razvojni tim, tim za testiranje
	Raspored	Tim za održavanje, tim za testiranje
Kreiranje treninga	Obrada podataka	Razvojni tim, tim za analizu podataka
	Planiranje	Razvojni tim, tim za testiranje
	Korisnički interfejs	Razvojni tim, tim za održavanje

Slika 8.4 Primer upotrebe stila dodeljivanja radnih zadataka na softversku arhitekturu aplikacije "Personalni trener" [Izvor: Nebojša Gavrilović]

STIL RASPOREĐIVANJA - SISTEM ZA MENTORISANJE

Ovim stilom je predstavljena povežanost elemenata softverskog okruženja i elemenata softverske arhitekture.

Sistem će biti postavljen na web serveru. Komunicira sa serverom baze podataka. Pristup sistemu imaju web browser-i preko HTTP (ili HTTPS) poziva.(20 min)

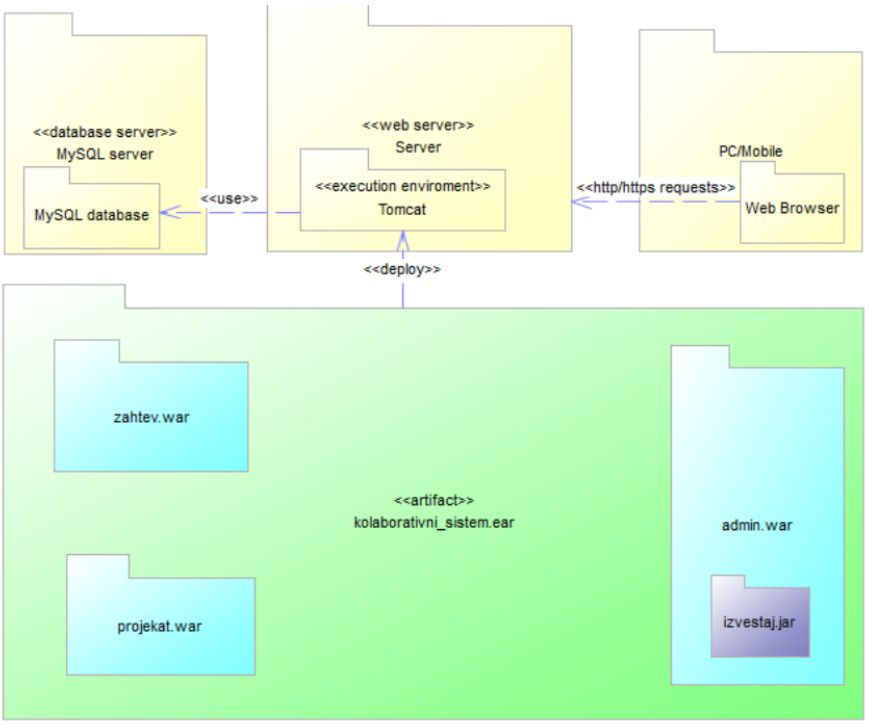


Slika 8.5 Stil raspoređivanja softverske arhitekture sistema za mentorisanje [Izvor: Nebojša Gavrilović]

STIL INSTALACIJE - SISTEM ZA MENTORISANJE

Slično stilu raspoređivanja, ovaj dijagram predstavlja povezanost elemenata ali i podsisteme glavne aplikacije.

Sistem je „upakovan“ u Enterprise archive formatu. To znači da je sistem sastavljen od nekoliko manjih podsistema. Ti podsistemi su zahtev, projekat i admin. Oni su u sistem upakovani kao Web archive, što predstavlja web aplikaciju. U sklopu svakog podsistema, nalaze se odgovarajući kontroleri i servisi koji ostvaruju taj modul. U admin podsistemu se nalazi i biblioteka izvestaj.jar koja služi za lakše generisanje izveštaja..(20 min)



Slika 8.6 Stil instalacije softverske arhitekture sistema za mentorisanje [Izvor: Nebojša Gavrilović]

Poglavlje 9

Individualna vežba - Zadaci

ZADATAK ZA SAMOSTALAN RAD

Na osnovu definisanih korisničkih zahteva primeniti različite stilove alokacije.

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 raspoređivanja (25 min)
- Prikazati softversku arhitekturu sistema kroz stil instalacije (25 min)
- Prikazati softversku arhitekturu sistema kroz stil dodeljivanja radnih zadataka (10 min)

Uporediti modelovane softverske arhitekture i dokumentaciju za svaki modelovani stil. Utvrdite i objasnite stil arhitekture koji najviše odgovara opisanom scenariju. (30 min)

ZADACI ZA SAMOSTALNI RAD - STIL RASPOREĐIVANJA

Potrebno je primeniti stil raspoređivanja na softversku arhitekturu proizvoljnog sistema.

Zadatak 1: Prikazati primenu stila raspoređivanja softverske arhitekture na proizvoljnom sistemu koristeći neformalnu notaciju. Prilikom modelovanja dijagrama pridržavati se primera neformalne notacije stila raspoređivanja. (20min)

Zadatak 2: Prikazati primenu stila raspoređivanja softverske arhitekture na proizvoljnom sistemu koristeći UML notaciju.(10min)
Zadatak 3: Uporediti tako predstavljene softverske arhitekture i opisati uočene razlike shodno korišćenju različitih tipova notacija.(10min)

ZADACI ZA SAMOSTALNI RAD - STIL INSTALACIJE

Potrebno je primeniti stil instalacije na softversku arhitekturu proizvoljnog sistema.

Zadatak 1: Prikazati primenu stila instalacije softverske arhitekture na proizvoljnom sistemu. Prilikom modelovanja dijagrama pridržavati se pravilne notacije stila instalacije. (20min)

Zadatak 2: Prikazati primenu stila instalacije softverske arhitekture na proizvoljnom sistemu koristeći UML notaciju.(10min)
Zadatak 3: Uporediti tako predstavljene softverske arhitekture i opisati uočene razlike shodno korišćenju različitih tipova notacija.(10min)

ZADACI ZA SAMOSTALNI RAD - STIL DODELJIVANJA RADNIH ZADATAKA

Potrebno je primeniti stil dodeljivanja radnih zadataka na softversku arhitekturu proizvoljnog sistema.

Zadatak 1: Prikazati primenu stila dodeljivanja posla na softversku arhitekturu na proizvoljnom sistemu. Prilikom kreiranja tabelarnog prikaza voditi se prethodnim primerom. (30min)

Zadatak 2: Objasniti kreirane tabele.(10min)

ZADACI ZA SAMOSTALNI RAD - KOMBINOVANJE RAZLIČITIH POGLEDA

Potrebno je prikazati kombinovanje različitih pogleda na softversku arhitekturu proizvoljnog sistema.

Zadatak 1: Prikazati primenu kombinovanog pogleda na softversku arhitekturu proizvoljnog sistema. (20min)

Zadatak 2: Dokumentovati kombinovani pogled na softversku arhitekturu.(10min)

ZADACI ZA SAMOSTALNI RAD-4+1 KRUČTENOVI POGLEDI

Potrebno je primeniti Kručtenove poglede na softversku arhitekturu proizvoljnog sistema.

Zadatak 1: Prikazati primenu 4+1 Kruchten pogleda na softversku arhitekturu proizvoljnog sistema. (10min)

Zadatak 2: Opisati uočene prednosti tako predstavljene softverske arhitekture sistema. (10min)

ZADACI ZA DISKUSIJU NA VEŽBI

Otvorite diskusiju po svakom od dole postavljenih pitanja.

1. Da li elementi stila alokacije utiču na okruženje? Objasniti.(10min)

2. Kako ograničenja određenog stila utiču na njegovu primenu na softversku arhitekturu? (10min)

3. Da li ograničenja mogu da budu fleksibilna u nekim situacijama kada je potrebno kombinovati različite stilove softverske arhitekture?(10min)

4. Da li je moguće kombinovati stil dodeljivanja radnih zadataka sa nekim drugim stilom softverske arhitekture?(10min)

5. Kada na razvoju softverske arhitekture učestvuje više članova tima ili više timova, ko donosi odluku o primeni određenog stila softverske arhitekture?(10min)

Poglavlje 10

Domaći zadatak br.5

PRAVILA ZA IZRADU DOMAĆEG ZADATKA

Student zadatake treba samostalno da uradi i da dostavi asistentu.

Domaći zadatak br.5: Koristiti stil dodeljivanja radnih zadataka (podeliti sistem na module). Module i radne zadatke predstaviti odgovarajućom tabelom.
Pri radu, koristite Power Designer.

Domaći zadaci su dobijaju se po definisanim instrukcijama. Domaći zadatak se imenuje: SE311-DZ05-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-DZ05. 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

Stilovi alokacije softverske arhitekture mogu se koristiti kada je potrebno predstaviti komunikaciju elemenata softvera sa okruženjem u kome se softver nalazi.

Primenom različitih stilova alokacije postiže se sledeće:

- Stil raspoređivanja mapiraju elemente softvera na hardver računarske platforme na kojoj se izvršava
- Stil instalacije opisuje mapiranje softverskih elemenata na strukturu datoteka i foldera u produkcionom okruženju
- Stil dodeljivanja radnih zadataka opisuje mapiranje modula na osobe zadužene za razvoj tog modula

Moguće je koristiti i druge stilove alokacije koji su usko specijalizovani za određene softverske arhitekture i omogućavaju ponovnu upotrebu u različitim delovima sistema.

Kombinovanje pogleda na softversku arhitekturu zahteva primenu asocijacija od jednog elementa određenog pogleda ka drugom elementu drugog pogleda.

Kručtenovi pogledi (4+1) omogućavaju prikaz softverske arhitekture kroz različite poglede obuhvatajući kompletan pregled sistema. Primenom logičkog, procesnog i fizičkog pogleda i pogleda raspoređivanja moguće je prikazati sistem različitim tipovima čitalaca softverske dokumentacije.

LITERATURA

U ovoj lekciji korišćena je kao obavezna literatura, referenca 2, poglavlje 5 i poglavlje 6 (2. izdanje) i referenca 3.

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 5 (strane od 189 do 214), poglavlje 6 (strane od 250 do 257)
3. P.B. Kruchten, “The 4+1 View Model of Architecture,” IEEE Software, vol. 12, no. 6, 1995, pp. 42–55

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. E. Gamma et al., Design Patterns:Elements of Reusable Object-Oriented Software, 1st ed., Addison-Wesley Professional, 1994.
5. J. Nielsen, Usability Engineering, Morgan Kaufmann, 1993.
6. Service-oriened Architecture, Concept, Technology, and Design, autora T. Erl u izdanju Pretince Hall, 2005, ISBN 0-13-185858-0.
7. P. Stevens, Using UML – Software Engineering with Objects and Components, Second Edition, Assison-Wesley, Pearson Eduction, 2006
8. 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/