**SE311 PROJEKTOVANJE I ARHITEKTURA SOFTVERA**

**ISPITNA PITANJA**

**Pripremljeno 01.10.2020**

Ovde se daje lista ispitnih pitanja koja treba da ukažu na razumevanje studenta izloženih programskih sadržaja na predavanjima, tj. stečena teorijska znanja koja su podloga za uspešno rešavanje zadataka koji se daju u drugom delu ispita. Pitanja pokrivaju ceo program predmeta, te ne ukazuju studentu koji deo predmeta je prioritetan za ispit. Ispit treba da pokaže u kojoj meri je student stekao znanja celog programa predmeta, te i pitanja, zbog toga, moraju da pokriju ceo program predmeta. Spisak sadrži pitanja po lekcijama, te za svaku od 15 lekcija, dat je spisak pitanja. Zbog ograničenog vremena (45 minuta) student dobija po jedno pitanje iz 10 slučajno odabranih lekcija, te se izbor menja od ispita do ispita.

Odgovori na 10 pitanja obezbeđuju do 10 poena. Odgovore na pitanja studenti kucaju u bilo kom tekstualnom editoru na računaru i odmah po završetku, a pre pristupa praktičnom delu ispita (rešavanje zadataka) šalju preko Zimbre svom asistentu. Po predaji odgovora na pitanja, student prelazi na rešavanje datih zadataka, koji obezbeđuju do 20 poena. Da bi student položio ispit mora da ima najmanje po 50% poena predviđenih i od odgovora na pitanja i od rešenja zadatih zadataka.

**Lekcija 1: Osnove projektovanja softvera**

Šta omogućava apstrakcija unutar projektovanja softvera?

Apstrakcija omogućava uklanjanje detalja iz opisa problema uz zadržavanje suštinskih osobina svoje strukture.

Na koji način se vrši podela procesa projektovanja na projektne aktivnosti?

* zahtevanje rešenja
* kreiranje modela rešenja
* izvršavanje evaluacije modela shodno početnim zahtevima
* elaboracija specifikacije rešenja

Kako faza održavanja ima uticaj na proces projektovanja softvera?

1. Održavanje koje se bavi proširivanjem i unapređenjem operativnog softvera uz dodavanje novih funkcionalnosti (Perfective maintenance)

2. održavanje kojim se vrše potrebne promene nametnute van specifikacije zahteva (primer može biti izmene u zakonodavstvu) (Adaptive maintenance)

3. održavanje koje za cilj ima ispravku uočenih grešaka u toku operativnog rada sistema (Corrective maintenance)

Kako testiranje softvera može ispuniti ciljeve verifikacije i validacije?

Kako unaprediti proces testiranja softvera tako da odgovara ciljevima verifikacije i validacije procesa projektovanja softvera?

1. Šta se podrazumeva pod razmenom znanja projektanta sa drugim učesnicima u procesu projektovanja softvera?

Poznavanje domena aplikacije i mapiranje strukture problema i eventualnog rešenja, veštine prenosa vizije drugim učesnicima(objašnjenje) i identifikovanje napretka.

1. Šta su kanali komunikacije projektanta softvera?

Kanali komunikacije predstavljaju nacin informisanja projektanta softvera o domenu znanja, ograničenjima planovima i ostalim informacijama koje se odnose na projekat, takođe, projektant mora da odgovara njima.

1. Koja je uloga ograničenja u procesu projektovanja softvera?

Ograničenja služe da ograniče ukupni prostor mogućeg rešenja projektantu softvera. Ograničenja su specifična za svaki novi identifikovani problem i potrebno ih je razmotriti u svakoj fazi projektovanja softvera

1. Šta je metoda prepoznavanja?

Metoda u okviru koje projektant softvera prepoznaje rešenje problema koje je bilo u samom identifikovanom problemu

1. Na koji način proces projektovanja softvera unapređuje proces razvoj softvera?

Problem se definiše kroz određenu specifikaciju zahteva a zadatak projektanta softvera je da omogući opis rešenja problema i način na koji će zahtevi biti ispunjeni. Svrha projektovanja softvera je da jednostavno proizvede rešenje uočenog problema.

1. Šta je dugoročni plan izmena? U kojim slučajevima se koristi?

Dugoročni plan izmena softvera sadrži nekoliko faza i aktivnosti kroz je potrebno proći u procesu projektovanja.

1. Objasniti da li proces projektovanja usložnjava razvoj softvera ili ga čini jednostavnijim?

Svrha projektovanja softvera je da jednostavno proizvede rešenje uočenog problema.

1. Šta je model rešenja?

Model rešenja predstavlja grubi prikaz rešenja projekta i treba da sadrži sve zahteve koji su dobijeni od strane naručioca softvera i ne mora biti konačan.

1. Kako može doći do neusaglašenosti u procesu projektovanja softvera?

Prilikom preraspodele unutar procesa projektovanja i upoređivanja informacija, u toku razvoja ili u toku implementacije softvera kada je otklanjanje greške i najskuplje.

**Lekcija 2: Arhitektonske strukture, pogledi i stilovi**

1. Kada je potrebno ostvariti visoke performanse softvera, na koji način je to moguće izvršiti kroz primenu softverske arhitekture?

Ispitati potencijalni paralelizam kroz dekompoziciju sistema i sinhronizaciju sistema kao i identifikovati potencijalna uska grla performansi

1. Šta predstavlja cilj pisanja softverske dokumentacije?

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

1. Na koji način se određuju korisnici dokumentacije softverske arhitekture?

Članovi tima koji imaju različite uloge u timu, korisnici softvera, naručioci softvera

1. Šta omogućavaju različiti pogledi na softversku arhitekturu?

• pregled elementa softverske arhitekture iz više različitih uglova

• dokumentovanje elemenata iz više različitih uglova

• uvid u atribute kvaliteta određenog elementa softverske arhitekture

• različite ciljeve i različite upotrebe softverske arhitekture

1. Šta se postiže dokumentovanjem različitih pogleda na arhitekturu softvera?

Objašnjenje projektne dokumentacije i navigaciju kroz nju, opisne informacije pogleda kao i njihov način funkcionisanja, ograničenja i obrazloženja kao i informacije o upravljanju softverom i način funkcionisanja istog.

1. Za koje aktivnosti u toku procesa razvoja softvera koristimo dokumentaciju softverske arhitekture?

Upoznavanje ljudi sa sistemom, odnosno kao sredstvo obrazovanja i kao sredstno komunikacije između strana zainteresovanih za razvoj softvera.

1. Koji koraci su potrebni da bi se izvršilo kombinovanje različitih stilova softverske arhitekture?

Potrebno je da dokumentacija sadrži: uvodni deo(kompletni paket i uputstva za navigaciju), opisni deo(informacije o povezanosti i načinu funkcionisanja), deo ograničenja i obrazloženja i deo sa informacijama o upravljanju softverom.

1. Šta je element softverske arhitekture?

Svaki element softverske arhitekture predstavlja rezultat pažljivog i preciznog projektovanja u cilju ispunjavanja zahteva za kvalitet softvera a i zahteve nametnute od strane logike poslovanja organizacije za koju se softver projektuje.

1. Zašto je potrebno pisati softversku dokumentaciju iz ugla čitaoca?

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

1. Kako primena pravila za pisanje softverske dokumentacije utiče na proces pisanja softverske dokumentacije?

Pravilo podrazumeva da projektant softvera u svakom trenutku razmišlja o krajnjem cilju dokumentacije softvera koji projektuje a to je da dokumentacija treba da bude od koristi određenim korisnicima.

1. Na koji način se vrši odabir elemenata koji se predstavljaju u softverskoj dokumentaciji?

Na osnovu imena, kratkog oisa i uloge ili odgovornosti u arhitekturi, svojstva najboljeg i najgoreg slučaja, maksimalni broj događaja.

1. Koje su prednosti pisanja dokumentacije u toku procesa razvoja softvera?

U slučaju izmene zahteva, potražnje ili performansi ili eventualnog odustajanja od određenih funkcionalnosti izmena dokumentacije je znatno efikasnija, jeftinija i jednostavnija nasuprot već urađenoj dokumentaciji.

1. U kojim slučajevima se koristi formalna notacija?

U slučajevima kada su poledi bazirani na preciznosti i matematičkoj semantici

1. Da li možete identifikovati neke nedostatke pisanja softverske dokumentacije?

Zahtevno, potrebno dodatno planiranje i dodatno utrošeno vreme za kreiranje adekvatne dokumentacije koja bi se trebala barem u teoriji koristiti pri radu na sofrtveru dok se u realnosti slabo koristi jednom kada je softver kreiran.

**Lekcija 3: Stilovi softverskih modula**

1. Kada se koristi izraz modul softvera?

Projektanti softvera koriste izraz modul za objašnjavanje softverske strukture uključujući jedinice programskog jezika kao što su: C, Java ili C# klase, PL/SQL procedure ili Java paketi.

1. Šta omogućavaju različiti pogledi na softversku arhitekturu?

Pogledi na softversku arhitekturu omogućavaju obrazovanje (budućih članova razvojnog tima, korisnika, naručioca softvera), komunikaciju (između različitih korisnika sistema), osnovu za izgradnju i analizu

1. Koja je osnovna relacija stila razlaganja?

Osnovna relacija ovog stila je "je deo od" relacija koja prikazuje da element može biti deo drugog elementa

1. Na koji način je stil razlaganja pogodan za nove članove razvojnog tima?

Pregled stila razlaganja je pogodan za proces učenja o sistemu. Novi članovi se mogu fokusirati samo na određeni deo sistema od koga žele da krenu i zatim pratiti veze sa ostalim identifikovanim modulima sistema.

1. Po čemu se stil upotrebe razlikuje od stila razlaganja?

Relacija je ’’zavisi od’’, dok je kod razlaganja ’’deo od’’, nema ograničenja, čak se preporučuje korišćenje petlji dok se kod razlaganja zabranjuju. Koristi se za debagovanje i testiranje kao i utvrđivanje efekata promene, dok se stil razlaganja koristi kako bi se olakšao zazvoj članovima razvojnog tima.

1. Šta podrazumevaju inkrementalni podskupovi modula?

Definisanje inkrementalnih podskupova podrazumeva da moduli moraju biti pravog nivoa granularnosti.

1. Koje su prednosti stila generalizacije na raspolaganju projektantu softvera?

Generalizacija se koristi kod OO projektovanja (nasleđivanje), omogućava ponovnu upotrebu kao i lokalne promene jer ima relaciju ’’je’’ odnosno samo produžetak već poznatog modula.

1. Na koji način je omogućeno "produžavanje" modula primenom stila generalizacije?

Generalizacija predstavlja mehanizam za proizvodnju inkrementalnih opisa u cilju formiranja potpunog opisa modula.

1. Kako se prikazuje softverska arhitektura kroz stil slojeva?

Stil slojeva vrši podelu softvera na jedinice koje mogu da koriste druge jedinice koje se nalaze na nižem nivou.

1. Koji stilovi softverske arhitekture su pogodni za korišćenje uz stil slojeva?

Stil razlaganja - slojevi u slojevitom pogledu su uvek povezani a sloj može da prikaže više od jednog modula. Dva podmodula mogu biti deo različitih slojeva.

1. Šta su aspekti?

Moduli koji su odgovorni za za unakrsno kretanje podataka moraju biti postavljeni u jedan ili više aspekata gledišta. Primenom aspekata omogućeno je poboljšanje modifikacije modula.

1. Koje informacije sadrži prikaz aspekta u softverskoj dokumentaciji?

Prikaz aspekata treba da sadrži informacije o svakom modulu i načinu povezivanja sa drugim modulima koji zahtevaju funkciju unakrsnog kretanja

1. Kada se koristi model podataka?

Kada želimo da predstavimo strukturu informacija u vidu entiteta podataka i njihovih međusobnih odnosa (relacija).

1. Koji su tipovi modela podataka?

Konceptualni (E-R), Logički(upravljanje podacima) i Fizički(implementacija).

1. Da li primena stilova arhitekture usložnjava ili olakšava proces projektovanja softvera projektantu softvera?

Iako zahteva dodatni vremenski period prilikom razvoja, u procesu održavanja se znatno olakšava rad dodatnim objašnjenjima

**Lekcija 4: Stilovi povezivanja softverskih komponenata**

1. Na koji način se u neformalnoj notaciji predstavljaju komponente i konektori?

Kutije predstavljaju komponente a konektori su predstavljeni linijama.

1. Koja svojstva ima elemenat komponenta u pogledu komponente i konektora softverske arhitekture?

Komponenta predstavlja glavne procese i jedinice kao i skladišta podataka, komponente komuniciraju međusobno preko skupova portova (interfejsa) posredstvom konektora koji imaju skup uloga, odnosno servisnih usluga.

1. Kada se koristi stil toka podataka?
2. Koja je osnovna razlika između klijent-server stila i stila peer-to-peer?

Kod p-t-p stila bilo koja komponenta može da komunicira sa bilo kojom drugom komponentom(Skype), dok kod k-s komponenta koja poziva uslugu pauzira sve dok ta usluga nije završena

1. Šta podrazumeva grafičko predstavljanje pogleda?

Grafičko predstavljanje pogleda prikazuje broj i vrstu interfejsa na komponentama i konektorima, koristi apstrakcije komponenti i konektora u slučaju kada su oni previše složeni i deluje kao ključ za dokumentaciju sa detaljima o elementima, relacijama i njihovim svojstvima.

1. Koja je svrha elementa konektor u pogledu komponente i konektora?

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.

1. Koja su svojstva koja se koriste u pogledu komponente i konektora?

Pouzdanost (verovatnoća prestannka rada komp ili kon), performanse (testiranje potrebnog vremena za odgovor), potrebni resursi, funkcionalnost (koju funkciju obavlja komp ili kon), bezbednost, konzistentnost i nivo.

1. Kada se koristi stil komponente i konektora za predstavljanje softverske arhitekture?

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

1. Koji pogledi na softversku arhteikturu mogu biti kombinovani sa pogledom komponente i konektora?

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

1. Koji stilovi softverske arhitekture spadaju u pogled komponente i konektora?

Peer-to-peer, poziv i povratak, stil zasnovan na događaju.

1. Šta je pružalac usluga? Koja su njegova svojstva?

Pružalac usluga je element Servisno-orjentisanog stila, on omogućava jednu ili više usluga kroz interfejs, može uključiti ograničenja i ovlašćenja kao i performanse.

1. Koji su elementi sistema koji koristi klijent-server softversku arhitekturu?

Klijent(poziva servise serverske komponente) i server koji omogućava servise klijentu.

**Lekcija 5: Stilovi alokacije i hibridni stilovi**

1. Šta se dobija specijalizacijom stilova alokacije?

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

1. U kojim stilovima alokacije je potrebno prikazivati hardverske elemente okruženja softvera?

Kod stila raspoređivanja, softverske komponente i hardverske komponente na kojima se sistem izvršava

1. Koja je osnovna relacija stila alokacije?

Relacija koja se koristi u stilu alokacije je dodeljen prema ("allocated to")

1. Na koji način se vrši specijalizacija stila raspoređivanja?

Propisuje se raspodela softverskih komponenti u višeslojnoj arhitekturi na hardverske elemente koji se koriste za pokretanje

1. Zašto je neophodno kombinovati različite poglede na softversku arhitekturu?

Pošto je softverska arhitektura podeljenja na više pogleda (delova), kombinacijom više pogleda možemo korisniku, odnosno čitaocu dokumentacije prikazali povezanost između pogleda i samim tim se smanjuje broj zasebnih pogleda na SI.

1. Da li je moguće kombinovati bilo koji pogled softverske arhitekture sa nekim drugim pogledom?

Moguće je npr: dekompozicija i generalizacija, Servisno-orijentisani i komunikaciono-procesni

1. Kada se primenjuje stil dodeljivanja radnih zadataka? Na koji način se vrši primena 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

1. Koja je razlika između predstavljanja softverske arhitekture kroz stilove alokacije i kroz 4+1 Kručtenove poglede?

Kručtenovi pogledi (4+1) omogućavaju prikaz softverske arhitekture kroz različite poglede obuhvatajući kompletan pregled sistema

1. Šta omogućava primena 4+1 Kručtenovih pogleda na softversku arhitekturu?

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

1. Zašto je potrebno kombinovati stil raspoređivanja i stil instalacije?

Stil raspoređivanja omogućava prikaz povezanosti elemenata softverskog okruženja i elemenata softverske arhitekture. Stil instalacije prikazuje komponente softverske arhitekture koje je potrebno alocirati na određene hardverske komponente platforme na kojoj se sistem izvršava.

1. Šta je specijalizacija stilova alokacije?

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

1. Šta su kombinovani pogledi na softversku arhitekturu? Koji su načini za kreiranje kombinovanje pogleda?

Kombinovani pogledi predstavljaju povezanost više pogleda na softversku arhitekturu. Kombinovanje pogleda na softversku arhitekturu zahteva primenu asocijacija od jednog elementa određenog pogleda ka drugom elementu drugog pogleda.

**Lekcija 6: Upotreba šablona projektovanja softvera**

1. Navedite primer kada je moguće koristiti šablon Abstraction-Occurrence.

Kada koristimo instance jedne klase koje su dovoljno slične, ali se ipak vidno razlikuju (epizode TV serije, avionski letovi, isti broj ali drugi termini, kopije knjiga u biblioteci – koriste ih različiti korisnici i drugačiji barkod).

1. Koja su ograničenja u primeni šablona Abstraction-Occurrence?

Potrebno je prikazati članove svakog skupa pojava bez dupliranja zajedničkih informacija što rezultira upotrebom nepotrebnog dodatnog prostora, kao i izbeći nekonzistentnost gde bi se zajednički podaci ažurirali samo u pojedinim objektima umesto u svim gde se oni koriste.

1. Da li šablon General Hierarchy koristi nasleđivanje?

Da, jedna instanca je , odnosno čvor postaje super čvor i ima ostale čvorove ispod sebe(zaposleni je apstraktna klasa koju nasleđuju svi zapošljeni dok je menadžer super čvor koji je nadređen ne superiornom čvoru)

1. Koja asocijacija se koristi u šablonu General Hierarchy?

Asocijacija ’’podređen’’, koja može biti opciono na više i više na više.

1. U kojim situacijama je potrebno koristiti šablon Player-Role?

Primena P-R šablona može rešiti problem predstavljanja klasa različitog tipa u okviru klasnog dijagrama

1. Šta predstavlja "uloga" (role) u primeni šablona Player-Role?

Uloga ("role") predstavlja skup mogućnosti povezanih sa objektom u određenom kontekstu gde on može "igrati" određene uloge.

1. Da li primena šablona Delegation omogućava uštedu resursa potrebnih za razvoj softvera?

Da, ne kreiramo podklase koje će pokrenuti metodu, već koristimo delegator koji poziva samo tu metodu koja se nalazi u određenoj klasi.

1. Da si primenom šablona Delegation duplira programski kod?

Ne duplira se kod, samo se kreira delegator koji poziva određenu metodu umesto kreiranja potklase.

1. Šta predstavlja nepromenljiv objekat?

Immutable objekat je objekat koji je nepromenljiv, tako da ti objekti mogu verovati svom sadržaju jer je fiksan, odnosno onakav je kakav je i kreiran, bez izmena.

1. Koji su srodni šabloni šablonu Immutable?

Read-only interface.

1. Kada se koristi šablon Read-Only Interface?

Primenom šablona Read-Only Interface moguće je definisati klase sistema koje mogu da modifikuju atribute objekata koji su nepromenljivi

1. Koje su prednosti korišćenja šablona Read-Only Interface?

Mogućnost kreiranja klase koja može da izmeni atribute nepomenljivih objekata.

1. Šta je klasa "teške kategorije"? (heavyweight class)?

Klase za čiju instancu je potrebno dosta vremena za pristup, najčešće u bazi.

1. Koji su srodni šabloni šablonu Proxy?

Delegation i može se koristiti zajedno sa Proxy

1. Šta predstavlja primetan sloj?

Primetan sloj predstavlja neki observer, odnosno klasa koja implementuje <<Oblervable>> i koja osluškuje dešavanja ili poruke.

1. Koji su glavni nedostaci primene šablona projektovanja?

Razvijanje šablona je teško. Potrebno je proceniti da li upotreba šablona može unaprediti sistem u okviru koga se primenjuje.

1. Na koji način primena šablona omogućava povećanje fleksibilnosti projektovanog sistema.

Objektni klijent-server okvir (OCSF) omogućava jednostavan način za brzo postavljanje klijent-server aplikacije

**Lekacija 7: Strategije i metodi projektivanja softvera**

1. Na koji način se dobija deklarativno znanje o korišćenju metoda?

Deklarativno znanje predstavlja kroz opis: "uraditi to, a onda uraditi to.

1. Kako skup heuristika utiče na aktivnosti u procesu projektovanja softvera?

Omogućava smernice o načinu definisanja određenih aktivnosti unutar procesnog dela i organizacije određenih klasa problema i baziraju se na iskustvu.

1. Navedite primer tehničkih problema koji mogu nastati primenom metoda projektovanja softvera.

Tehnički problemi se sastoje od problema vezanih za projektovanje, mehanizam prenosa znanja – predviđanje mogućih grešaka i toka razvoja. Primeri su aplikacije za obradu transakcija u kojima su ključni kriterijumi pravilno obrađivanje podataka (podaci o finansijama, hotelske rezervacije) a ne postoji vremensko ograničenje za određene procese ili transakcije.

1. Šta je dizajn virtuelna mašina i u kojim slučajevima se koristi?

Predstavlja okvir koji projektant softvera treba da razvije i pridržava se u toku projektovanja. Primena dizajn virtuelne mašine omogućava definisanje skupa pretpostavki o rešenju problema u procesu projektovanja softvera.

1. Šta su dekompozicione metode? Kako se primenjuju?

Odozgo na dole i odozdo na gore.

1. Na koji način je moguće primeniti proceduralni model procesa projektovanja softvera?

Metoda projektovanja softvera može biti opisana kroz sledeće komponente: • deo predstavljanja • procesni deo • skup heuristika

1. Na koji način se vrši projektovanje po kompoziciji?

Suprotno od dekompozicije (odozgo na dole), kompozicija (na gore) je zasnovana na razvoju opisa za određeni skup entiteta objekata koji mogu biti prepoznati kao mogući problemi. Pored opisa entiteta, opisuju se i veze između tih entiteta

1. Koji su nedostaci u primeni metoda projektovanja softvera?

Metod projektovana softvera ne omogućava automatsko uklanjanje svih problema, nijedna od navedenih smernica ne može biti specifična za problem, potrebno je razviti ideju za određeni problem.

1. Kako se izvršava transformacija u procesu projektovanja softvera?

Predstavlja korak u kome projektant softvera modifikuje strukturu modela sistema. Sastoji se od ponovne interpretacije kroz drugačije poglede.

1. Kako faktori organizacije utiču na proces projektovanja softvera?

Metod organizacionog procesa projektovanja predstavlja oblik koji snažno utiče na proces projektovanja kroz ne-tehničke faktore koji proizilaze iz prirode i strukture organizacije.

1. Kako faza elaboracije i transformacije utiče na proces projektovanja softvera?

Proces transformacije zahteva od projektanta softvera da premosti svoje ideje preko dva ili više različitih skupova svojstva dok se proces elaboracije više bavi prepoznavanjem šablona i struktura.

1. Koje informacije projektant dobija iz faza procesa projektovanja?

Deo predstavljanja, procesni deo i heuristike.

**Lekcija 8: Tradicionalni metodi projektovanja softvera**

1. Šta čitalac dokumentacije softverske arhitekture može da vidi iz "P-Spec" specifikacije procesa?

Ovakav tip specifikacije procesa, "P-Spec", omogućava tekstualni opis i funkcionalni pogled osnovnog procesa koji je predstavljen krugom u dijagramu toka podataka.

1. Šta je rezultat strukturnog projektovanja softvera?

Detaljni dizajn i dopuna strukturne analize podacima

1. Šta je cilj strukturne sistemske analize?

Odnosi se na modelovanje funkcija koje su usmerene ka problemu i definisanju skupa opisnih oblika koje se mogu koristiti za projektovanje arhitekture u cilju rešavanja problema.

1. Koje informacije projektant softvera beleži u rečnik podataka?

Najčešće obuhvata opise svih oblika podataka koji su prikazani ili pomenuti u dijagramu toka podataka ili specifikaciji procesa, notacije i primere.

1. Na koji način se predstavlja strukturni način projektovanja?

Pogled na strukturno projektovanje omogućen je kroz strukturni grafički prikaz, hijerarhija poziva kod stila poziv povratak.

1. Kako se vrši proces transformacije u SSA?

Kreiranje dijagrama konteksta, kreiranje dijagrama toka podataka, analiza transakcija, dopuna toka podataka, finalni strukturni dijagram.

1. Koje su aktivnosti koje je potrebno izvršiti u koraku "Analiza transakcije"?

Komponente "velikog projektovanja sistema" budu razdvojene na mrežu međusobno povezanih podsistema i transakcija između njih.

Kreiranje dijagrama konteksta, dijagrama toka podataka, analiza, dopuna dijagrama toka podataka, kreiranje finalnog dijagrama.

1. Kojim dijagramom se predstavlja JSP?

Strukturnim dijagramom koji predstavlja statičku strukturu podataka.

1. Šta je JSP proces?

JSP predstavlja programski orijentisanu metodu koja služi za: realizaciju procesa projektovanja softvera kao pojedinačnog sekvencijalnog procesa i definisanje ulaznih i izlaznih tokova podataka

1. Šta je predstavljeno dijagramom specifikacije sistema?

Interakcije između entiteta modela sistema

1. Šta je entitet u dijagramu strukture entiteta?

Entitet, u ovom kontekstu, predstavlja "aktivan" element koji se identifikuje kroz operacije procesa modelovanja

1. U kom slučaju se koristi JSD? Navesti i objasniti primer primene JSD.

Može se koristiti i za projektovanje klijent-server sistema sa naglaskom na dugoročne interakcije, modelovanjem procesa a ne objekata.

1. Šta je analiza transakcije u okviru procesa transformacije?

Glavna akcija unutar ovog koraka je da komponente "velikog projektovanja sistema" budu razdvojene na mrežu međusobno povezanih podsistema i transakcija između njih.

1. Koje su operacije u okviru analize transformacije?

**Lekcija 9: Ponovna upotreba softvera**

1. Zašto se pri razvoju softvera koristi i stari softver (ranije razvijene softverske komponente pripremljene za ponovnu upotrebu i dr.)?

Zbog mogućnosti da je takav softver već ranije kreiran i potpuno funkcionalan za nas i čija je kupovina jeftinija od izrade istog iz početka, ubrzavamo razvoj i povećavamo upotrebu standarda i smanjujemo rizike.

1. Koje su koristi od ponovne upotrebe ranije razvijenih softverskih jedinica?

Kupovina jeftinija od izrade istog iz početka, ubrzavamo razvoj i povećavamo upotrebu standarda i smanjujemo rizike.

1. Zašto ponovna upotreba softvera povećava pouzdanost softvera?

Softver je korišćen i prehodno testiran pre same kupovine

1. Zašto ponovna upotreba softverskih jedinica smanjuje rizik razvoja novog softvera?

Cena je poznata dok je kod rezvoja nestabilna, smanjena verovatnoća grešaka

1. Zašto ponovna upotreba softverskih jedinica povećava efektivnost upotrebe softverskih specijalista?

Razvijaju se komponente za ponovnu upotrebu nasuprot iscrpnom razvoju ispočetka

1. Zašto ponovna upotreba softverskih jedinica povećava usaglašenost sa standardima?

Aplikacije već koriste standarde, samo ih implementiramo

1. Zašto ponovna upotreba softverskih jedinica ubrzava razvoj softvera?

Kupujemo gotov proizvod

1. Koji se problemi mogu javiti pri razvoju softvera uz upotrebu ranije razvijenih softverskih jedinica? Objasni svaki od ovih problema.

Ponovna upotreba softvera može da poveća troškove: održavanja-ako nema izvornog koda(nekompaktibilne sa promenama sistema) i ako nedostaju alati za podršku(ne podržavaju razvoj), održavanja biblioteka(skup proces dok komponenta bi trebalo da koristi biblioteke), nalaženja i prilagođavanja komponenti.

1. Koje bi faktore imali u vidu pri izboru softvera za ponovnu upotrebu?

Vreme trajanja, znanje razvojnog tima, kritičnost i zahtevi, domen, platforma

1. Šta je radni okvir (framework) za razvoj softvera? Koje su tri kategorije radnog okvira?

Radni okvir (framework) je opšta (generička) struktura koja se proširuje radi kreiranja specifičnog podsistema ili aplikacije. Okviri infrastrukture, za nitegraciju posredničkih softvera, za aplikacije preduzeća

1. Objasni radni okvir za razvoj veb aplikacija (MVC šablon). Koju funkcionalnost obezbeđuje MVC šablon?

Radni okviri veb aplikacija podržavaju razvoj dinamičkih veb sajtova. Oni najčeše koriste MVC šablon koji odvaja stanje od korisničkog interfejsa.

1. Šta su povratni poyivi (callbacks)?

Povratni pozivi (callbacks) su metodi koji se pozivaju kao reakcija na događaje koje prepoznaje radni okvir

1. Šta je linija proizvoda (software product line)? Kako se projektuje jezgro softverskog sistema?

Linija softverskog proizvoda (software product line) je skup aplikacija sa zajedničkom arhitekturom i deljivim komponentama, pri čemu je svaka aplikacija specijalizovana prema zadovoljenju različitih zahteva. Jezgro sistema je projektovano tako da može da se konfiguriše i prilagodi potrebama različitih korisnika sistema

1. Koja je razlika između aplikacionog radnog okvira i linje softverskog proizvoda?

Linije softverskog proizvoda obično sadrže i domenske i platformske informacije. Dok ARO samo domenske informacije.

1. Kako se može razviti specijalizovani tip softverskog proizvoda primenom lije softverskog proizvoda.

Kreiranjem aplikacionog okvira kao osnove sistema, kreiranjem jezgra proširenjem radnog okvira specifičnostima kompanije

1. Kako se vrši konfigurisanje softverskog sistema tokom procesa njegovog razvoja?

Konfigurisanje u vreme projektovanja- prilagođavanjem komponenti, razvijanjem jezgra i konfigurisanje u vreme raspoređivanja – projektrovan za konfiguraciju od strane kupca(kalibraciju)

1. Kako se vrši konfigurisanje softverskog sistema u vreme njegovog rapoređivanja 'stavljanja u upotrebu)? Koji su nivoi konfigurisanja sistema?

Konfigurisanje sistema u vreme raspoređivanja se može izvršiti izborom potrebnih komponenti, definisanjem radnog toka i pravila za obradu informacija i definisanjem vrednosti parametara.Nivoi : izbor komponenata ,definisanje workflow-a i definisanje parametara.

1. Šta je komercijalni gotov proizvod ( a commercial-off-the-shelf ili COTS)? Koje su prednosti njihove primene?

Komercijalni gotov proizvod COTS) je softverski sistem koji se može dalje prilagođavati potrebama različitih kupaca bez promene njegovog izvornog koda i omogućava brzo raspoređivanje, prikaz funkcionalnosti, izbegavanje rizika, čuvanje resursa kao i olakšano održavanje koje obavlja proizvođač

1. Koji se problemi mogu javiti pri primeni gotovih komercijalnih softverskih proizvoda?

Neusaglašeni zahtevi, pogrešne i nepromenljive pretpostavke, težak i često pogrešan odabir, nedostatak znanja za razvoj, evolucija mimo želje korisnika.

1. Objasni razliku između COTS-sistemskih rešenja (COTS-solution systems) i COTS-integrisanih sistema (COTS-integrated systems)?

COTS sistemska rešenja sadrže generičku aplikaciju razvijenu od jenog proizvođača a koja je konfigurisana u skladu sa zahtevima kupca.

COTS integrisani sistemi integrišu dva ili više COTS sistema moguće proivedenih od strane različitih proizvođača, a da bi kreirali željenu aplikaciju.

1. Opiši tipičnu arhitekturu ERP sistema. Koja su ključna svojstva ove arhitekture?

ERP sistemi se koriste kod skoro svih velikih kompanija radi podrške nekim ili svim poslovnim funkcijama. Glavno ograničenje ovog načina ponovne upotrebe softevera je funkcionalnost koja je ugrađena u generičko jezgro sistema.

1. Navedi uobičajene aktivnosti konfigurisanja COTS-sistema, kao što su ERP sistemi.

1. Izbor zahtevanih funkcija od sistema (npr. izbor odgovarajućih modula)

2. Uspostavljanje modela podataka koji definiše kako se podaci organizacije strukturišu u bazi podataka sistema.

3. Definisanje poslovnih pravila koji se odnose na ove podatke.

4. Definisanje očekivanih interakcija sa spoljnim sistemima.

5. Dizajniranje ulaznih formi i izlaznih izveštaja generisanih od strane sistema.

6. Projektovanje novih poslovnih procesa koji su usaglašeni sa procesima definisanih od strane sistema.

7. Postavljanje parametara koji definišu kako se sistem raspoređuje na njegovoj platformi.

1. Kako bi izabrao COTS sistem? Koje faktore bi uzeo u obzir?

Funkcionalne zahteve, trajanje sistema, saglasnost korisnika sa ažuriranjima, domen i znanje razvojnog tima.

1. Koji se problemi mogu javiti pri razvoju softvera integracijom više COTS proizvoda?

**Lekcija 10: Projektovana softvera primenom komponenata**

1. Šta je softverska komponenta? Nevedite pet karakteristika softverskih komponenata i objasnite ih.

Softverska komponenta je jedinica sastava sa ugovorom specificiranim interfejsima i sa zavisnostima eksplicitnog konteksta.Softverska komponenta treba biti: standardizovana(zadovoljava standard za modele komponenata), nezavisna (može se primeniti bez drugih komponenata), kompozitna(interakcije preko interfejsa), raspodeljiva(može da radi kao samostalni enititet), dokumentovana(specificirana sintaksa, interfejsi i komponente).

1. Objasnite dve kritične komponente softverske komponente: nezavisnost i nuđene servisa posredstvom interfejsa komponente? Koja je uloga interfejsa u slučaju primene softverskih kompšonenata.

nezavisna (može se primeniti bez drugih komponenata), kompozitna(interakcije preko interfejsa), interfejsi se koriste za međusobnu komunikaciju komponenti.

1. Postoje dve vrste interfejsa kompšpnenata. Koje su to vrste i objasnite razliku između njih.

Interfejs " provides" definiše servise koje obezbeđuje odnosno pruža komponenta i interfejs "requires" specificira servise koje treba da joj obezbede druge komponente da bi komponenta ispravno radila.

1. Šta je model komponente? Koji su osnovni elementi modela komponenata? Šta oni sadrže?

Model komponente je standard za implementacijum dokumentaciju i raspoređivanje komponente, elementi su interfejs, potreba i raspoređivanje.

1. Koja je razlika između servisa platforme i servisa podrške?

Servisi platforme, koji omogućavaju komponentama da komuniciraju i međusobno usaglašeno rade u distribuiranom okruženju i podređeni su svim sistemima na platformi dok se servisi podrške nude kao zajednički servisi koje mogu koristiti različite komponente.

1. Procesi softverskog inženjerstva zasnovanim na komponenatama omogućavaju projektovanje softvera primenom komponenata (PSPK) ili, na engleskom, CBSE procesi . Postije dva tipa PSPK: Razvoj za ponovnu upotrebu i Razvoj sa ponovnom upotrebom (development with reuse). Koja je razlika između ova dva tipa projektovanja softvera primenom komponenata?

Razvoj za ponovnu upotrebu se bavi razvojem komponenata ili servisa koji će biti ponovo korišćeni u drugim aplikacijama dok razvoj sa ponovnom upotrebom (development with reuse) je proces razvoja novih aplikacija koji upotrebljava postojeće komponente i servise.

1. Razvili ste softver i razmišljate da li da na osnovu njega napravite nekoliko kompoenata koje bi kasnije mogli da koristiti pri razvoju softvera. Kako ćete odlučiti? Koje faktore ćete uzeti u obzir prilikom odlučivanja?

Da li te komponente imaju široku primenu, da li mogu da funkcionišu kao zasebne jedinice, odnosno da li su potpuno nezavisne, da li je njihova izrada jednostavna ili kompleksna i zahtevna.

1. Pri korišćenju softverskih komponenata, postavlja se pitanja rada sa izuzecima.

svaka komponenta bi trebalo da definiše izuzetke koji se mogu javiti i trebalo bi da ih javno objavi , kao deo njenog interfejsa gde može doći do zatrpavanja interfejsa ili dovođenje do toga da komponenta postane zavisna od lokalne obrade izuzetaka.

1. Šta se podrazumeva pod upravljanjem komponentama?

Razvoj komponente za ponovnu upotrebu

1. Navedite aktivnosti procesa projektovanja softvera primenom komponenti (PSPK). U čemu je specifičnost ovog procesa u odnosu na projektovanje softvrea bey primene komponenti?

PSPK proces sa ponovo upotrebljenim komponentama sadrži aktivnosti koji nalaze i integriše ove komponete, to znači da ove komponente moraju da postaje u vreme početka razvoja.

1. Postoje tri vrste spajanja (sastavljanja) komponenata pri razvoju softvera. Nevedite te tri vrste i ukratko opišite.

Sekvencionalno (kreiranje nove spajanjem 2 postojeće), hijerarhijsko (jedna komponenta poziva servise druge) i aditivno (sastavljanje radi zajedničke funkcionalnosti)

1. Pri povezivanju komponenata, često se suočavamo sa nekompatibilnim interfejsima komponenata. Koje su to nekompatibilnosti interfejsa?

Nekompatibilnosti parametara, operacija i nekompletnost operacija

1. Šta su adaptori i čemu oni služe pri spajanju komponenata?

Komponenta adapter konvertuje jedan interfejs u drugi.

1. Navedite najčešće odluke koje projektanti treba da donesu kada koriste komponente pri rayvoju softvera.

1. Koji sastav komponenata daje najveće efekte na zadovoljenje funkcionalnih i nefunkcionalnih zahteva sistema?

2. Koji sastav komponenata čini sistem lakši za prilagođavanje komponenata kada dođe do promena u zahtevima?

3. Koje ća svojstva nastati u sistemu? Ova svojstva se odnose na performanse i pouzdanost. Ovo možete utvrditi tek kada je sistem kompletiran i primenjen.

**Lekcija 11: Projektovane distribuiranih softverskih sistema**

1. Šta su distribuisani softverski sistemi? Zašto se oni koriste? Koje su prednosti koje njihova primena dovodi?

Distribuirani sistem je onaj koji se realizuje na nekoliko kompjutera, koriste se kako sistemi ne bi bili zavisni od hardvera. Prednosti su deljenje resursa, otvorenost, konkurentnost, proširivost i otpor u slučaju otkaza.

1. Šta utiče na performanse distribuiranih softverskih sistema?

Performanse distribuiranih sistema najviše zavise od propusnog kapaciteta računarske mreže, od njenog trenutnog opterećenja, kao i od brzina svih računara distribuiranog sistema.

1. Projektan softvera treba d abude svestan problema upravljivosti distribuiranim sistemom. Na koja pitanja projektan treba da obrati pažnju pri projektovanju sistema, kako bi oni bolje upravljiv? Objasnite zašto su ta pitanja relevantna.

Kako se mogu otkriti otkazi u sistemu, šta učiniti da se minimizira njihov negativni efekat na druge komponente sistema i kako se mogu izvršiti popravke sistema? Kako bi sistem bio otporan na otkaz.

1. Šta je posrednički softver, tj. middlver (middleware)?

Posrednički softver (middleware) se koristi za mapiranje logičkih resursa koje program poziva u stvarne fizičke resurse i njihovu međusobnu interakciju.

1. Šta su otvoreni distribuirani sistemI? Šta podrazumeva njihova «Otvorenost»?

DIstribuirani sistemi su najčešće otvoreni sistemi, što znači da su projektovani da primenjuju standardne protokole, što mogućava kombinovanje opreme i softverskih komponenti različitih proizvođača

1. Šta je proširivost (scalability) distribuiranih sistema? Koje su tr i dimenzije proširivosti sistem? Objasnite ih.

Proširljivost (scalability) je svojstvo softvera da obezbeđuje visoki kvalitet servisa i kada se zahtevi za korišćenjem sistema povećavaju. Postoje tri dimenzije proširivosti sistema: veličina (dodavanje resursa za opsluživanje više korisnika), distribucija (raštrkavanje komponenti bez gubitka performancsi), upravljivost(upravljanje sistemom čak i u nezavisnim organizacijama)

1. Šta je razlika između priširenja sistema (scaling-up) i povećanja sistema (scaling-out)?

Priširenje sistema (scaling up) podrazumeva zamenu komponenata sa moćnijim komponentama, a povećanje sistema (scaling out) podrazumeva povećanja povećanje broja istih komponenata, što je najčešće isplativije.

1. Distribuirani sistem su ranjive na napade, te su manje bezbedni od centralizovanih sistema. Zašto? Nevdite i objasnite vreste napada kojima su distribuirani sistemi izloženi.

Napadač, upadom u neki deo sistema "ulazi na mala vrata" u sistem, i onda može da pristupi i drugim delovima sistema. Presretanje (presret komunikacije), presecanje(bombardovanje čvora lažnim zahtevima i ne dozvoliti mu da odgovori na legitimni zahtev), promena( menjanje podataka), fabrikacija(generisanje lažne inf.).

1. Šta je kvalitet servisa? Koji problemi prate zahteve za većim kvalitetom servisa? Navedite neki primer.

Kvalitet usluga koje nudi distribuirani sistem odražava sposobnost sistema da isporući traženi servis pouzdano, sa prihvatljivim vremenom odziva, i prihvatljivim protokom informacija. Problemi nastaju kod neprihvatanja cene meksimalnog opterećenja ili kontradiktornost parametara QoS – pouzdanost i protok umanjuju jedan drugi.

1. Kako se upravlja otkazima u slučaju distriburanih softverskih sistema?

Upravljanje otkazima obuhvata primenu tehnika za obezbeđenje otpornosti na otkaze. To znači da sistem treba da pronađe komponentu u kojoj je došlo do otkaza i da sistem nastavi da pruža što više servisa, bez obzira na otkaz u nekoj komponenti

1. Postoje dva osnovna modela interakcija između čvorova distribuiranog sistema. Koja su to dva modela i objasnite ih.

Proceduralna interakcija uključuje računar koji poziva poznati servis koji nudi neku drugi računar i obično čeka isporuku tog servisa. Interakcija porukama uključuje računar koji definiše informaciju o onome što bi trabalo da sadrži poruka, koja se onda šalje drugom računaru

1. Šta je proceduralni poziv (RPC)? Šta je tačka povezivanja (stub)? Gde se nalazi udaljena procedura i koja je njena funkcija?

Proceduralni poziv koristi komponenta koja poziva drugu komponentu kao u slučaju pozivametoda. RPC zahteva "tačku povezivanja" (stub) za proceduru koju poziva da bi ona bila pristupačna računaru koji inicilizira poziv.

1. Šta je interakcija sa porukeama? Oja je učaga midlvera u tome?

Interakcija porukama normalno uključuje komponentu koja kreira poruku , preko midlver sistema poruka se šalje komponenti koja treba da je primi. Komponenta koja prima poruka, vrši konverziju (čitanje) poruke, vrši odgovarajuću obradu podataka, i kreira poruku za komponentu koja je poslala poruku, poruka se daje midlveru radi transmisije komponenti koja je poslala poruku.

1. Koja je razlika proceduralnog poziva (RPC) i razmene poruka? Objasnite..

U RPC obe komponente moraju biti raspoložive, odnosno zahteva sinhronizovanu komunikaciju dok kod razmene midlver preuzima i čuva podatke koje prosleđuju komponente.

1. Koje su najčešće funkcije midlvera? Kako midlver podržava interakciju čvorova?

Midlver se nalazi u sloju između komponenata sistema i operativnog sistema. Obezbeđuje podršku interakcijama komponenti, kao i više zajedničkih servisa svojim komponentama.

1. Navedite i objasnite nivoe slojevite arhitekture distribuiranih klijent-server sistema. Koji su problemi dvoslojne rahitekture?

Sloj prezentacije – prikaz i interakcija sa korisnikom, upravljanje podataka – generiše web str., sloj obrade – logika i funkcionalnost i baza podataka.

1. Koja je razlika klijent-server arhitekture sa «debelim» i «tankim» klijentom? Navedite prednosti i nedostatke primene «tankih» a posebno, «debelih» klijenata?

Model sa tankim klijentom - klijent primenjuje prezentacioni nivo, a sve ostalo je na serveru. Model sa debelim klijentom - deo ili cela aplikaciona obrada je na klijentu. Server upravlja podacima i funkcijama baze podataka.

Tanki klijent dobar za sisteme sa više klijenata, debeli klijent je dobar za jake mašine i velike transakcije(bankomat)

1. Koje su predmosti višeslojne arhitekture? A koji su njeni problemi?

Višeslojna klijent-server arhitektura omogućava da se različiti slojevi sistema mogu nezavisno izvršavati jer se izvršavaju na različitim procesorima.

1. Koja je korist od primene sistema sa distribuiranim komponentama

Odlaganje odluka za realizaciju servisa, otvorena arhitektura, fleksibilnost i proširljivost sistema i dinamičko rekonfigurisanje sistema

1. Koji su nedostaci sistema sa distribuiranim komponentama?

Složenije su za projektovanje a koriste nestandardizovani midlver.

1. Na kojim principa rade sistemi ravnopravnih računara (peer-to-peer, or p2p)? Kako su njeniu čvorovi povezani? Kada se koristi p2p arhitektura?

Decentralizovani sistemi u kojima obrada podataka može da se vrši u bilo kom čvoru mreže. Nema nikakve razlike između između servera i klijenta, standardi i protokoli koji omogućavaju komunikacije među čvorovima. Koriste se kod velike obrade podataka i razmeni informacija u nekim slojevima.

1. Šta je polucentralizovana arhitektura sa ravnopravnim računarima? U čemu je njena prednost? Da li ona ima problem bezbednosti?

Polucentralizovana arhitektura ima , jedan ili više čvorova koji služe kao serveri za olakšavanje komunikacija među čvorovima. Ovo smanjuje količinu prenosa podataka između čvorova. Da zbog direktne komunikacije sa ostalim računarima.

1. Šta e «softver-kao-servis» (SaaS) ? Koji su ključni elementi SaaS koncepta? Koje su njegove prednosti?

Softver kao servis podrazumeva instalaciju softvera u nekom udaljenom serveru koji omogućava pristup klijentima preko Interneta i veb pretraživača.

Softver na serveru, u vlasništvu provajdera i korisnici plaćaju usluge ili pregled reklama. Korisnici su se oslobodili troškova upravljanja softvera

1. Koji su problemi primene SaaS koncepta?

Glavni problem je u troškovima transporta podataka ka udaljenom servisu. Transport se realizuje u skladu sa brzinom koju mreža može da obezbedi, što u slučaju velike količine podataka može da potraje dosta dugo

1. Koja je razlika SaaS i SOA?

SaaS je način isporuke funkcionalnosti aplikacije korisnicima, dok je SOA tehnologija implementacije aplikacionih sistema

1. Na koje faktore morate d aobratite pažnjupri primeni SaaS?

Konfiguraciju, višestruki zakup i proširljivost

1. Šta je dinamičko konfigurisanje softvera? Šta dozvoljava dinamičko konfigurisanje?

Preko interfejsa konfiguracije, korisnici specificiraju svoje potrebe (opcije) u skladu sa čim se sistem dinamički konfiguriše tako da daje utisak da je korisnik jedini korisnik sistema. Brendiranje, poslovna pravila, proširenje baze i kontrola pristupa.

1. Šta je proširivost SaaS sistema? Koje su preoruke za realizaciju priširenja SaaS sistema?

Proširljivost je sposobnost sistema da radi sa povećanim brojem korisnika bez smanivenja ukupnog kvaliteta servisa, provajder može da koristi bazu podataka za različite korisnike koji su virtualno izolovani unutar te baze

**Lekcija 12: Servisno-orijentisano softversko inženjerstvo**

1. Šta je veb servis? Koji tip inyterfejsa koristi veb servis?

Labavo povezana, ponovo upotrebljiva softverska komponenta koja sadrži diskretnu funkcionalnost, koja može biti distribuirana i programski pristupačna. Interfejs veb servisa je interfejs koji obezbeđuje ( provides) i definiše funkcionalnost servisa i parametre.

1. Šta su servisno-orijentisani softverski sistemi?

Servisno-orijentisani sistemi imaju labavo povezane (loosely coupled) arhitekture u kojime su veze među servisima menjaju za vreme izvršenja sistema

1. Šta je koncept «softver kao servis»? Koja je korist od preimena ovog koncepta?

Sofver kao servis znači ponudu funkcionalnosti softvera udaljenim korisnicima na Internetu, umesto preko aplikacija instaliranih na računaru korisnika

1. Kakva je arhitektura servisno-orijentisanih softverskih sistema? Ko su akteri u servisno-orijentisanim sistemima? Koja je njihova uloga?

Servisno-orijentisani sistemi su sistemi koji primenjuju ponovo upotrebljive softverske komponente kojima se pristupa preko drugih programa., a ne direktno povezivanjem korisnika na sam servis. Nuđeni od strane bilo kog provajdera, javne informacije o servisu, izrada novih servisa, male aplikacije

1. Šta je XML? Yašta se on koristi?

jezik koji razume i čovek i računar, a koji definiše strukturu podataka, koristeći označen (tagovan) tekst, tj. koji koristi identifikator za određenim značenjem.

1. Šta je SOAP? Čemu služi?

standard za razmenu poruka koji podržava komunikaciju između servisa. On definiše neophodne i opcione komponente poruka koje razmenjuju servisi.

1. Šta je WSDL? Čemu služi?

standard za definisanje interfejsa servisa. On određuje kako se definišu servisne operacije (nazivi operacija, parametri, i njihovi tipovi) i veze servisa

1. Šta je WS-BPEL? Čemu služi?

standard definiše jezik za specifikaciju radnog toka (workflow), tj. precesno orijentisane programe koji koriste nekoliko različitih servisa

1. Šta je UDDI? Čemu služi?

standard pretraživanja definiše komponente specifikacije servisa koje pomažu potencijalnim korisnicima servisa da ga otkriju na Internetu

1. Navedite kljućne standarde sa veb SOA?

WS-Reliability Messaging, WS-Security, WS-Addressing, WS-Transakcije

1. Koji su nedostaci veb servisa?

Standardi za veb servise su kritikovani kao teški standardi, koji su isuviše opšti i neefikasni. Njihova primena zahteva značajne obrade podataka pri kreiranju, prenosu i interpretiranju XML poruka

1. Do čega je dovale primena nove paradigme u softverskom inženjerstvu – primena servisno-orijentisanog sistema. Šta radi servis?

Ova promena paradigme je sada ubrzana primenom inženjerstvom oblaka (cloud computing) u kome se servisi nude sa računarske infrastrukture instalirane kod provajdera, kao što su Google i Amazon.

1. Šta je URI? Čemu služi?

Uniform Resource Identifier- lokacija servisa u WSDL dokumentu je definisan u interfejsu, koji sadrži operacije koje servis podržava i definiše format poruke koja mu se šalje, odnosno, koju on može da primi

1. Šta je WSDL? Koja tri aspekta podržava WSDL?

WDSL specifikacija definiše tri aspekta veb servisa: šta servis radi, kako komunicira i i gde se nalazi:

1. Koji su elementi WSDL modela?

Uvodni deo, opcioni opis podataka, opis interfejsa, opis ulaznih i izlaznih poruka, vezivanje i specifikacija krajnje tačke.

1. Šta je resurs u konteksta servisno-orijentisanih sistema? Koje su osnovne operacije koje s emogu izvršiti nad jednim resursom? Koje četori akcije obezbeđuju HTTP i HTTPS?

Veb protokoli HTTP i HTTPS se zasnivaju na četiri akcije: POST, GET, PUT i DELETE. Oni mapiraju osnovne operacije resursa

1. Šta je RESTful servis? Koji princip projektovanja on podržava?

REST je je arhitektonski stil koji se bazira na prenosu predstavljanja resursa od servera do klijenta. CRUD.

1. Koja je primena RESTful servisa u kod servisa «oblaka», tj. klaud servisa?

mogu se istovremeno koristiti i RESTful i SOAP servisi (slika 2) za slučaj istog servisa ili istog resursa

1. Koji su problemi pri primeni \*RESTFul servisa?

Složeni interfejs, nema standarda za interfejse, sopstvena infrastruktura

1. Šta je inženjerstvo servisa? Koje zahteve opn mora da podrži?

Inženjerstvo servisa je proces razvoja servisa za korišćenje od strane servisno-orijentisanih aplikacija

1. Koje su tri faze inženjering servisa?

Utvrđivanje mogućih servisa, kada utvrđujete moguće servise koje bi mogli da iskoristite i kada definišete zahteve za servis.

Projektovanje servisa, kada projektujete logičke i WSDL interfejse servisa.

Implementacija i raspoređivanje servisa, kada razvijete i testirate programski kod komponeneta koje obezbeđuju servise i kada ga činite dostupnim korisnicima

1. Koja su tri osnovna tipa mogućih servisa? Objasnite ih.

Uslužni – servis za opštu funkcionalnost(menjačnica), osnovni servisi- poslovne funkcije (registracija), koordinacije ili procesa – više aktera

1. Koja je razlika servisa za zadatke od servisa za entitete?

Servisi za zadatke: Povezani su sa nekom aktivnošću.

Servisis za entitete: Povezani su sa nekim poslovnim entitetom

1. Kako bi birali servis preko Interneta? Na koja pitanja bi obratili pažnu?

Da li je nazvistan, da li mora da održava stanje, da li koristi neko van org. Različiti nefunkc. zahtevi?

1. Koje su faze projektovanja interfejsa servisa? Navedite operacije koje taj interfejs treba da podrži.

1. Logičko projektovanje interfejsa, kada utvrđujete operacije koje su vezane sa servisom, njihove ulaze i izlaze, kao i izuzetke koje su povezane sa ovim operacijama.

2. Projektovanje poruka, kada vi projektujete strukturu poruka koje se šalju i primaju od strane servisa.

3. Izrada WSDL dokumenta, kada prevodite vaše logičko projektno rešenje i projektno rešenje poruka u apstraktan opis interfejsa napisan u WSDL.

1. Šta je katalog interfejsa servisa? Šta on sadrži?
2. Kako se vrši implementacija i testiranje interfejsa?

Posle utvrđivanje mogućih servisa i projektovanja njihovih interfejsa, u poslednjoj fazi procesa servisnog inženjerstva se radi implementacija servisa

1. Kako se vrši raspoređivanje servisa, tj. omogućavanje korišćenja servisa? Koje informacije o servisu treba objaviti?

Raspoređivanje servisa omogućava korišćenje servisa, na veb serveru, od strane njegovih krajnjih korisnika. Za to je dovoljno postaviti fajl sa izvršnim softverom u određeni diretorijum veb servera

1. Kako se vrši sastavljanje i konfigursanje servisa?

Osnovni princip servisno-orijentisanog softverskog inženjerstva je sastavljanje i konfigurisanje servisa prilikom kreiranja novog, složenijeg servisa. Ovo se može integrisati sa korisničkim interfejsom na veb pretraživaču, ili se može koristiti kao komponenta koja se koristi pri kreiranju nekog drugog složenijeg servisa.

1. Navedi faze konstruisanja sistema sastavljanjem više servisa

Formulisanje šeme radnog toka, Otkrivanje servisa, Izbor mogućih servisa, Prerada radnog toka, Kreiranje programa radnog toka, Testiranje postavljenog servisa ili aplikacije.

1. Kako se vrši testiranje sastavlenog servisa? Koji su problemo ovog testiranja?

Testiranje ima za cilj da proveri da li servis ispunjava sve postavljene ciljevem tj. funkcionalne i nefunkcionalne zahteve i otkrivanje grešaka koje su napravljene u toku procesa razvoja. Oni mogu da Poglavlje 5 Sastavljanje servisa 30 zaviste od otkaza drugih servisa. Obezbediti da ovi servisi padnu za vreme testiranje je dosta teško

**Lekcija 13: Inženjerstvo softvera u realnom vremenu**

1. Šta su sistemi u realnom vremenu? U čemu je razlika ovih sistema u odnosi na ostale sofverske sisteme? Šta su “meki” i “tvrdi” sistemi u realnom vremenu?

Softverski sistem u realnom vremenu je sistem čiji ispravan rad zavisi i od rezultata koji proizvodi i od vremena u kome se ti rezultati proizvedu.

Pod "mekim" sistemima u realnom vremenu: podrazumevaju se sistemi koji neispravno rade ako se rezultati ne proizvode u skladu sa specificiranim vremenskim zahtevima. Pod "tvrdim" sistemima u realnom vremenu se podrazumevaju sistemi koji ne proizvode rezultate u skladu sa specifikacijom vremena reagovanja, i u tom slučaju, sistem je neuspešan;

1. Šta je podsticaj, a šta je ponašanje sistema u realnom vremenu? Koje vrste podsticaje postoje? Objsnite ih.

Podsticaj je događaj koji se javlja u okruženju softverskog sistem koji prouzrukuje da sistem reaguje na neki način – a to je signal ili poruka koje sistem šalje u svoje okruženje a ponašanja su odgovori sistema na signale.

Periodički podsticaji – u intervalima i aperiodički- prekid unosa podataka.

1. Koja je funkcija senzora, a koja- pokretača (actuator)?

Podsticaji dolaze od senzora u okruženju sistema, a odgovori se šalju pokretačima.

1. Navedite aktivnosti procesa projektovanja sistema u realnom vremenu.

Izbor platforrme, identifikacija podsticaja i odgovora, analiza vremena, projektovanje procesa, projektovanje algotama, projektovanje podataka, planiranje procesa.

1. Zašto je potrebna sinhronizacija procesa? Šta je cilj sinhronizacije? Šta je kružni bafer? Ćemu služe Get i Put operacije?

Ako proces koji proizvodi informaciju radi brže od procesa koji je koristi, nova informacija može zameniti prethodnu koju još nije preuzeo proces koji bi trebalo da je koristi. Da bi se izbegle ove situacije, potrebno je da koristiti bafere informacija i mehanizme za sprečavanje istovremenog pristupa podacijma u baferu. Put operaciju koristi proces koj proizvodi informaciju, a Get operaciju koristi proces koji koji koristi informaciju, tj. koji je preuzima iz bafera.

1. Šta je cilj statičke anaize sistema?

proverite da li vaš sistem zadovoljava postavljene zahteve korišćenjem poznate vremenske konstante ponašanja komponenti sistema ili koristite simulaciju rada sistema

1. Koji su problemi programiranja sistema u realnom vremenu?
2. Koji se prolemi javljaju pri programiranja sofvera u realnom vremenu, ako se primenjuje objektno-orijentisani programerski jezik? Šta se postiže primenom Java SE Real-Time Systems, verzijom Jave za programiranje sistema u realnom vremenu?

ona uključuje modifikovane mehanizme sa nitima, koji dozvoljavaju specifikaciju niti koje neće biti prekidane prikupljanje iskorišćenih podataka ( garbage collection).

1. Zašto je potrebno korićenje posebnih šablona projektovanja sistema u realnm vremenu? Da li se šabloni mogu kombinovati prilikom projektovanja softverskog sistema?

šabloni se mogu kombinovati pri projektovanju konkretnog sistema u realnom vremenu

1. Opiši šablon «Osmatraj i reaguj» (Observe and React), navodeći i njegove stimulanse, odgovore, procesi i upotrebu.

Moniring sistem istražuje svoj okruženje preko seta senzora i najšešće prikazuju njihovo stanje očitanih vrednosti. Stimulansi su senzori prikačeni na sistem,odgovori su prikazi, upozorenja, procesi posmatrač, analiza, prikaz, alarm, reaktor. Koriste se kod monitoringa sistema.

1. Opiši šablon «Kontrola okruženja» (Environmental Control), navodeći i njegove stimulanse, odgovore, procesi i upotrebu.

Kontroni sistemi mogu da koriste šablon Konrola okruženja za svaki tip pokretača kao opšti šablon za upravljanje senzorima i porkretačima. Sistem analizira inf. Sa senzora, odgovori su upravljački signali, procesi u monitor, upravljanje, prikaz, pogon pokretača. Koriste se u upravljačkim sistemima.

1. Opiši šablon «Procesni kanal» (Process pipline) navodeći i njegove stimulanse, odgovore, procesi i upotrebu.

Šablon Procesni kanal (Process Pipline) omogućava brzu obradu podataka jer obradu deli na sekvenciju posebnih transformacija. Podaci se dele posredstvom bafera. Stimulansi su ulazne vrednosti iz drugog bafera, odgovori su izlazi ka baferu i bazi, procesi su proizvođač, bafer i potrošač, koriste se za prikupljanje podataka.

1. Šta je analiza vremena i u čemu je njen značaj? Koja su tri ključna faktora projektovanja ugrađenih sistema?

proračunavate koliko često svaki proces mora da se izvrši da bi se obezbedila obrada svih ulaza i svih sistemskih odgovora tako da budu proizvedeni na vreme. Rokovi, Frekvencija i Vreme izvršavanja.

1. Zašto se primenjuju posebni operativni sistemi u realnom vremenu pri primeni softvera u realnom vremenu? Yašto se ne koriste uobičajeni operativni sistemi?
2. Koje su funkcije OS u realnom vremenu?
3. Navedite dva nivoa prioriteta upravljanje procesima i objasnite ih.
4. Šta su periodički procesi?
5. Šta su procesi vođeni prekidima?
6. Šta je planiranje bez predpražnjenja?
7. Šta je planiranje sa predpražnjenjem?
8. Kako se vrši upravljanje resursima u operativnim sistemima za rad u realnom vremenu?

**Lekcija 14: Projektovanje pouzdanog proizvoda**

1. Navedite i objasnite četiri kategorije grešaka.

1. Ljudske greške (human error): Ponašanje ljudi koje dovodi do unošenja grešaka u softverski sistem.

2. Sistemske greške (system fault): Karakteristika softverskog sistema koja dovodi do greške.

3. Greška u sistemu (system error): Pogrešno stanje sistema za vreme izvršavanja koje vodi od ponašanja sistema koje korisnici sistema ne očekuju. 4. Otkaz sistema (system failure): Događaj koji se pojavljuje u nekom vremenskom trenutku kada sistem ne isporučuje servis koji korisnici očekuju.

1. Navedite greške koje ne dobodi do otkaza rada (pada) sistema.

Ne izvršava se ceo program, Greška je privremena, Sistem sadrži mehanizam detekcije grešaka i zaštite.

1. Navedite pristupe koje vode poboljšanju pouzdanosti softvera.

Izbegavanje greške, detekcija i otklanjanje, tolerancija

1. Šta je pouzdanost (reliability), a šta je dostupnost (availability) sistema. Koja je povezanost pouzdanosti i dostupnosti sistema?

Dostupnost softvera (availability) znači da je servis sofvera raspoloživ i da on radi. Pouzdanost softvera (reliability) znači da servis isporučuje tačne rezultate

1. Šta je otkaz (kvar) sistema? Koja su dva problema koja često dovode do kvarova u sistemu?

Ako se skup sa pogrešnim ulazima češće koristi, to će dovesti i do češćeg dobijanja pogrešnih razultata. I suprotno, ako se skup ulaza Ie ređe koristi od strane sistema, onda softver izgleda kao da ima veću pouzdanost, jer koristici se retko suočavaju sa padom sistema.

1. Šta je pouzdanost sistema? Koja je metrika za pouzdanost?

Pouzdanost se može specificirati kao verovatnoća javljanja pada sistema kada je on radi u okviru specificiranog okruženja njegovog rada. Verovatnoća otkaza, stopa javljanja otkaza i dostupnost.

1. Šta je verovatnoća otkaza na zahtev – POFOD? Dajte primer. Kada se upotrebljava POFOD?

Ako koristite ovu metriku, vi definišete verovatnoću otkaza sistema kada zahtevate neki njegov servis. Na primer, POFOD= 0,001 znači da postoja 1/1000 šanse da sa javi otkaz pri postavljenjanju tražnje za servisom. POFOD se upotrebljava u situacijama kada pad pri zahtevu korišćenja sistema vodi do ozbiljnih padova sistema

1. Šta je stopa javljanja otkaza sistema – ROCOF? Dajte primer. Kada se koristi ROCOF?

Ova metrika definiše verovatnoću javljanja broja otkaza sistema relativno u odnosu na određeni vremenski period (na primer, na sat). Na primer, u gornjem primeru, ROCOF = 1/1000. ROCOF se koristi u situacijama kada kvar pri pozivu dovodi do ozbiljnog pada

1. Šta je srednje vreme otkaza – MTTF? Dajte primer. Kada se koristi MTTF?

Recipročna vrednost od ROCOF je srednje vreme otkazivanja ( mean time to failure - MTTF), koja se povremeno koristi kao metrika pouzdanosti. MTTF je prosečan broj vremenskih jedinica između otkaza sistema. ROCOF=2 znači javljanje dva otkaza po satu, a to znači da je srednje vreme otkaza 30 minuta.

1. Šta je dostupnost (availabity – AVAIL)? Dajte primer.

verovatnoća da će sistem raditi kada neko zahteva njegov servis. Na primer, dostupnost od 0,9999 znači da je sistem, u proseku, dostupan 99,99% u toku vremena rada

1. Šta definišu nepfunkcionalni zahtevi pouzdanosti?

Nefunkcionalni zahtevi pouzdanosti su specifikacije o zahtevanoj pouzadanosti i dostupnosti sistema upotrebom jedne od metrika pouzdanostu

1. Koja je korist od kvantifikovane pouzdanosti?
2. Navedite tri preporuke (uputstva) za specifikaciju pouzdanosti sistema.

Specificirajte zahteve dostupnosti i pouzdanosti za različite tipove otkaza, Specificirajte zahteve dostupnosti i pouzdanosti za različite tipove servisa sistema, Razmislite da li vam je visoka pouzdanost zaista potrebna.

1. Kako se postiže visok nivo funkcionalne pouzdanosti sistema? Šta obuhvata specifikacija funkcionalne pouzdanosti?

Visoki nivo pouzdanosti i dostupnosti softverskog sistema se postiže kombinacijom tehnika, kao što su izbegavanje grešaka (fault-avoidance), detekcija grešaka (fault-detection) i tolerancija grešaka (fault-tolerance).

1. Navedite četiri četiri tipa zahteva funkcionalne pouzdanosti.

Postoji četiri tipa zahteva funkcionalne pouzdanosti: zahtevi provere, zahtevi oporavka, zahtevi redudantnosti i zahtevi pouzdanosti.

1. Šta je tolerancija grešaka (fault tolerance)?

Tolerancija grešaka (fault tolerance) je pristup pouzdanosti u fazi izvršenja sistema koji uključuje mehanizme za nastavak rada i kada dođe do pojave grešaka u softveru ili hardveru, i kada je sistem u stanju greške

1. Šta sadrži arhitektura sistema koji je otporan na greške?

uključi redudantne (ponovljive) i različite hardverske i softverske komponente.

1. Šta su zaštitni sistemi? Šta je cilj dejstva zaštitog sistema? Opišite funkcionalnost zaštitnog sistema. Kako zaštitni sistem radi?

Zaštitni sistem sadrži samokritičnu funkcionalnost koja je neophodna da sistem prebaci iz nebezbednog u bezbedno stanje.

1. Šta je samoosmatrajuća arhitektura? Kako se ostvaruje samoosmatrajuća arhitektura?

Samoosmatrajuća arhitektura (self-monitoring architecture) je arhitektura u kojoj je sistem projektovan da osmatra sam svoj rad i da preduzima akcije ako je otkriven neki problem. Računanje se vrši posebnim kanalima, i rezultati ovih računanja sa upoređuju

1. Šta je trostruka modularna redudantnost-TMR?

Kod TMP sistema hardverske komponente se repliciraju tri puta. Izlaz iz svake jedinice se šalje u komparator izlaza koji se implementira u obliku sistema za glasanje

1. Šta je programiranje N verzija softvera? Kda ga primeniti?

Upotrebom iste specifikacije, različiti timovi razvijaju isti softverski sistem. Ovako razvijene verzije istog softvera se izvršavaju na različitim kompjuterima o u slučajevima kada je nepraktično obezbeđivanje zaštitnog sistema koji štiti sistem od grešaka opasnih po bezbednost

1. Koja su moguća dodatnia pravila kompanija?
2. Kako dolazi do pogrešne interpretacije specifikacije?
3. Koja su mogućna rešenja problema posgrešne interpretacije specifikacije?
4. Koje su preporuke za programiranje za pouzdanost?
5. Koji su potrebni podaci za utvrđivanje pouzdanosti softvera?
6. Koje su faze statističkog testiranja radi merenja pouzdanosti?
7. Koje su teškoće u statitskičkom testiranju?
8. Šta je profil rada?

**Lekcija 15: Analiza i ocena kvaliteta projektnog rešenja softvera**

1. Šta je cilj upravljanja kvalitetom softvera?
2. Kada se primenjuje razvoj vođen planom, tj. formalizovano upravljanje kvalitetom (quality management – QM)? Šta je specifično za formalizovano upravljanje kvalitetom na organizacionom i na projektnom nivou?
3. Koja je razlika upravljanja kvalitetom i kontrole kvaliteta softvera?
4. Koja je uloga QM tima (tima koji upravlja kvalitetom projekta).
5. Šta je plan kvaliteta? Koja je stuktura plana kvaliteta?
6. Zašto je subjektivnost prisutna u oceni kvaliteta softvera?
7. Na koje odgovore QM tim traži odgovore pri pravljanju kvalitetom softvera?
8. Navedite atribute kvaliteta softverskog proizvoda.
9. Opiši proces ocenjivanja kvaliteta softvera.
10. U slučaju fizičkih proizvoda, kvalitet proizvoda se postiže ako se poštuje specifikacija proizvodnog procesa. Međutim, kod softverskih proizvoda to nije tako. Zbog čega? Šta je specifičnost procesa razvoja softverskog proizvoda?
11. Koje su teškože u određivanju atributa kvaliteta?
12. Šta je to «kultura kvaliteta»? Kada možemo da kažemo da se u nekoj organizaciji neguje «kultura kvaliteta»
13. Šta definišu standardi proizvoda, a šta definišu standardi procesa?
14. Šta standardi softverskog inženjerstva definišu?
15. Inženjeri obično ne vole da primenjuju standarde. Zašto? Kako bi ubedio svoje kolege da primenjuju standarde?
16. Da li menadžer projekta može da modifikuje standard koji želi da primeni? Obrazložite svoj stav.
17. Šta je ISO 9001? Koji su ključni procesi koje obuhvata standard ISO 9001?
18. Šta je uputstvo o kvalitetu?
19. Navedite sitandarde koji zu od značaja z akvalitet softvera.
20. Šta je recenzija kvaliteta? Šta je sve predmet recenzije? Koja je korist od recenzije?
21. Navedite aktivnosti procesa recenzzije softvera.
22. Šta je kontrola softvera? Kako se vrši kontrola programa?
23. Klasifikujte greške koje se javljaju pri razvoju softvera.
24. Kako se vrši kontrolu kvaliteta softvera i donosi odluke o kvalitetu softvera.
25. Pri primeni Scrum metode razvoja softvera, programiranje se vrši u paru. Zašto? Međutim, pored dobrih strana, programiranje u paru ima i negative spekte. Koji su problemi kontrole pri programiranja u paru?
26. Kako se vrši merenje softvera? Šta su metrike softvera? Koje su dve vrste metrike softvera?
27. Navedite metrike kontrole procesa. Koja je razlika metrike predviđanja i metrike kontrole?
28. Šta su metrike softverskog proizvoda? Navedite primere
29. Merenje softverskog sistema se može koristiti na dva načina. Koja su ta dva načina? Opišite ih.
30. Navedite spoljne i unutrašnje atribute kvaliteta. Koji su uslovi predviđanja spoljnjih atribita?
31. Opišite kako bi izvršili analizu kvaliteta komponente.
32. Koja je razlika dinamičkih i statičkih metrika?