dr Alempije V. Veljović

P R A K T I K U M

iz

Projektovanja informacionih sistema

Beograd, 2005.

alempije@beotel.rs

Budi ljubazan prema ljudima dok se penješ,

jer ćeš ih sresti kad budeš silazio

Vil Durant

Predgovor

Praktikum je dodatna literatura za predmet Projektovanje informacionih sistema(PIS) i predstavlja dodatno objašnjenje za udžbenik Objektno modeliranje informacionih sistema (1). Ovaj praktikim nastao je kao rezultat rada sa studentima. Naime, pokazalo se da studenti neshvataju da je seminarski rad elaborat koji ima odgovarajuća poglavlja i da predstavlja prvi korak koji oni čine u pravcu izrade projekata vezanih za PIS. Autorova želja je da se shvati da to što se "isprogramira" mora sadržati tačno definisana poglavlja. Kako je cilj da se dobije korisnička aplikacija to se izlazi iz okvira ovog praktikuma i definiše se fizički model baze podataka i pogled na korisničku aplikaciju koji su dati u praktikumu za analizu informacioih sistema(2). Na ovaj način se studenti izlaze iz sveta programera i uvode u profesionalni svet projektanata informacionog sistema.

U uvodnom izlaganju definišu se osnovni principi UML korišćenjem RatinalRous CASE alata.

U sledećem poglavlju definišu se UML dijagrami kroz korišćenje CASE alata RationalRous. Prikazani primeri su rezultat izrade projekata za pojedina preduzeća (pogledaj literaturu), diplomskih radova i školskih primera korišćenih u nastavi. Primeri su namerno pojednostavljivani i predstavljeni kao informatička ostrva da bi studenti shvatili suštinu.

Prvi primer odnosi se na poslove fakturisanja kroz primer EDIFACT fakture. U ovom primeru su prikazani minimalni zahtevi za seminarski rad iz predmeta Projektovanje infromacionih sistema.

Drugi primer se odnosi na poslove praćenja ispita. Opisuje se proces prijavljuja ispit korišćenjem UML dijagrama, na osnovu kojih se izadjuje plan polaganja ispita i na kraju prati realizacija plana ispita i obradjuju spiskovi za ispit.

Treći primer odnosi se na poslove izrade tehnološkog procesa korišćenjem UML dijagrama i to za procese održavanje tehnoloških lokacija, izrada tehnološkog postupka, definisanje parametara i izrada tehnoloških izveštaja i pregleda.

Četvrti primer se odnosi na poslove cirkulacije u biblioteci korišćenjem UML dijagrama. Ona uključuje vođenje evidencije o članovima cirkulacije u biblioteci, zaduživanje i razduživanje članova sa naslovima, opominjanje korisnika i rezervisanje.

Svi ovi primeri mogu da budu uzor za izradu sopstvenih korisničkih aplikacija. Kako je ovo prvo izdanje ovakvog praktikuma to autor očekuje od budućih korisnika korisne sugestije i obećava da će drugo izdanje sadržat i neke druge primere.

Autor

Sadržaj

1. Uvod

U ovom praktikumu prikazan je na konkretnim primerima UML kao metod za opisivanje detalja arhitekture sistema korišćenjem CASE alata Rational Ruse. CASE alat Rational Rose je jedan od alata koji podržava razvoj aplikacija koristeći UML. On omogućava kreiranje dijagrama slučajeva upotrebe, aktivnosti, sekvenci, saradnje, stanja, komponenti i razmeštaja. Pomoću inženjeringa i reinženjeringa CASE alat Rational Rose omogućava generisanje programskog koda u programskim jezicima C++, Java, Visual Basic i XML DTD. Takođe, postoje dodaci za ostale objektno-orjentisane programske jezike.

Alat Rational Rose poseduje dodatne funkcionalnosti, kao što je razvoj Web aplikacija. Neke od novina koje donosi alat Rational Rose su: Modeliranje procesa, Dijagrami aktivnosti, Web modeliranje, Podrška za programski jezik Java i J2EE, Podrška za EJB, Podrška za XML DTD, Modeliranje podataka, Podrška za programski jezik ANSI C++, Podrška za SQL Server 2000 i Podrška za Sun Java Server Pages i Microsoft Active Server Pages.

Izdvojio bih na samom početku svoje mišljenje o modeliranju procesa jer smatram da za modeliranje procesa mnogo bolje koristiti semantički bogatiji standard IDEF0 prikazan u (2) i realizovan kroz CASE alat Bpwin..

Rational Rous ranijih verzija nije ima modelar podataka što je predstavljao veliki nedostatak a smatram da trenutni modelar može da poslužI samo kao prelazno rešenje za korišćenje profesinalnijih modeara podataka kao što je Erwin koji realizovan korišćenjem standarda IDEF1X i prikazan je u (2).

Ovaj praktikum je namenjena programerima, arhitektama softvera i analitičarima a pogotovo onima koji nemaju mnogo iskustva sa UML-om i nisu uključeni svi aspekti UML-a.

2. Vizuelno modelovanje

Modeli u svetu softvera su planovi za sistem. Planovi pomažu da se isplanira izgradnja pre nego što se stvarno krene na izradu aplikacija.Rezlutat procesa modelovanja je mogućnost da se prate poslovni zahtevi.

Vizuelno modelovanje je proces uzimanja informacija sa modela i njihovo grafičko prikazivanje korišćenjem niza standardnih grafičkih elemenata. Standardi su neophodni kako bi se izvukla bitna korist od modelovanja: komunikacija.

Komunikacija između korisnika, programera, analitičara, menadžera i svih onih koji su uključeni u projekat, je osnovna svrha vizuelnog modelovanja. Komunikaciju možete ostvariti korišćenjem nevizuelnih (tekstualnih) informacija, ali ipak ljudi mogu vizuelno mnogo lakše prihvatati i uočavati stvari.

Jedno od važnih pitanja u vizuelnom modelovanju je grafička notacija i koristi se za opis različitih aspekata sistema. Ta notacija mora da bude poznata svim zainteresovanim grupama, u suprotnom model neće biti koristan. Do sada su mnogi predlagali koju notaciju koristiti za vizuelno modelovanje. Neke od popularnih notacija, koje imaju jaku podršku, su Bučova notacija, OMT (eng. Object Modeling Technology) i UML.

Rational Rose podržava ove tri notacije, međutim, UML je standard koji je usvojen od većine korisnika i grupa za standardizaciju kao što su ANSI i OMG (eng. Object Managment Group). UML notacija je posledica saradnje Grady Booch-a, dr James Rumbaugh-a, Ivar Jacobson-a, Rebecca Wirfs-Brock, Peter Yourdon-a i mnogih drugih.

Na početku treba definisati granice do kojih će se ići u modelovanju poslovnih procesa. Da li se modeluje čitava organizacija ili samo jedno odeljenje? Koji tokovi poslovnih procesa su bitni za projekat ?. Granice sistema kao i tokove poslovnih procesa definisali smo korščenjem IDEF0 standarda realizovanim u BPwin CASE alatu u (2).

Kad se definiše obim projekta, veoma je važno okupiti pravi tim. Potrebni su pojedinci koji poznaju poslovne procese, kao i pojedinci koji poznaju modelovanje poslovnih procesa. Uopšteno, članovi tima ne moraju da budu informatičari, čak je bolje da ne budu. Informatičari vrlo brzo kreću ka rešenju, dizajnu sistema, ne analizirajući dovoljno poslovne procese.

Obavezni članovi tima su:

Vođa tima - On treba da poseduje dovoljno znanja i o poslovnim procesima i o modelovanju poslovnih procesa. On ili ona će biti zadužen za koordinaciju napora članova tima i za održavanje

pravca rada.

Predstavnik(ci) poslovnih procesa-Oni su predstavnici različitih delova organizacije koje modelujemo. Oni treba da su dosta upoznati sa tokovima poslovnih procesa, uključujući i probleme koji se u njima javljaju i dobit od tih tokova poslovnih procesa.

Predstavnik(ci) menadžmenta kompanije-Neko ko ima autoritet da odluči koji će delovi ili poslovni procesi biti modelovani. Oni mogu da pomognu timu da razume tokove poslovnih procesa iz perspektive menadžmenta.

Opisom poslovnih procesa korišćenjem BPwina dobjamo semantički bogat model koji predstavlja dobru osnovu za korišćenje UML dijagrama.

Model u Rose-u je slika sistema iz različitih perspektiva. Takav model uključuje sve UML dijagrame, aktere, slučajeve upotrebe, objekte, klase, komponente i čvorove razmeštaja elemenata sistema.Ovakav model opisuje šta će sistem sadržati i kako će raditi, tako da inženjeri koji razvijaju sistem mogu taj model koristiti kao skicu za izradu sistema.

Crtež je dobra analogija za Rose model. Kao što za izgradnju kuće postoje crtež koji omogućavaju da više članova tima, koji gradi kuću, imaju poglede iz različite perspektive na te nacrte (vodoinstalateri, električari itd.), tako se modelom u Rose-u omogućava, različitim dijagramima, da članovi tima koji sistem projektuju imaju pogled iz različitih perspektiva (korisnik, dizajneri, menadžeri projekta, inženjeri koji vrše testiranje itd.).

Rational Rose podržava osam različitih tipova UML dijagrama: dijagrame slučajeva upotrebe, dijagrame aktivnosti, dijagrame sekvenci, dijagrame saradnje, dijagrame klasa, dijagrame stanja, dijagrame komponenti i dijagrame razmeštaja.

UML dijagrami prikazuju sistem iz više uglova i za taj zadatak se koriste:

dijagram slučajeva upotrebe (Use-Case Diagram) je grafička ilustracija funkcionalnosti sistema (statički pristup) koja prikazuje učesnike (Actor) i njihove veze sa slučajevima upotrebe (use cases) tj. to je korisnički pogled funkcionisanja sistema (šta sistem radi a ne kako sistem funkcioniše). Slučajevi upotrebe i učesnici su specijalne vrste klasa i njihovih relacija;

dijagram klasa (Class Diagram) prikazuje skup klasa, interfejsa i saradnja i njihovih relacija. Dijagram klasa je statički struktura klasa u sistemu koje izmedju sebe uspostavljaju relacija tipa asocijacija (veza sa svakom drugom), zavisnosti (jedna klasa zavisi/koristi se od druge klase), specijalizacije (jedna klasa je specijalizacija druge klase, i paketa (grupisanje u jednu jedinicu tj. paketi);

dijagram sekvenci (Sequence Diagram) opisuje vreme trajanja poruke i način su na koji objekti u sistemu medjusobno komuniciraju, ostvarujući očekivano ponašanje. Dakle, prikazuje se vremenska komponenta i poruke koje se prosledjuju izmedju objekata u cilju izvršenja posmatrane operacije. Objekti su imenovane ili neimenovane instance klasa, ali mogu da budu i instance drugih stvari kao što su saradnja, komponente ili čvorovi. Dijagram sekvenci je grafička ilustracija dinamičke interakcije gde objekti

komuniciraju preko sekvenci poruka tj. prikazuje dinamičku saradnju izmedju objekata u vremenu. Dijagram sekvenci se može prevesti u kolaboracioi i obrnuto;

dijagram saradnje (Collaboration Diagram) definiše komunikaciju, pa i veze izmedju objekata neophodne za ostvarivanje posmatrane komunikacije. Dijagram saradnje pored objekata i veza prikazuje i poruke koje objekti medjusobno prosledjuju, ostvarujući na taj način očekivano ponašanje. Dijagram saradnje opisuje strukturnu organizaciju objekata koji šalju i prikazuju poruke.

dijagram promene stanja (State Diagram) je konačni automat koji sadrži stanje, prelaze, dogadjaje, i aktivnosti. Dijagram promene stanja je dinamički dijagram koji prikazuje sekvencu stanja kroz koje objekat prolazi tokom vremena (tokom životnog veka), a kao reakcija na spoljne ili unitrašnje pobude (vezan za samo jedan objekat i odredjenu operaciju unutar njega za odredjenu klasu); Opis stanja obuhvata aktivnosti koje se izvršavaju u pojedinim stanjima, akcije koje se izvršavaju pri prelasku iz jednog stanja u drugo, kao i poruke koje uslovljavaju promenu stanja posmatranog objekta. Kreiranjem dijagrama promene stanja prikazuju se reakcije sistema izazvane dogadjajima. Dijagram promene stanja se može prevesti u dijagram aktivnosti koji se fokusira na tok kontroje (i obrnuto).

dijagram aktivnosti (Activity Diagram) prikazuje sekvencijalan tok aktivnosti, a sastoji se od: stanja, akcija i prelaza i služi za prikaz dinamičkog odvijanja poslovnih proces; Dijagram aktivnosti opisuje aktivnosti koje se izvršavaju u okviru jedne operacije tj. predstavljaju sam algoritam operacija. Dijagram aktivnosti je specijalna vrsta dijagrama promene stanja kojim se prikazuje tok od aktivnosti do aktivnosti kroz sistem.

dijagram komponenti (Component Diagram) prikazuje fizičku strukturu korisničkog softvera (izvorni kod, binarni i exe kod). Dijagram komponenti prikazuje organizaciju i zavisnost izmedju skupa komponenti. Dijagram komponenti je povezan sa dijagramom klasa zbog toga što svaka komponenta ukazuje na jednu ili više klasa, interfejsa ili saradnja;

dijagram razvoja (Deployment Diagram) prikazuje konfiguraciju vremena procesiranja i komponente u njemu. Čvor dijagrama razvoja obuhvata jednu ili više komponenti. Dijagram razvoja se koristi za prikaz statičkog pogleda na arhitekturu tj. na fizičku strukturu hardvera i softvera.

Izrada šeme baze podataka u RationalRous-u

Modeliranje baze podataka preporučuje se da se izvodi paraleno sa objektnim modeliranjem. Kako smo u (2) izmodelirali i izgenerisali baze podataka korišćenjem CASE alata ERwin postupkom reverznim inženjeringom u RationalRose modeliraćemo model podataka u istom.

Da biste počeli proces, potrebno je da izaberete Tools > Data Modeler > Reverse Engineer. Nakon toga bićete upitani da li se radi reverzni inženjering iz DDL izvornika ili iz baze podataka. Ako izaberete DDL, bićete upitani za DBMS koji želite da koristite i ime fajla. Ako izaberete bazu podataka, bićete upitani za informacije o konekciji prema bazi podataka.

Sledeće, izaberite stavku(e) koje želite da provučete kroz reverzni inženjering. Tabele i

ograničenja će biti uvek ubačeni, ali možete da odaberete i druge elemente. Kada završite, kliknite Next. Rational Rose će automatski da kreira šemu u Logical view. Unutar nje biće sve tabele, ograničenja i drugi elementi modela iz Vaše baze podataka. Ako Rational Rose detektuje bilo kakav problem tokom reverznog inženjeringa, upisaće taj problem u prozor Log. Rose će takođe kreirati novu komponentu u Componenet view. Komponenta će reprezentovati bazu podataka.

Kada to radite, Rational Rose će kreirati komponente baze podataka u Component view i tabele i druge elemente baze podataka u šemi u Logical view. Kada jednom izvršite reverzni inženjering baze podataka, možete generisati objektni model ili izvršiti sinhronizaciju koristeći metode koje su ranije objašnjene.

Generisanje objektnog modela iz modela podataka

Jedna nova osobina Rational Rose je sposobnost da automatski generiše objektni model iz modela podataka. Ova osobina je posebno praktična kada radite na projektu koji dobijate reverznim inženjeringom iz neke već postojeće baze podataka ili aplikacije. Naravno, ova osobina je korisna i za druge tipove projekata. Bilo kada, kada želite da proverite konzistentnost objektog modela i modela podataka, ili želite da iskoristite mogućnost reverznog inženjeringa. Ova osobina je veoma korisna i od velike pomoći.

Nemaju sve konstrukcije u modelu podataka značenje u objektnom modelu. Indeksi, procedure smeštanja i drugi elementi baze podataka ne mapiraju se u objektni model. Na sledešoj slici prikazan je spisak elemenata modela podataka i njima odgovarajućih elemenata objektnog modela.

Element modela podataka	**Element objektnog modela**
Šema	Paket
Tabela	Klasa
Kolona	Atribut
Tabela sa samo primarnim i spoljnim ključevima	Veza više prema više
Tabela bez primarnih i spoljnih ključeva	Veza više prema više sa asocijativnom klasom
Identifikujuća veza	Kompozitna agregacija
Neidentifikujuća veza	Asocijacija
Kardinalnost	Kardinalnost
Indeks	nema
Baza podataka	nema
Ograničenje	nema
Domen	nema

Kreiranje objektnog modela iz modela podataka vrši se na sledeći način:

1. Desni-klik na šemu i izaberite Data Modeler > Transorfm to Object Model.

2. Unesite ciljno ime paketa. Ciljni paket je ime paketa koji će da bude kreiran u Logical view kako bi sačuvao nove objekte.

3. Unesite prefiks. Prefiks će da bude dodat na imena svih tabela od kojih nastaju klase u objektnom modelu.

4. Izaberite Include Primary Key polje potvrde da bi kreirali atribute za kolone primarnih ključeva kao i za druge kolone. Ako nije selektovano, Rose će generisati atribute samo za kolone koje nisu primarni ključevi.

Kreiranje modela podataka iz objektnog modela

Kada se generiše model podataka, Rational Rose će tražiti klase sa perzistentim atributima. Možete da podesite da klasa bude Persistent ili Transient u standardnom prozoru specifikacije klase na kartici Detail. Ako želite da generišete tabelu za klasu, podesite je na Persistent.

Paket sa klasama postaće šema u modelu podataka. Ako već postoji šema sa istim imenom, Rose će dodati nove klase kao tabele u šemi. Pri tome, neće se promeniti tabele u šemi ako se klase u objektnom modelu promene. U stvari, promene će biti zabeležene u log tako da ih možete primeniti ako želite.

U sledećoj tabeli prikazan je spisak elemenata objektnog modela i njima odgovarajućih elemenata modela podataka.

Element objektnog modela	**Element modela podataka**
Paket	Šema
Perzistentna klasa	Tabela
Atribut	Kolona
Operacija	nema
Veza više prema više	Međutabela
Kompozitna agregacija	Identifikujuća veza
Asocijacija	Neidentifikujuća veza
Kardinalnost	Kardinalnost
Asocijativna klasa	Međutabela

Kreiranje modela podataka iz objektnog modela vrši se na sledeći način:

1. Kreirate bazu podataka u Component view.

2. Desni-klik na bilo koji atribut u klasama za koji želite da postane primarni ključ u generisanoj tabeli. Izaberite Data Modeler Part of Object Identity. Ako ne izaberete primarne ključeve Rational Rose će kreirati primarni ključ kao kolonu sa imenom "<table name>ID>".

3. Desni-klik na paket u Logical view i izaberite Data Modeler > Transform to Data Model.

4. Unesite ciljno ime šeme koje predstavlja ime šeme koja će biti kreirana da bi sadržala nove

elemente podataka.

5. Unesite ciljnu bazu podataka koja predstavlja naziv postojeće baze podataka u Component pogledu.

6. Unesite prefiks koji će da se doda svakoj klasi koja kreira neku tabelu u bazi podataka.

7. Izaberite Create Indexes for Foreign Keys polje potvrde da se automatski kreiraju indeksi za prenešene ključeve.

Generisanje baze podataka iz modela podataka

Bilo kad tokom procesa projektovanja, možete da generišete bazu podataka ili DDL izvornik (script) iz modela podataka. Rational Rose Vam da izaberete jednostavno generisanje DDL-a, ili pokretanje DDL-a da bi kreirali bazu podataka.

Rational Rose ima pomoć (wizards) koji Vas vode kroz sam proces kreiranja baze podataka. Da biste počeli, desni-klik na šemu koju želite da generišete i izaberite opciju Data Modeler > Forward Engineering. Nakon toga izaberite elemente koje želite da generišete:

Sledeći korak, jeste unošenje imena DDL fajla koji kreirate. Ako želite da kreirate DDL, ali ne i da je pokrenete, kliknite Next. U suprotnom izaberite Execute polje potvrde. Unesite informacije o konekciji za DBMS i pritisnite test Connection dugme da bi proverili da li su unešeni parametri ispravni.

Pritisnite Finish da bi završili proces. Rational Rose će generisati DDL i opciono ga pokrenuti. Ako se pojavi neka greška Rational Rose će je upisati u prozor Log.

Sve tabele, kolone i relacije u šemi biće generisane u DDL ili bazu podataka. Naredni primer prikazuje tabelu u Rational Rose modelu.

3. Poslovi fakturisanja

Uvod

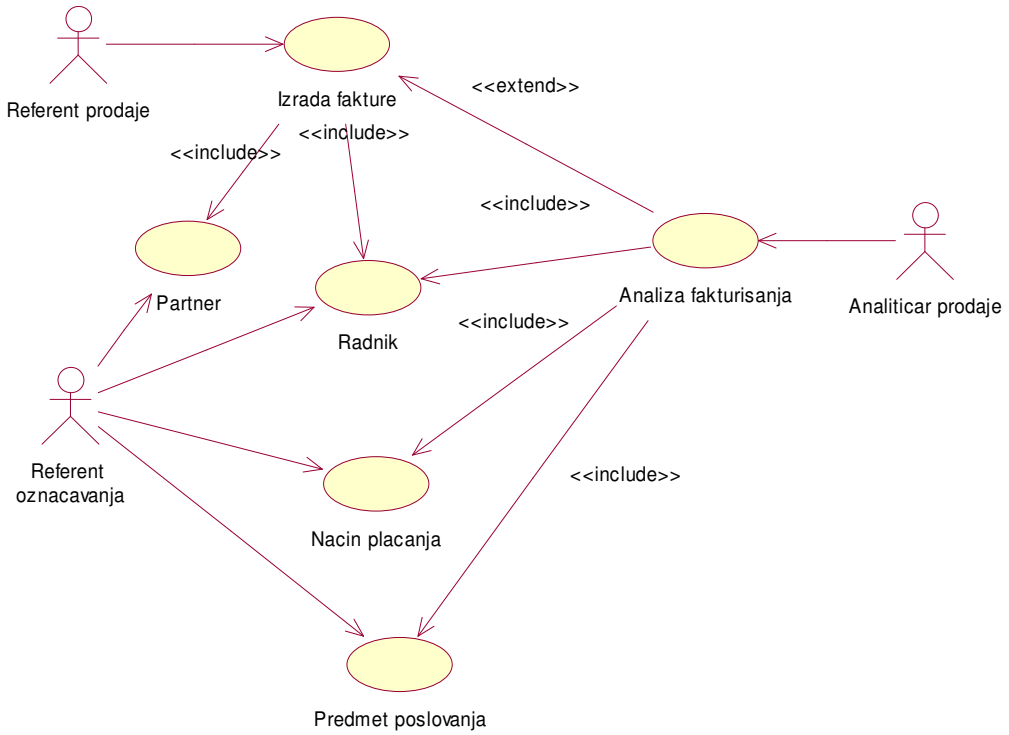
Na primeru procesa fakturisanja, prikazaće se faze objektnog modeliranja informacionog sistema definisanog u (1). Ovaj primer predstavlja svojevrsan vodič kroz definisanje osnovnih elemenata projekta razvoja informacionog sistema korišćenjem objektno orijentisanih CASE alata (Rational Rous). Osnove vezane za poslove fakturisanja date su u (2). Ovo je primer sa minimalnim zahtevima za izradu seminarskog rada za predmet Projektovanje informacionih sistema

Potrebno je uraditi sledeće:

- Izrada dijagrama slučajeva upotrebe
- Izrada dijagrama aktivnosti
- Izrada konceptualnog modela
- Izrada dijagrama sekvenci
- Izrada potpunih dijagrama klasa
- Izrada šeme baze podataka

Izrada dijagrama slučajeva upotrebe za poslove fakturisanja

U daljem radu koristićemo CASE alata RationalRose koji u potpunosti podržava UML notaciju za modelovanje dijagrama slučaja korišćenja. Model se ne sastoji samo od dijagrama već i od detaljnog opisa slučaja upotrebe. Slede dijagrami slučajeva upotrebe i tekstualni opis slučajeva upotrebe fakturisanje.



Slika 3.1. Dijagram slučajeva upotrebe za poslove fakturisanja

Slučajevi upotrebe: Partner, Način plaćanja, Radnik i Predmet poslovanja

Kratak opis: Vođenje podataka o ažurnosti sifarnika Partnera, Način plaćanja, Radnika i Predmet poslovanja.

Učesnici: Referent označavanja i Referent prodaje

Uslovi koji moraju biti zadovoljeni pre izvršavanja: Uvek ažurni šifarnici.

Opis: Svaka aplikacija mora imati ažurne šifarnika. Tu se vode podaci o partnerima, načinu plaćanja, radnicima i predmetima poslovanja.

Izuzeci: Nema.

Uslovi koji moraju biti zadovoljeni posle izvršavanja: Uredna i tačna evidencija zapisa.

Slučaj upotrebe: Izrada fakture

Kratak opis: Potreba za slanjem robe izaziva izradu fakture.

Učesnici: Referent prodaje

Uslovi koji moraju biti zadovoljeni pre izvršavanja: Moraju biti azurni sifarnici.

Opis: Referent prodaje evidentira faktire koje se fakturisu, proveri tačnost podataka i štampa potrebnu fakturu .

Izuzeci: (štampač ne štampa fakturu) proveriti da li u štampaču ima papira;

Uslovi koji Uslovi koji moraju biti zadovoljeni posle izvršavanja: Poslata informacija analitičaru prodaje.

Slučaj upotrebe: Analiza fakturisanja

Kratak opis: Analiza fakturisanja na osnovi poslatih faktura.

Učesnici: Analiticar prodaje

Uslovi koji moraju biti zadovoljeni pre izvršavanja: Evidentirana naplata potrazivanja po fakturi.

Opis: Analiza faktura je osnova za pračenje poslovanja .

Izuzeci: (naslov ne postoji) pogledati da li su dobro uneti podaci za pretragu

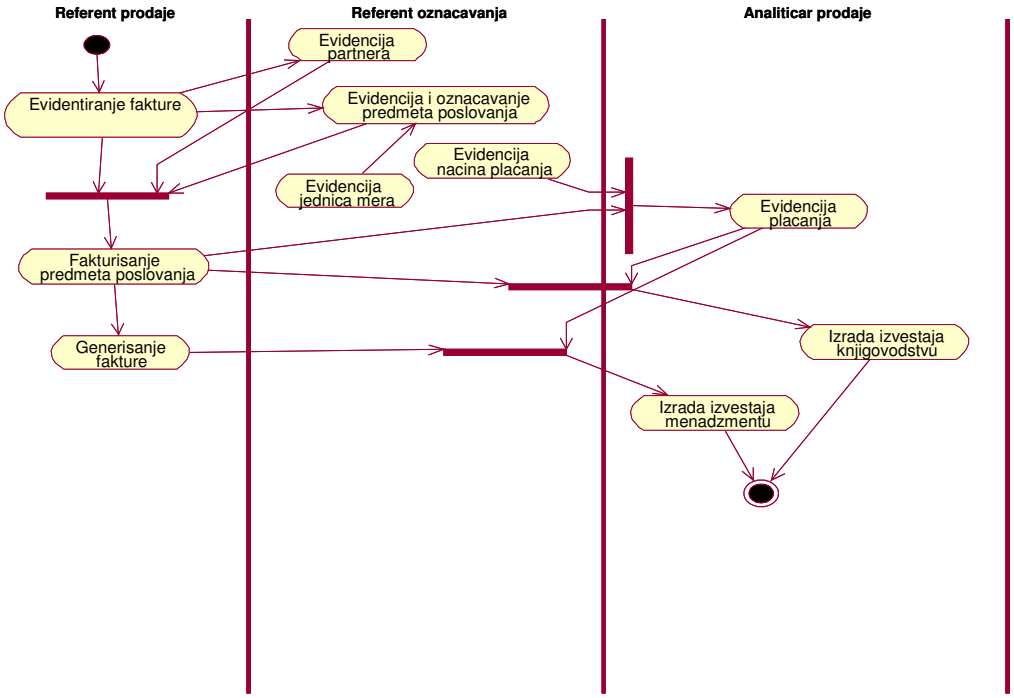
Izrada dijagrami aktivnosti za poslove fakturisanja

Svaki dijagram aktivnosti, počinje startnom aktivnošću koja se predstavlja ispunjenim crnim krugom.

Dijagram aktivnosti definisan je plivačkim stazama, stanjem i tranzicijom.

Plivačke staze ili procesori su referent prodaje, referent označavanja i analitičar prodaje i one specificiraju odgovornosti za delove celokupne aktivnosti i nemaju neku duboku semantiku. *Stanja* pripadaju stazama, a *tranzicije* mogu prelaziti iz jedne staze u drugu.

Na sledećoj slici prikazan je dijagram aktivnosti za poslove fakturisanja.

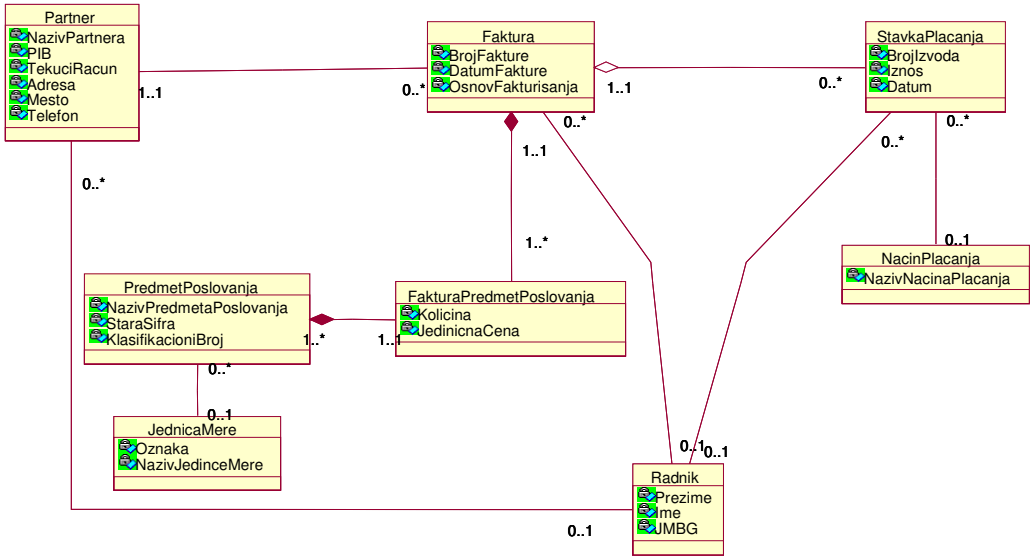


Slika 3.2. Dijagram aktivnosti za proces fakturisanje

Izrada konceptualnog modela za poslove fakturisanja

Suština konceptualnog modela je skeniranje realnog sistema u potrazi za klasama objekata i značenjem njihovih međusobnih veza. Može se reći da svaka koncept u konceptualnom modelu opisan je dodeljenim atributima i nekim elementarnim operacijama.

Na sledećoj slici prikazan je konceptualni model za proces fakturisanja.



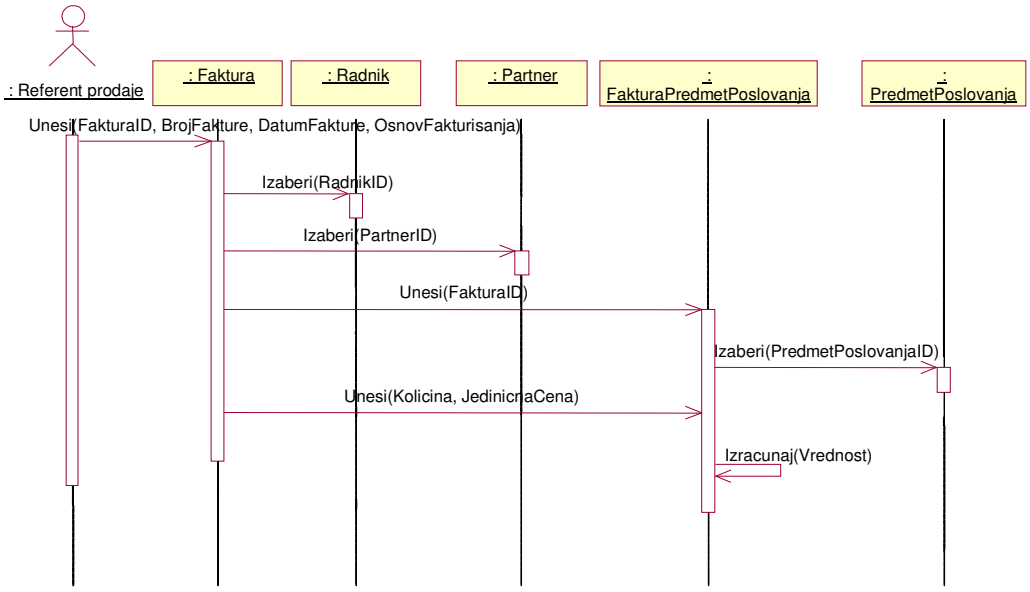
Slika 3.3. Konceptualni model za proces fakturisanja

Iidentifikovani su sledeći koncepti u konkretnom problemu realnog sistema: Sifarnici (Partner, Radnik, JednicaMere, NacinPlacanja i PredmetPoslovanja), Faktura, StavkaPlacanja i FakturaPredmetPoslovanja.

Izrada dijagrama sekvenci za poslove fakturisanja

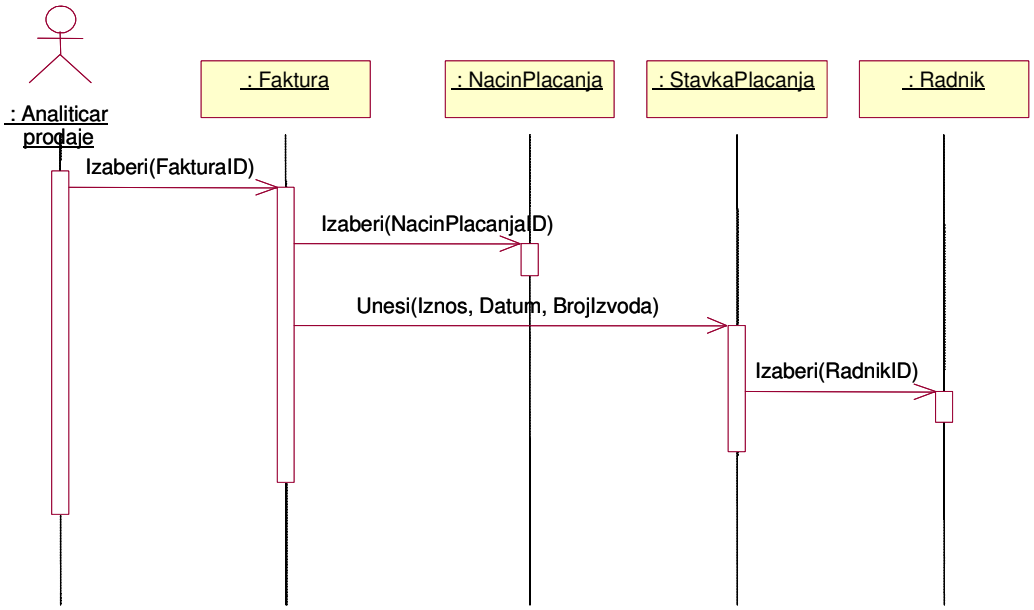
Kod sekvencijalnih dijagrama naglasak je na vremenski redosled odvijanja poruka između objekata različitih klasa. Dakle, sekvencijalni dijagrami se crtaju na vremenskoj osi, i predstavljaju specifikaciju vremenski zahteva u pogledu šta sistem treba da radi u realnom vremenu. Vremenskim redosledom poruka u sekvencijalnom dijagramu opisaće se logika odvijanja poslova fakturisanja za slučajeve upotrebe izrada fakture i analiza fakturisanja. Dakle, videće se šta podsistem treba da radi u realnom vremenu da bi obavio svoju funkciju predstavljenu slučajom upotrebe.

Na sledećoj slici prikazan je dijagram sekvenci za proces izrada fakture.



Slika 3.4. Dijagram sekvenci za proces izrada fakture

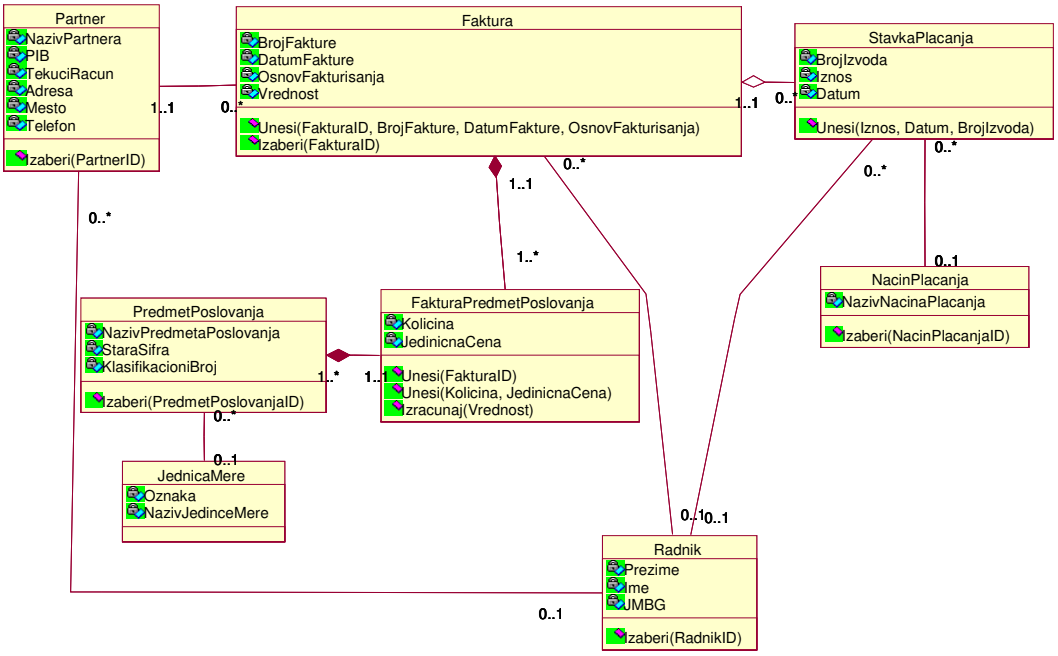
Na sledećoj slici prikazan je dijagram sekvenci za proces analiza fakturisanja.



Slika 3.5. Dijagram sekvenci za proces analiza fakturisanja

Izrada dijagrama klasa za poslove fakturisanja

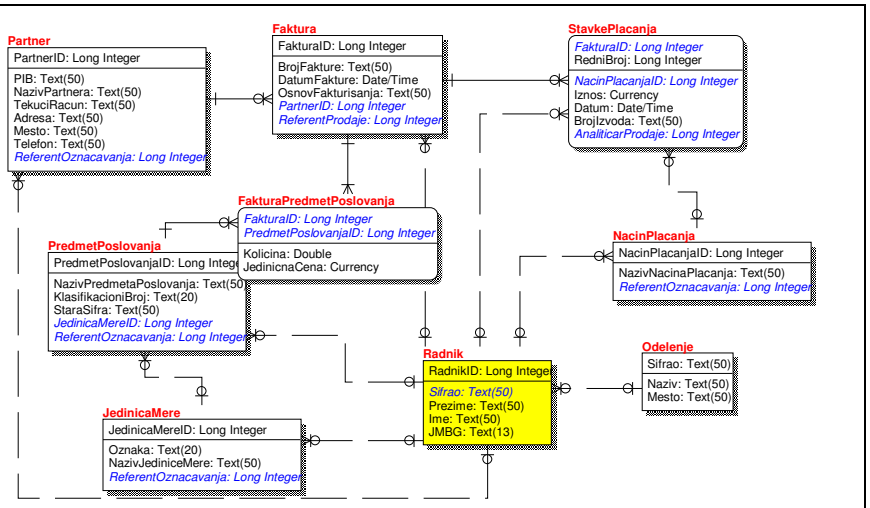
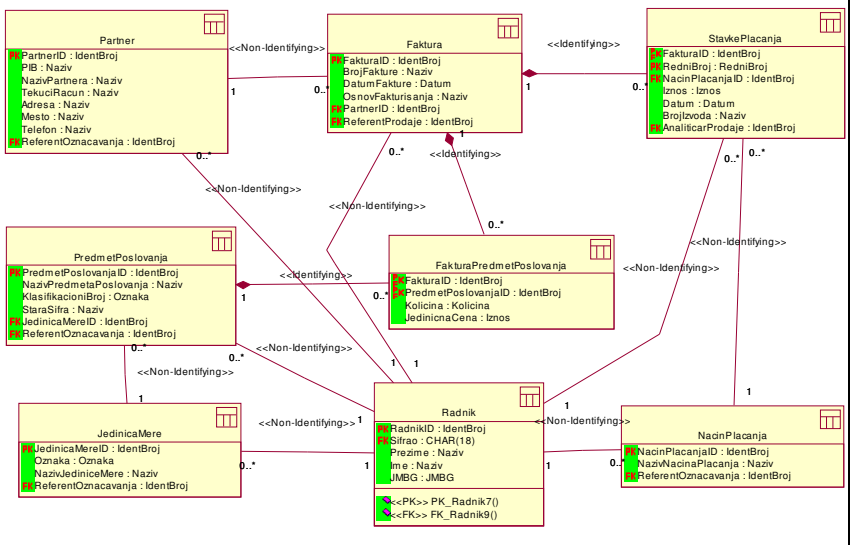
Dijagram klase se sastoji iz klasa i veza između njih. Naziva se još i dijagram statičke strukture. Klasa predstavlja složeni tip, kolekciju, struturu koja se sastoji od više atributa (podata članova) i operacija (metoda, funkcija članica). Na sledećoj slici prikazan je potpuni dijagram klasa za poslove fakturisanja.



Slika 3.6. Dijagram klasa za poslove fakturisanja

Izrada šeme baze podataka za poslove fakturisanja

Na sledećoj slici prikazan je fizički model podataka definisan u Erwinu i i odgovarajući fizički model podataka u RationalRous-u dobijen postupkom reverznog inzinjeringa iz skript fajla generisanog u Erwin-u.

Fizicki model podataka u ERwin-u(2)	
Model podataka u Rational Rous-u	

Slika 3.7. Model podataka

Zaključak

Na osnovu definisanih dijagrama pristupa se izradi dva sledeća koraka. U prvom koraku prelazi se na definisanje server strane generisanjem baze podataka kao što je pokazano za ovaj primer u(2). Drugi korak podrazumeva izradu klijent strane u zavisnosti od potreba korisnika ili Web programiranjem ili korišćenjem Visual Basica ili na kraju i MS Access-om.

Literatura

1 Veljović A. Objektno modeliranje informacionih sistema, Fakultet za poslovne studije, MEGATREND Univerzitet, 2003. godina

2. Veljović A. Praktikum iz analize informacionih sistema, Fakultet za poslovne studije, MEGATREND Univerzitet, 2005. godina

4. Poslovi praćenja ispita

Uvod

Na primeru procesa praćenja ispita, prikazaće se faze objektnog modeliranja informacionog sistema definisanog u (1). Ovaj primer predstavlja svojevrsan vodič kroz definisanje osnovnih elemenata projekta razvoja informacionog sistema korišćenjem objektno orijentisanih CASE alata (RationalRous). Osnove vezane za poslove praćenja ispita date su u (2). Ovo je primer sa minimalnim zahtevima za izradu seminarskog rada za predmet Projektovanje informacionih sistema i orjentisan slučajevima upotrebe (use case driven).

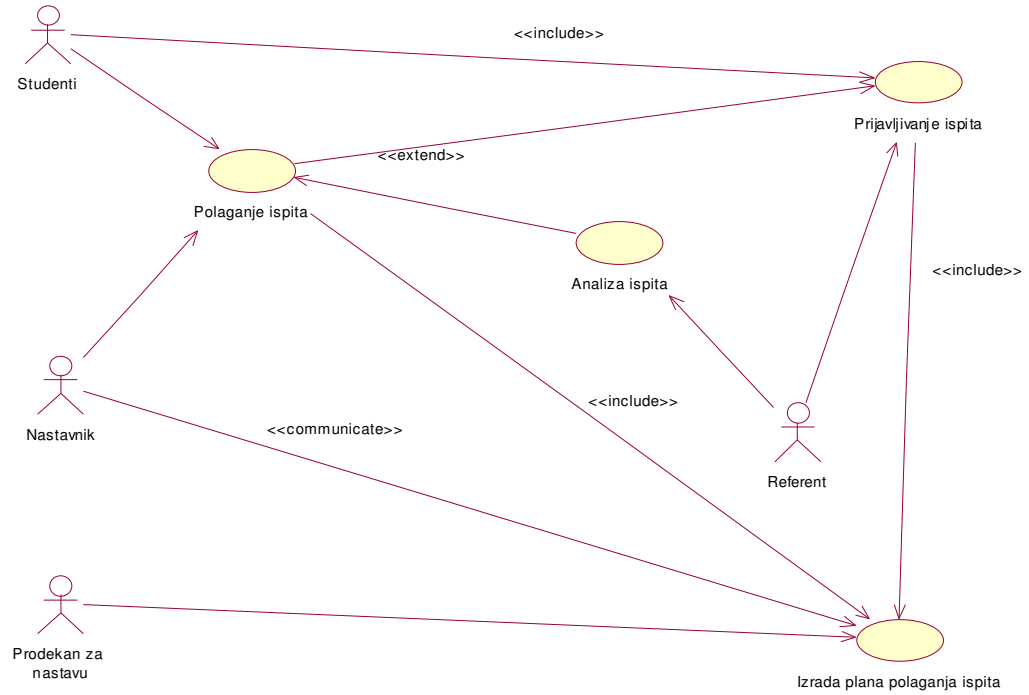
Potrebno je uraditi za svaki slučaj upotrebe u okviru dijagrama slučajeva upotrebe:

- Dijagram aktivnosti
- Konceptualni model
- Dijagram sekvenci
- Dijagram klasa
- Šemu baze podataka

Izrada dijagrama slučajeva upotrebe za poslove praćenja ispita

U daljem radu koristićemo CASE alata RationalRose koji u potpunosti podržava UML notaciju za modelovanje dijagrama slučaja korišćenja. Model se ne sastoji samo od dijagrama već i od detaljnog opisa slučaja upotrebe.

Slede dijagrami slučajeva upotrebe i tekstualni opis slučajeva upotrebe praćenja ispita.



Slika 4.1. Dijagram slučajeva upotrebe za poslove praćenja ispita

Poslovi vezani za slučaj upotrebe Prijavljivanje ispita

Za slučaj upotrebe Prijavljivanje ispita opisuje se slučaj upotrebe i definišu dijagram aktivnosti, dijagram sekvenci i dijagram klasa.

Slučaj upotrebe: Prijavljivanje ispita

Kratak opis: Prijavljivanje ispita ima za osnovu popunjavanje prijave za ispit.

Učesnici: Referent i Student

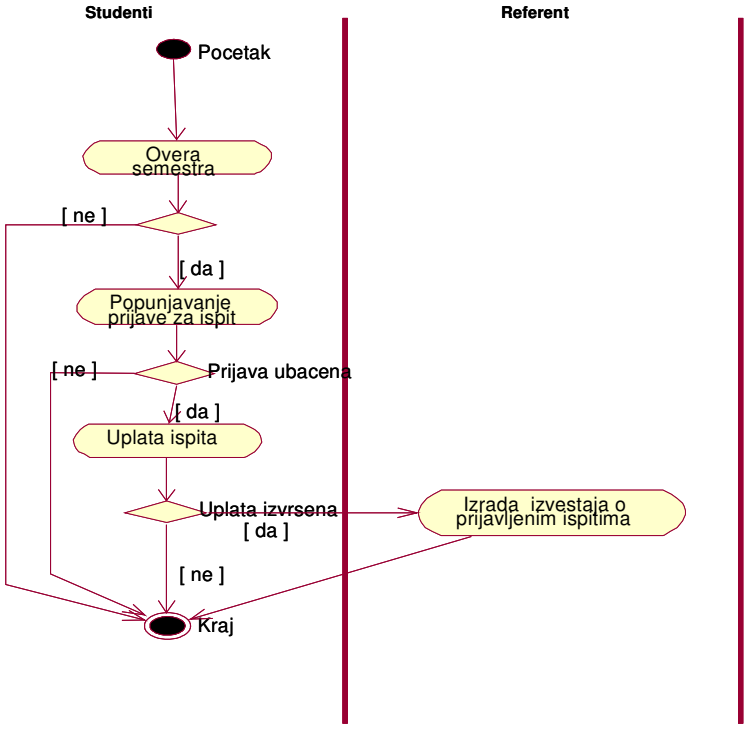
Uslovi koji moraju biti zadovoljeni pre izvršavanja: Moraju biti azurni sifarnici.

Opis: Student prilikom prijavljivanja ispita overava semestar, popunjavanje prijave za ispit i uplaćuje ispit. Referent izradjuje izvestaj o prijavljenim ispitima.
Izuzeci: (štampač ne štampa fakturu) proveriti da li u štampaču ima papira;

Uslovi koji moraju biti zadovoljeni posle izvršavanja: Poslata informacija studentskoj službi u obliku spisak prijavljeni studenta po ispitima.

Dijagram aktivnosti: Prijavljivanje ispita

Dijagram aktivnosti Prijavljivanje ispita definisan je plivačkim stazama, stanjem i tranzicijom. Plivačke staze ili procesori su student i referent. *Stanja* pripadaju stazama, a *tranzicije* mogu prelaziti iz jedne staze u drugu. Na sledećoj slici prikazan je dijagram aktivnosti za poslove Prijavljivanje ispita

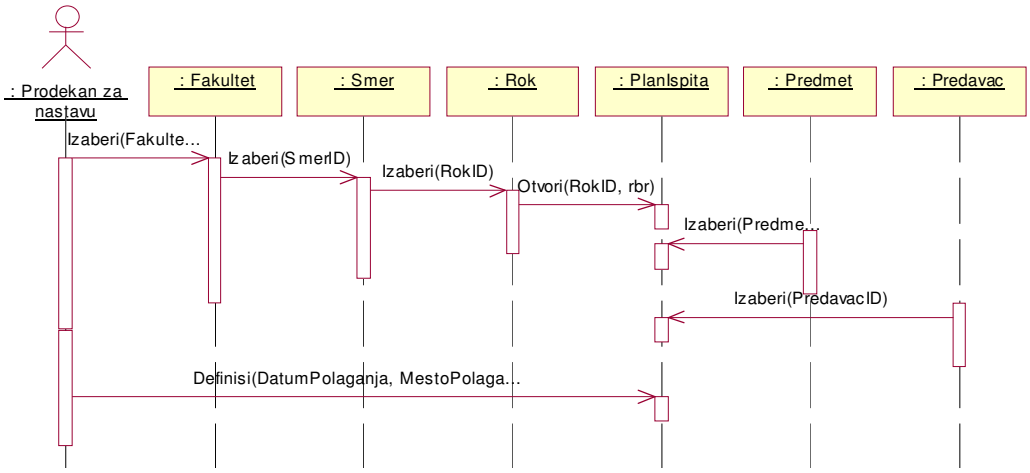


Slika 4.2. Dijagram aktivnosti za slučaj upotrebe Prijavljivanje ispita

Dijagram sekvenci: Prijavljivanje ispita

Prikazaće se šta slučaj upotrebe Prijavljivanje ispita treba da radi u realnom vremenu da bi obavio svoju funkciju. Vremenskim redosledom poruka u sekvencijalnom dijagramu opisaće logiku odvijanja poslova Prijavljivanja ispita za slučaj upotrebe Prijavljivanje ispita.

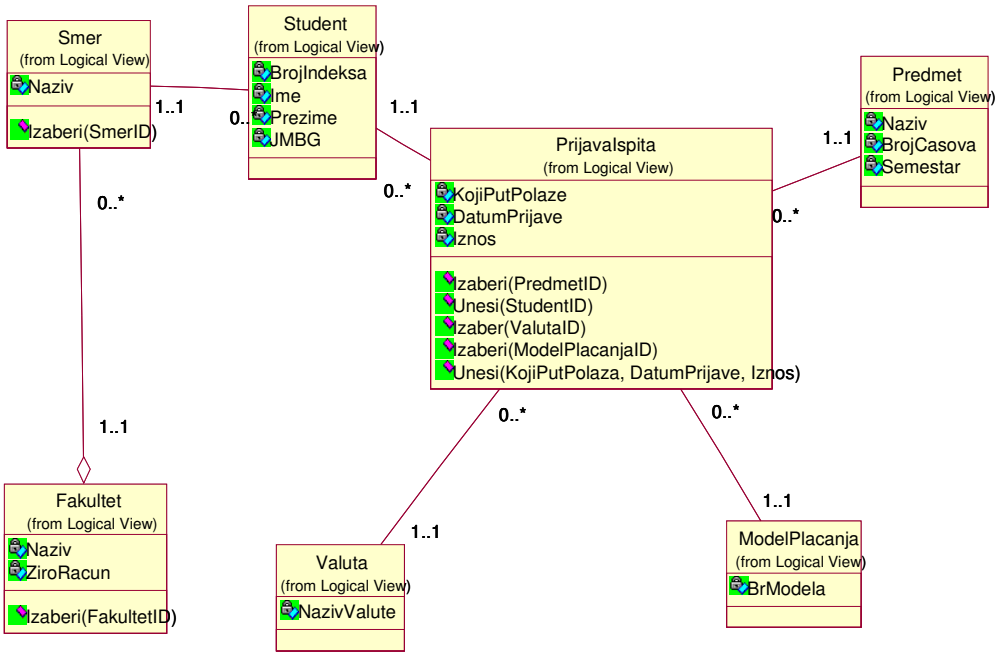
Na sledećoj slici prikazan je dijagram sekvenci za proces Prijavljivanje ispita



Slika 4.3. Dijagram sekvenci za slučaj upotrebe Prijavljivanje ispita

Dijagram klasa: Prijavljivanje ispita

Dijagram klase Prijavljivanje ispita se sastoji iz klasa i veza između njih. Klasa predstavlja složeni tip, kolekciju, strukturu koja se sastoji od više atributa (podata članova) i operacija (metoda, funkcija članica). Na sledećoj slici prikazan je dijagram klasa za poslove Prijavljivanje ispita.



Slika 4.4. Dijagram klasa za slučaj upotrebe prijavljivanje ispita

Poslovi vezani za slučaj upotrebe Izrada plana polaganja ispita

Za slučaj upotrebe Izrada plana polaganja ispita opisuje se slučaj upotrebe i definišu dijagram aktivnosti, dijagram sekvenci i dijagram klasa.

Slučaj upotrebe: Izrada plana polaganja ispita

Kratak opis: Izrada plana polaganja ispita definiše termine polaganja ispita.

Učesnici: Prodekan za nastavu i Nastavnik

Uslovi koji moraju biti zadovoljeni pre izvršavanja: Konsultovani nastavnici i pregled raspoloživih prostorija.

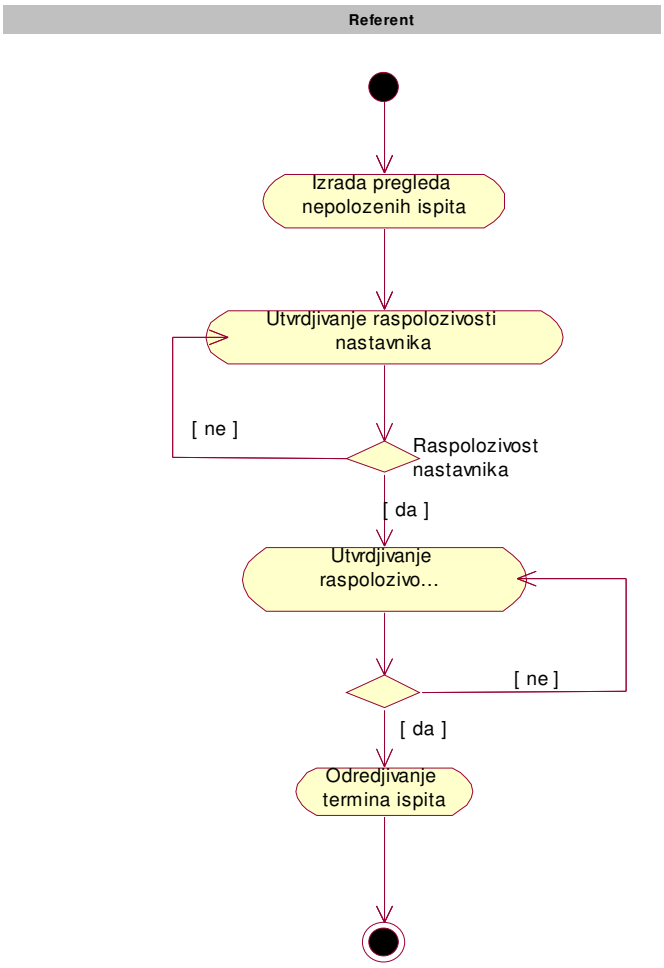
Opis: Izrada plana polaganja ispita počinje sa izradom pregleda nepolozenih ispita, i

utvrdjivanjem raspolozivosti prostorija i utvrdjivanjem raspolozivosti nastavnika. Na osnovu ovih elemanata odredjuju se termini ispita.
Izuzeci: Nema.

Uslovi koji moraju biti zadovoljeni posle izvršavanja: Uredna i tačna evidencija zapisa.

Dijagram aktivnosti: Izrada plana polaganja ispita

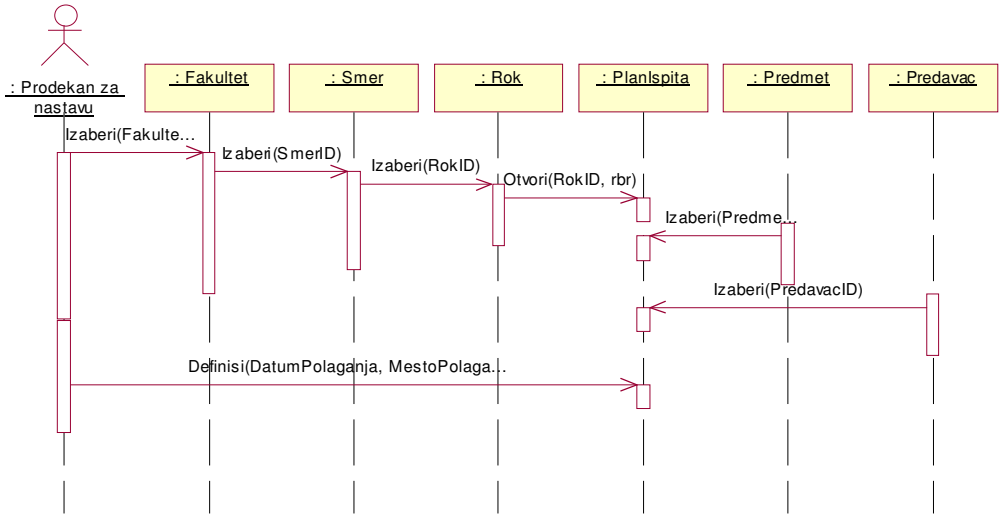
Dijagram aktivnosti Izrada plana polaganja ispita definisan je plivačkim stazama, stanjem i tranzicijom. Plivačke staze ili procesori su Prodekan za nastavu i Nastavnik. *Stanja* pripadaju stazama, a *tranzicije* mogu prelaziti iz jedne staze u drugu. Na sledećoj slici prikazan je dijagram aktivnosti za poslove Izrada plana polaganja ispita



Slika 4.5. Dijagram aktivnosti za slučaj upotrebe Izrada plana polaganja ispita

Dijagram sekvenci: Izrada plana polaganja ispita

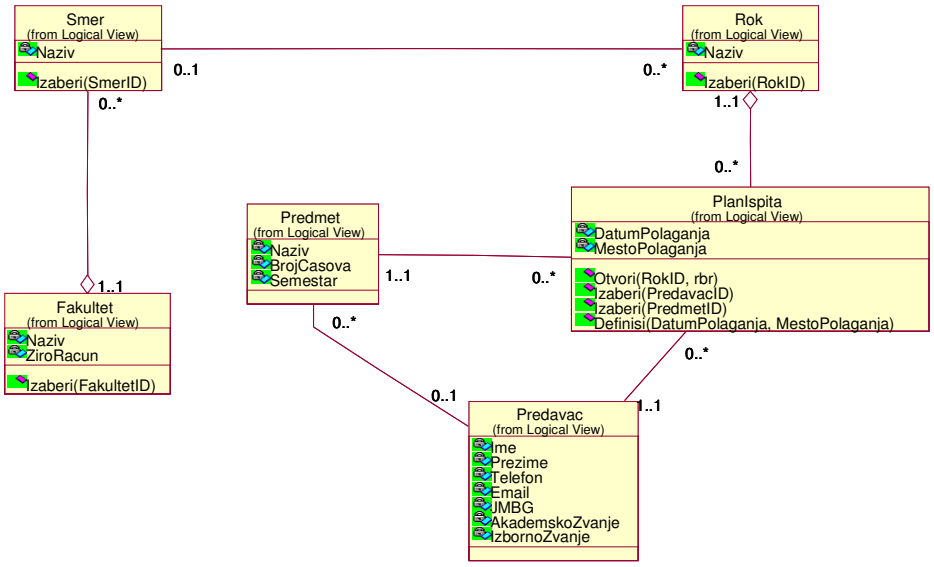
Prikazaće se šta slučaj upotrebe Izrada plana polaganja ispita treba da radi u realnom vremenu da bi obavio svoju funkciju. Vremenskim redosledom poruka u sekvencijalnom dijagramu opisaće logiku odvijanja poslova Izrada plana polaganja ispita za slučaj upotrebe Izrada plana polaganja ispita. Na sledećoj slici prikazan je dijagram sekvenci za proces Izrada plana polaganja ispita



Slika 4.6. Dijagram sekvenci za slučaj upotrebe Izrada plana polaganja ispita

Dijagram klasa: Izrada plana polaganja ispita

Dijagram klase Izrada plana polaganja ispita se sastoji iz klasa i veza između njih. Klasa predstavlja složeni tip, kolekciju, struturu koja se sastoji od više atributa (podata članova) i operacija (metoda, funkcija članica). Na sledećoj slici prikazan je dijagram klasa za poslove Izrada plana polaganja ispita.



Slika 4.7. Dijagram klasa za slučaj upotrebe Izrada plana polaganja ispita

Poslovi vezani za slučaj upotrebe polaganje ispita

Za slučaj upotrebe polaganja ispita opisuje se slučaj upotrebe i definišu dijagram aktivnosti, dijagram sekvenci, dijagram klasa i dijagrami baze podataka.

Slučaj upotrebe: Polaganje ispita

Kratak opis: Izrada zapisnika o polaganju ispita.

Učesnici: Nastavnik, student

Uslovi koji moraju biti zadovoljeni pre izvršavanja: Prijavljen ispit.

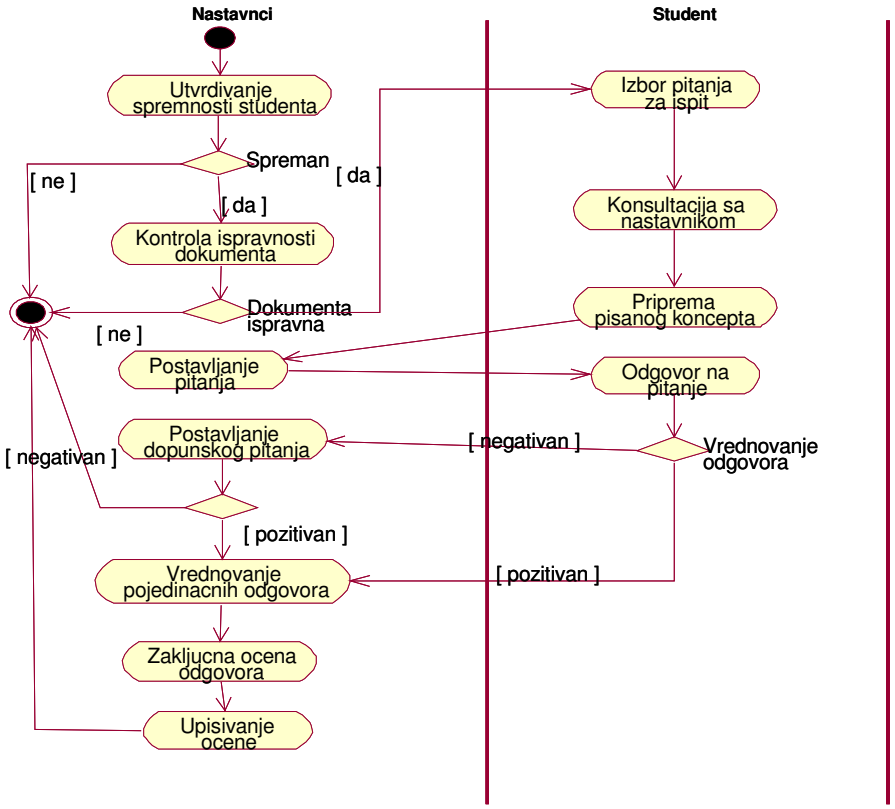
Opis: Prvi korak u polaganju ispita je pristupanje ispitu gde nastavnik utvrdjuje spremnost studenta i izvodi kontrola ispravnosti dokumenata. Nastavnik kontroliše spisak prijavljenih studenata, uplatnice, prijavnice za ispit i indeks. Student se neposredno priprema za ispit izborom pitanja za ispit, konsultacijom sa nastavnikom i pripremom pisanog koncepkta za ispit. Nastavnik u sledećem koraku izvodi ispitivanje postavljanjem pitanja, dobijanjem odgovara na pitanja i postavljanjem dopunskih pitanja. Na kraju nastavnik izvodi postupa ocenjivanja kroz vrednovanje pojedinacnog odgovora, zaključna ocena odgovora i upisivanjem ocena.
Izuzeci: Nema

Uslovi koji moraju biti zadovoljeni posle izvršavanja: Uredna i tačan zapisnik o polaganju ispita.

Dijagram aktivnosti: Polaganje ispita

Dijagram aktivnosti Polaganje ispita definisan je plivačkim stazama, stanjem i tranzicijom.

Plivačke staze ili procesori su Nastavnik, student. *Stanja* pripadaju stazama, a *tranzicije* mogu prelaziti iz jedne staze u drugu. Na sledećoj slici prikazan je dijagram aktivnosti za poslove Polaganje ispita

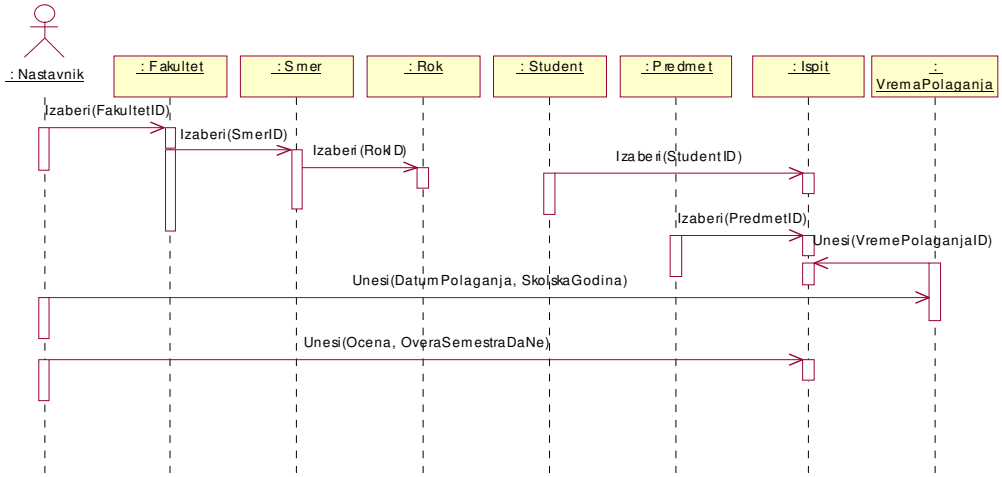


Slika 4.8. Dijagram aktivnosti za slučaj upotrebe Polaganje ispita

Dijagram sekvenci: Polaganje ispita

Prikazaće se šta slučaj upotrebe Polaganje ispita treba da radi u realnom vremenu da bi obavio svoju funkciju. Vremenskim redosledom poruka u sekvencijalnom dijagramu opisaće logiku odvijanja poslova Polaganje ispita za slučaj upotrebe Izrada plana polaganja ispita.

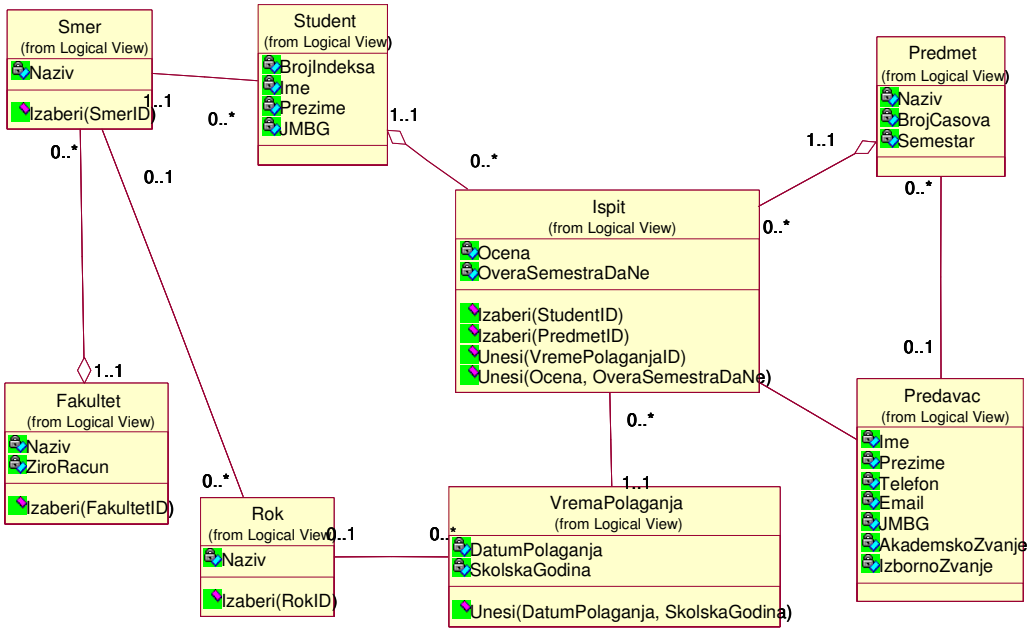
Na sledećoj slici prikazan je dijagram sekvenci za proces Polaganje ispita



Slika 4.9. Dijagram sekvenci za slučaj upotrebe Polaganje ispita

Dijagram klasa: Polaganje ispita

Dijagram klase Polaganje ispita se sastoji iz klasa i veza između njih. Klasa predstavlja složeni tip, kolekciju, struturu koja se sastoji od više atributa (podata članova) i operacija (metoda, funkcija članica). Na sledećoj slici prikazan je dijagram klasa za poslove Polaganje ispita.



Slika 4.10. Dijagram klasa za slučaj upotrebe Polaganje ispita

Poslovi vezani za slučaj upotrebe Analiza ispita

Za slučaj upotrebe Analizu ispita opisuje se slučaj upotrebe i definišu dijagram aktivnosti, dijagram sekvenci i dijagram klasa.

Slučaj upotrebe: Analiza ispita

Kratak opis: Analiza ispita ima za osnovu zapisnik o polaganju ispita.

Učesnici: Referent

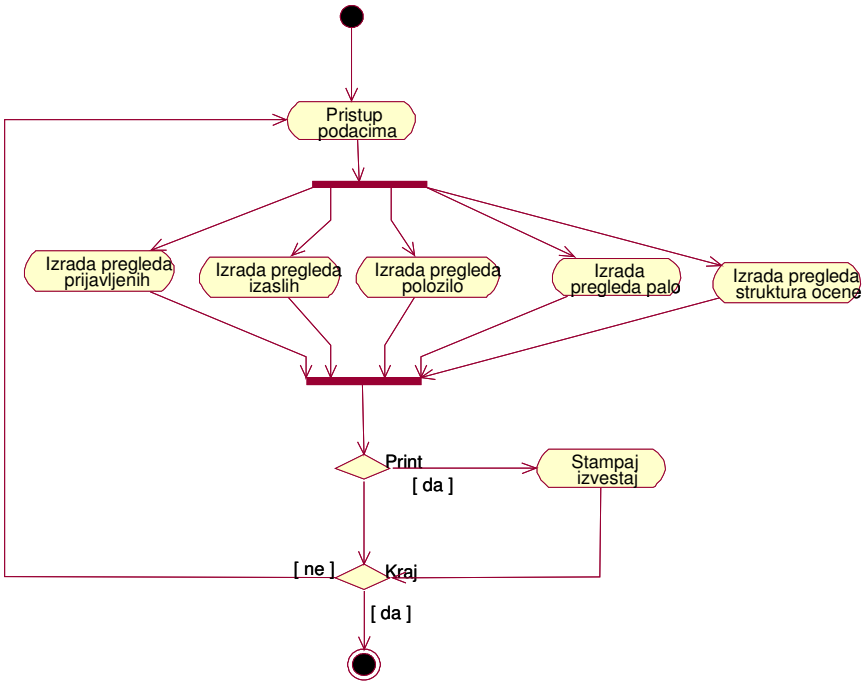
Uslovi koji moraju biti zadovoljeni pre izvršavanja: Moraju biti azurni sifarnici.

Opis: Referent izvodi analizu ispita izradom pregleda struktura ocene studenata koji su prijavili, izasli, polozili ili pali.
Izuzeci: (štampač ne štampa fakturu) proveriti da li u štampaču ima papira;

Uslovi koji moraju biti zadovoljeni posle izvršavanja: Poslata informacija studentskoj službi u obliku spisak prijavljeni studenta po ispitima.

Dijagram aktivnosti: Analiza ispita

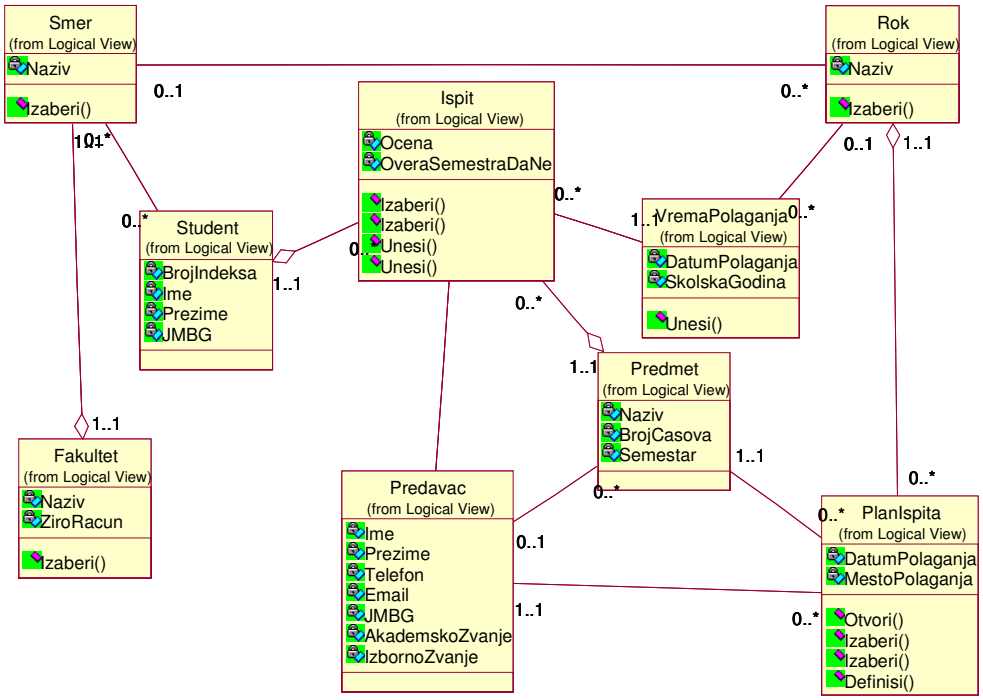
Dijagram aktivnosti Analiza ispita definisan je plivačkim stazama, stanjem i tranzicijom. Plivačka staza ili procesor je referent. *Stanja* pripadaju stazama, a *tranzicije* mogu prelaziti iz jedne staze u drugu. Na sledećoj slici prikazan je dijagram aktivnosti za poslove Analiza ispita.



Slika 4.11. Dijagram aktivnosti za slučaj upotrebe Analiza ispita

Dijagram klasa: Analiza ispita

Dijagram klase Analiza ispita se sastoji iz klasa i veza između njih. Klasa predstavlja složeni tip, kolekciju, struturu koja se sastoji od više atributa (podata članova) i operacija (metoda, funkcija članica). Na sledećoj slici prikazan je dijagram klasa za poslove Analiza ispita.



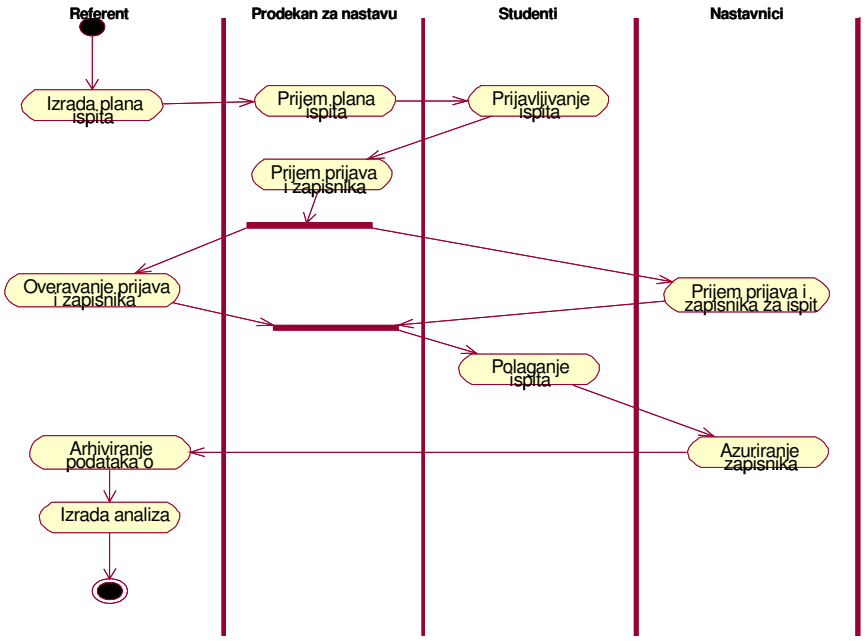
Slika 4.12. Dijagram klasa za slučaj upotrebe Analiza ispita

Izrada dijagrami aktivnosti za poslove praćenja ispita

Svaki dijagram aktivnosti, počinje startnom aktivnošću koja se predstavlja ispunjenim crnim krugom.

Dijagram aktivnosti definisan je plivačkim stazama, stanjem i tranzicijom.

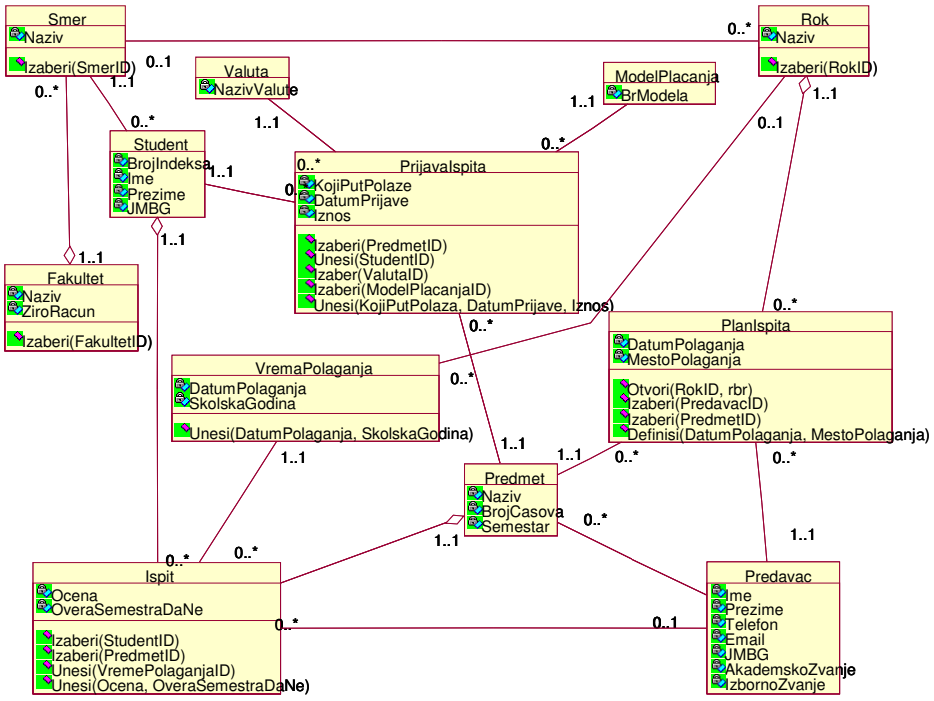
Plivačke staze ili procesori su referent prodaje, referent označavanja i analitičar prodaje i one specificiraju odgovornosti za delove celokupne aktivnosti i nemaju neku duboku semantiku. *Stanja* pripadaju stazama, a *tranzicije* mogu prelaziti iz jedne staze u drugu. Na sledećoj slici prikazan je dijagram aktivnosti za poslove praćenja ispita.



Slika 4.13. Dijagram aktivnosti za poslove praćenja ispita

Izrada dijagrama klasa za poslove praćenja ispita

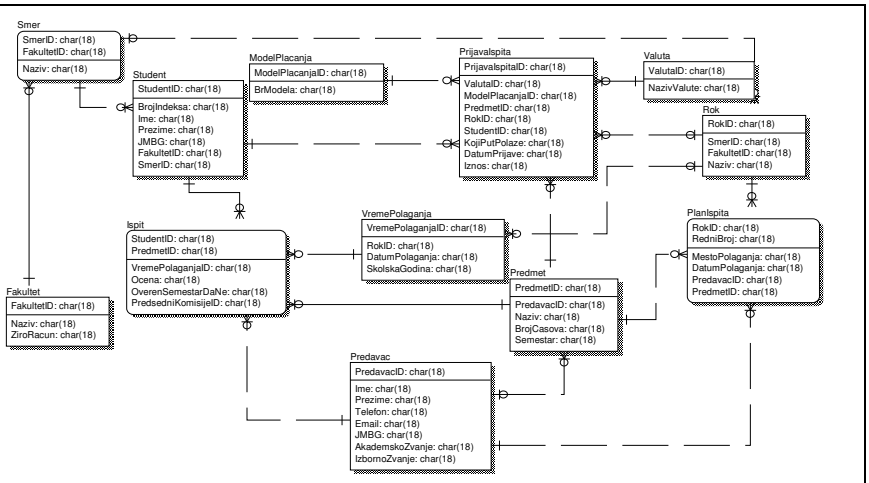
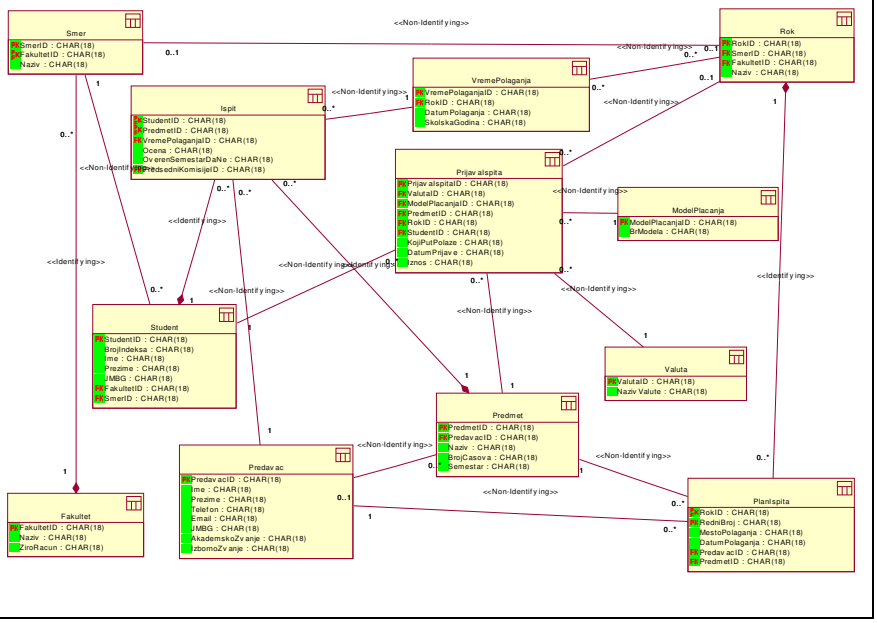
Dijagram klase se sastoji iz klasa i veza između njih. Naziva se još i dijagram statičke strukture. Klasa predstavlja složeni tip, kolekciju, struturu koja se sastoji od više atributa (podata članova) i operacija (metoda, funkcija članica). Na sledećoj slici prikazan je potpuni dijagram klasa za poslove praćenja ispita.



Slika 4.14. Dijagram klasa za poslove praćenja ispita

Izrada šeme baze podataka za poslove praćenja ispita

Kako je definisan model podataka u (2) korišćenjem CASE alata ERwin postupkom reverznim inženjeringom u RationalRose modeliraćemo model podataka u istom.Da biste počeli proces, potrebno je da izaberete Tools > Data Modeler > Reverse Engineer. Na sledećoj slici prikazan je fizički model podataka definisan u Erwinu i i odgovarajući fizički model podataka u RationalRous-u dobijen postupkom reverznog inzinjeringa iz skript fajla generisanog u Erwin-u.

Fizicki model podataka u ERwin-u(2)	
Model podataka u Rational Rous-u	

Slika 4.15. Model podataka

Zaključak

Na osnovu definisanih dijagrama pristupa se izradi dva sledeća koraka. U prvom koraku prelazi se na definisanje server strane generisanjem baze podataka kao što je pokazano za ovaj primer u (2) i (3). Drugi korak podrazumeva izradu klijent strane korišćenjem Web programiranjem.

Literatura

1 Veljović A. Objektno modeliranje informacionih sistema, Fakultet za poslovne studije, MEGATREND Univerzitet, 2003. godina
2. Veljović A. Praktikum iz analize informacionih sistema, Fakultet za poslovne studije, MEGATREND Univerzitet, 2005. godina.
3.Veljović A., Put ka integranolm inforacionom sistemu na primeru Megatrend univerziteta, Megatrend revija, 2005. godina.

5.Poslovi izrade tehnološkog postupka

Uvod

Na primeru poslova izrade tehnološkog postupka, prikazaće se faze objektnog modeliranja informacionog sistema definisanog u (1). Ovaj primer predstavlja svojevrsan vodič kroz definisanje osnovnih elemenata projekta razvoja informacionog sistema korišćenjem objektno orijentisanih CASE alata (Rational Rous). Osnove vezane za poslove izrade tehnološkog postupka date su u (2). Ovo je primer sa minimalnim zahtevima za izradu seminarskog rada za predmet Projektovanje informacionih sistema

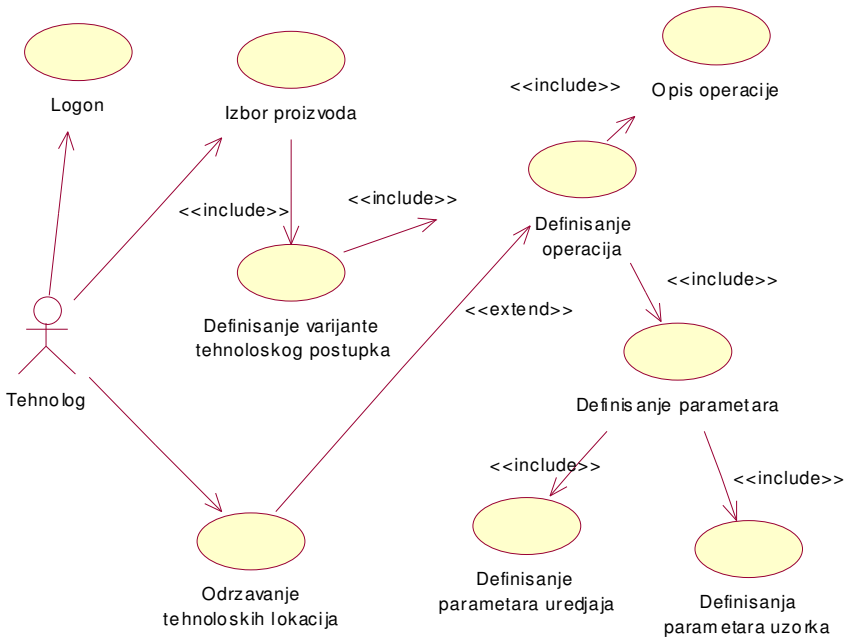
Potrebno je izvesti sledeće aktivnost i to:

- Dijagram slučajeva upotrebe
- Dijagram koncepta
- Dijagram sekvenci
- Dijagram kolaboracije
- Dijagram klasa

Izrada dijagrama slučajeva upotrebe TehnIS

Korišćenjem IDEF0 metodologije(2) opisuje se funkcionalnost sistema, tj. definiše se šta sistem rad. Izradom dijagrama slučajeva upotrebe treba da se da odgovore i na pitanje kako sistem funkcioniše unutra. Drugim rečima, može se reći da je *dijagrami slučajeva upotrebe korisnički pogled funkcionisanja sistema.*

Na narednoj slici prikazan je dijagram slučajeva upotrebe TehnIS.



Slika 5.1. Dijagram slučajeva upotrebe TehnIS

Učesnici (Actor) je tehnolog, koji komunicira sa sistemom TehnIS. Učesnik prima i odašilje poruke koje nisu formalno prikazane..

Slučaj upotrebe predstavlja:

- "atomsku transakciju" jer po njegovom završetku IS sistem ostaje u konzistentnom stacionarnom stanju, dok na njega ne počnu da deluju drugi slučaj upotrebe.
- "logičku jedinicu posla" u realnom vremenu, nešto što ima značenje za korisnika, bez obzira na njegovu složenost.

Slučaji upotrebe: Izbor proizvoda, Definisanje Varijante tehnološkog postupka, Održavanje tehnološke lokacije i Opis operacije

Kratak opis: *Izbor proizvoda i Definisanje Varijante tehnološkog postupka* treba da omogući nalaženje proizvoda za koji je potrebno izbrati ili projektovati varijantu tehnološkog postupka. *Održavanje tehnološke lokacije* je zajednički šifranik i definiše ga po potrebi onaj kome to treba a svi ga koriste. *Opis operacije* je tekstualno opisuje operacije.

Učesnici: Tehnolog

Uslovi koji moraju biti zadovoljeni pre izvršavanja: Moraju biti azurni sifarnici.

Opis:. Pretpostavka za izvodjenje ovog slučaja upotrebe je da je izvršena identifikacija i standardizacija naziva proizvoda.Izbor proizvoda je asocijativnom vezom povesan preko stereotipa <<include>> sa slučajem upotrebe *Definisanje Varijante tehnološkog postupka.* Ovo znači da slučaj upotrebe *Izbor proizvoda* uključuje slučaj upotrebe *Definisanje varianate tehnološkog postupka.*
Održavanje tehnološke lokacije je slučaj upotrebe koji je izvan TehnIS-a i on se ovde samo koristi. Nastao je u internoj standardizaciji. *Opis operacije* je slučaj upotrebe gde se za dogovarajuću operaciju tekstualno opisuje operacije.

Izuzeci: Nema

Uslovi koji moraju biti zadovoljeni posle izvršavanja: Nema.

Slučaj upotrebe: Definisanje operacije

Kratak opis: *Definisanje operacije* generiše se tehnološki postupak (spisak operacija koji je definisan rednim brojem, nazivom (nazivi su takodje standardizovani), vremenom pripreme i trajanjem).

Učesnici: Tehnolog

Uslovi koji moraju biti zadovoljeni pre izvršavanja: Moraju biti azurni sifarnici.

Opis:. Stereotipom <<include>> ovaj slučaj upotrebe koristi slučajeve upotrebe *Opis operacije* i *Definisanje parametara.* Steretipom <<extend>> ovaj slučaj upotrebe proširuje se slučajem upotrebe *Održavanje tehnološke lokacij*e.
Izuzeci: Nema

Uslovi koji moraju biti zadovoljeni posle izvršavanja: Nema

Slučaj upotrebe: Definisanje parametara

Kratak opis: *Definisanje parametara* je slučaj upotrebe gde se za dogovarajuću operaciju bira izbor definisanja parametara uredjaja i parametara uzorka.

Učesnici: Tehnolog

Uslovi koji moraju biti zadovoljeni pre izvršavanja: Moraju biti azurni sifarnici.

Opis: Stereotipom <<include>> ovaj slučaj upotrebe koristi slučajeve upotrebe *Definisanje parametara uredjaja* i *Definisanje parametara uzorka. Definisanje parametara uredjaja* je slučaj upotrebe gde se definišu parametri mašine za izvodenje tehnološkog procesa. *Definisanje parametara uzorka* je slučaj upotrebe gde se definišu parametri uzorka koji se u okviru labaratorije ispituju.

Izuzeci: Nema

Uslovi koji moraju biti zadovoljeni posle izvršavanja: Nema

Izrada konceptualnog modela TehnIS

Suština konceptualnog modela je u pronalaženju klasa u domenu izrade tehnoloških procesa. To je svojevrsno razmatranje realnog sistema koji stoji pred nama.

Klase u konceptualnom modelu su samo rezultat pokušaja da se sagledavanjem realnog sistema prepoznaju neki entiteti koji bi mogli da konkurišu za softverske klasu, te da se prepoznaju neke veze i njihova kardinalnost.

U okviru izrade konceptualnog modela definišu se sledeće aktivnosti:

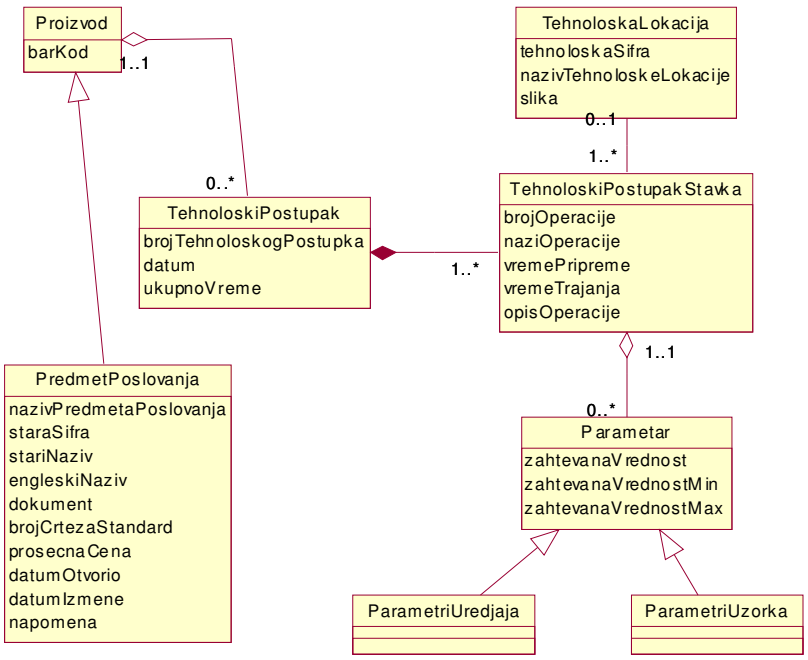
- Definisanje koncepta
- Definisanje asocijacija izmedju koncepata

Konceptom se opisuju stvari u realnom sistemu i na osnovu njih se kasnije, u "Izradi dijagrama klasa", definišu odgovarajuće klase i objekti koji definišu odgovarajuća softverska rešenja.

Definisanjem veza izmedju koncepata uspostavljaju se asocijacije izmedju predhodno definisanih koncepata. U okviru UML asocijacija se opisuje kao "strukturne relacije" izmedju objekata za različite tipove.

Apstrakcijama, će se otkriti da veoma mali broj koncepata stoje same.Većina njih sarađuje sa drugima na više načina. Stoga, kada se pravi konceptualni model, ne samo da se moraju identifikovati stvari koje formiraju rečnik modela, već, takođe, moraju se definisati i kako te stvari stoje jedna u odnosu na drugu.

Na narednoj slici prikazan je koncept TehnIS gde se mogu uočiti neke od gore opisanih osobina.



Slika 5.2. Konceptualni model TehnIS

Dakle, identifikovani su sledeći koncepti u konkretnom problemu realnog sistema: Predmet poslovanja, Proizvod, tehnološki postupak, Tehnološki postupak stavka, Tehnološka lokacija, Parametri, parametri uredjaja i parametri Uzorka.

U daljem tekstu detaljno će se specificirati svaki pojedinačni koncept.

Predmet poslovanja je generalizovani koncept nastao na višem nivou u okviru rečnika podataka ProteIS i ovde se prikazuje da bi se pokazala veza sa njegovom specijalizaciojom Proizvod. Predmet poslovanja je vezni koncept prema ostalim podsistemima u Sojaproteinu u okviru projekta ProteIS. *Ovde je definisan u okviru koncepta jer treba da zadovolji zahteve koji su postavljeni od strane održavanja.*

Proizvod je specijalizacija od predmeta poslovanja i moguće ga je samo koristiti a njegov nastanak je vezan za posistem interna standardizacija. Nasledjuje atribute iz klase PredmetaPoslovanja a njegov specifični atribut kojim se opsuje je BarKod.

Tehnološki postupak je koncept koji se ovde stvara i koji za jedan proizvod može da ima više varijanti. Za tehnološki postupak potrebno je definisati varijantu, datum nastanka i kao povratna informacija (izveden podatak) ukupno vreme.

Tehnološki postupak stavka su operacije koje se definišu za konkretan proizvod i odgovarajuću varijantu tehnoloskog postupka. Potrebno je definisati redni broj operacije, standradizovani naziv operacije, vreme pripreme i vreme trajanja.Posebno je moguće za svaku operaciju opisati operaciju kao i definisati parametre.

Parametri su generalizovani koncept i definiše za odgovarajuću operaciju čije su specijalizacije parametri uredjaja i parametri uzorka. Parametri su opisani zahtevanom vrednošću, zahtevana vrednost minimum i zahtevana vrednost maksimum.

Tehnološka lokacija je koncept kojim se definiše lokacija gde se operacija izvodi.

Na osnovu ovako definisanog koncepta u sledećem poglavlju definisaće se dijagrami interakcije kojima se definiše dinamika sistema tj. opisuje se klijent strana buduće aplikacije.

Dijagram sekvenci TehnIS

Dijagram sekvenci (Sequence Diagram) opisuje vreme trajanja poruke i način su na koji objekti u sistemu medjusobno komuniciraju, ostvarujući očekivano ponašanje. Dijagram sekvenci poseduju dve dimenzije: vreme i kolekciju objekata (objekat je instanca klase). Uobičajeno je da se vreme prikazuje po vertikalnoj, a kolekcija objekata po horizontalnoj dinmenziji. Na vertikalnoj liniji se može, na pogodan način predstaviti i vremensaka skala.

Objekti na dijagramu sevenci su predstavljeni vertikalnim linijama. Na vrhu linije se navodi naziv objekt i/ili simbol objekta. Aktiviranje objekta se predstavlja uskim pravougaonikom na liniji objekta i predstavlja operaciju (akciju) koju objekat, u periodu predstavljenom dužinom aktivacije, obavlja. Na vrhu aktivacionog objekta se prikazuje događaj (poruka) koja je aktivirala objekat, a na dnu povratna poruka objektu koji je aktivirao posmatrani objekat. Povratna poruka se često ne prikazuje, pogotovo kada daje očigledna i daje samo status okinute operacije.

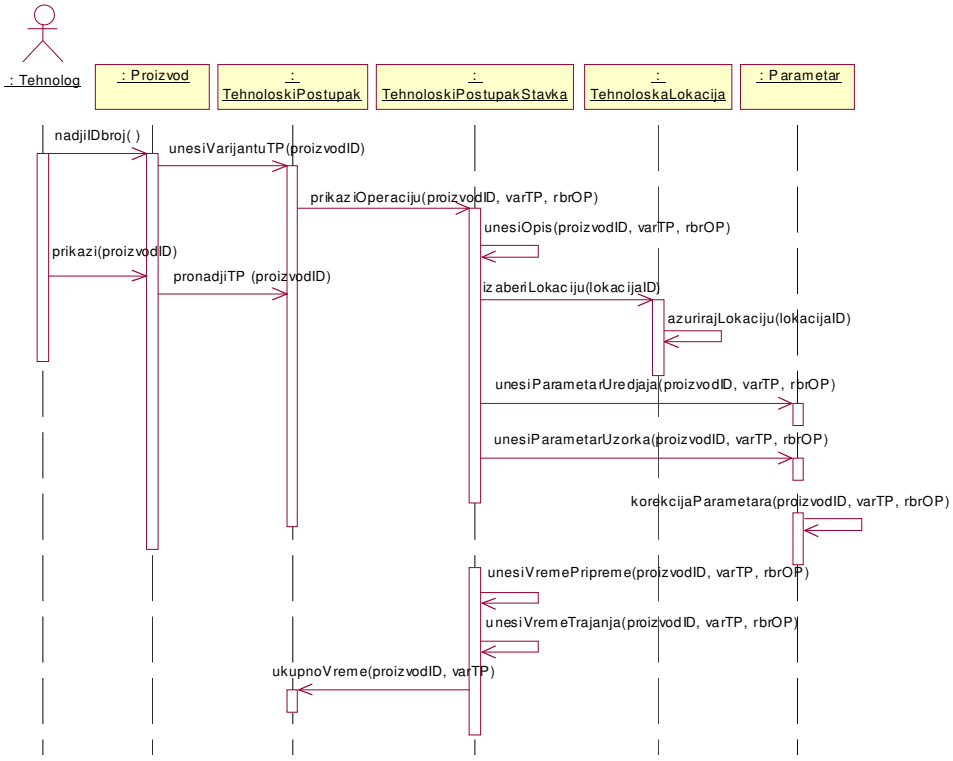
Iz svakog objekta polazi na dole isprekidna linija koja predstavlja njegov životni vek (lifeline). Životni vek predstavlja postojanje nekog objekta u periodu vremena. Većina objekata koji se pojavljuju u sekvencijalnom dijagramu postojaće i dok traje interakcija, pa su svi ti objekti poređani na vrhu dijagrama, sa svoji životnim vekom povučenim od vrha ka dnu dijagrama.

Na narednoj slici prikazan je dijagram sekvenci TehnIS.

Tehnolog u prvom koraku porukom NadjiID() pristupa objektu :Proizvod i bira odgovarajući ident broj proizvoda (ProizvodID) za koji će se praviti tehnološki postupak. Takođe je moguće porukom prikazi (proizvodID) daje prikaz svih proizvoda.

U sledećem koraku se porukom pronadjiTP (proizvodID) pristupti objektu :TehnoloskiPostupak i dobiti spisak projketovanih tehnoloških postupaka za taj ident broj proizvoda Ako se definiše novi tehnološi postupak porukom unesiVarijantu(proizvodID) pristupa se objektu :TehnoloskiPostupak počinje se sa projektovanjem.

Na osnovu predhodno definisan proizvod i za definisanu varijantu tehnoloskog postupka definiše se redni broj operacija i sve to prosledi porukom prikaziOperaciju (proizvodID, varTP, rbrOP) koja se kao operacija izvodi u objektu :TehnoloskiPostupakStavka. U okviru istog objekta se izvode operacije unesiOpis (proizvodID, varTP, rbrOP), unesiVremePripreme (proizvodID, varTP, rbrOP) i unesiVremeTrajanja (proizvodID, varTP, rbrOP).



Slika 5.3. Sekvencijalni dijagram TehnIS

Ovaj objekat prosledjuje poruku izaberiLokaciju(lokacijaID) objektu :TehnoloskaLokacija na osnovu koje biramo tehhnološku lokaciju na kojoj se izvodi operacija.

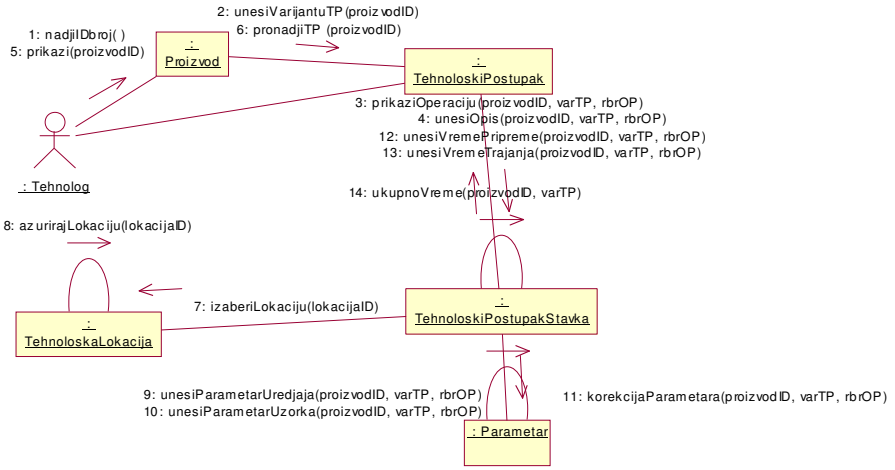
Objekat :TehnoloskiPostupakStavka prosledjuje poruke unesiParametarUredjaja (proizvodID, varTP, rbrOP) i unesiParametarUzorka(proizvodID, varTP, rbrOP) objektu :Parametri gde se izvod istomene operacije i izvrši unos odgovarajućih podataka u atribute objekta : Parametri.

Kada se definišu za sve operacije vremena prpreme i trajanja porukom ukupnoVreme (proizvodID, varTP) aktivira se istoimena operacija u objektu :TehnoloskiPostupak koja izvrši unos podatka u atribut ukupnoVreme.

Dijagram kolaboracije TehnIS

Dijagram saradnje (Collaboration Diagram) definiše komunikaciju, pa i veze izmedju objekata neophodne za ostvarivanje posmatrane komunikacije. Dijagram saradnje pored objekata i veza prikazuje i poruke koje objekti medjusobno prosledjuju, ostvarujući na taj način očekivano ponašanje. Dijagram saradnje opisuje strukturnu organizaciju objekata koji šalju i prikazuju poruke.Dijagram saradnje prikazuje interakciju izmedju skupa objekata koji se predstavljaju kao čvorovi nekog grafa. Kako razvoj sistema napreduje i kako struktura koncepta prelazi u strukturu klasa, tako i značaj dijagrama saradnje raste, a dijagrama sekvenci opada, jer je semantika dijagrama saradnje bogatija.

Drugim rečima, pošto su izvedeni iz istih informacija u UML-ovom metamodelu, sekvencijalni i dijagram saradnje su međusobno ekvivalentni. Kao posledica toga jedan se može prevesti u drugi bez bilo kakvog gubitka informacija kao što je pokazano na narednoj slici.



Slika 5.4. Dijagram kolaboracije TehnIS

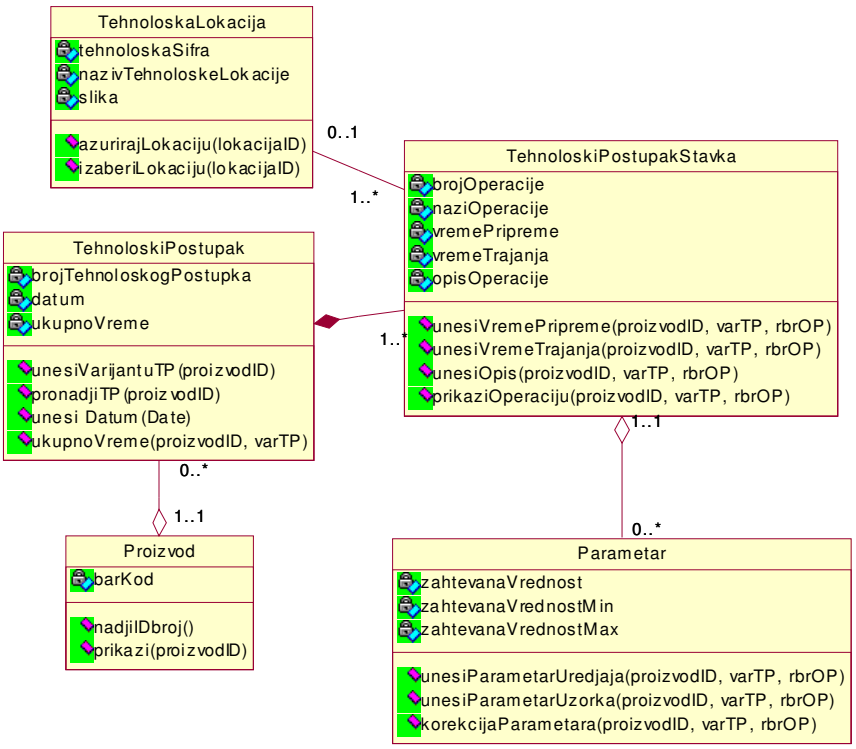
Saradnja (kolaboracija) izmedju objekata prikazuje se objektima i njihovim medjusobnim vezama. Komunikacija izmedju objekata opisuje se porukama koje objekti medjusobno razmenjuju, ostvarujući na taj način očekivano ponašanje i odredjenu funkcionalnost sistema.

Definisanje dijagrama klasa

Dijagram klasa (Class Diagram) prikazuje skup klasa, interfejsa i saradnja i njihovih relacija. Dijagram klasa je statički struktura klasa u sistemu koje izmedju sebe uspostavljaju relacija tipa asocijacija (veza sa svakom drugom), zavisnosti (jedna klasa zavisi/koristi se od druge klase), specijalizacije (jedna klasa je specijalizacija druge klase, i paketa (grupisanje u jednu jedinicu tj. paketi);

Potpuni dijagram klasa ili kako se kraće kaže dijagram klasa se tako zove da bi se odvojio od konceptualnog modela (koji se prikazuje kao dijagram klasa bez operacija). Klasa predstavlja složeni tip, kolekciju, strukturu koja se sastoji od više atributa (podata članova) i operacija (metoda, funkcija članica).

Na narednoj slici prikazan je dijagram klasa za TehnIS.



Slika 5.5. Dijagram klasa TehnIS

Dijagram klasa koji je dat na slici. opisuje logiku aplikacije TehnIS i definisan je sa tri ključne klase:TehnoloskiPostupak, TehnoloskiPostupakStavka i Parametri. Klase Proizvod i TehnoloskaLokacija ne nastaju u okviruTehnIS-a već služe za izbor i proširivanje TehnIS-a.

Za jedan tehnoloski postupak definiše se jedna ili vise stavki i ova veza je agregacija tj. *kompozicija* (*Composition*). Kod kompozicije, objekti-delovi su u ekskluzivnom vlasništvu samo jednog objekta-celine koji je odgovoran za njihovo kreiranje i uništavanje.

U okviru klase TehnoloskiPostupakStavka definiše se operacijai vezuje agregacijom tj. referenca. Ovaj tip agregacije je samo specijalna vrsta asocijacije i definisana je romboidom i predstavlja strukturalni odnos, što znači da su oba klase na istom nivou. Agregacija definiše odnos: celina/deo" tj. odnos :ima" (has-a), što znači da objekat celine ima objekte delova.

Klasa parametri objedinjuje u konceptu definisane klase ParametaraUredjaja i ParamataraUzorka.

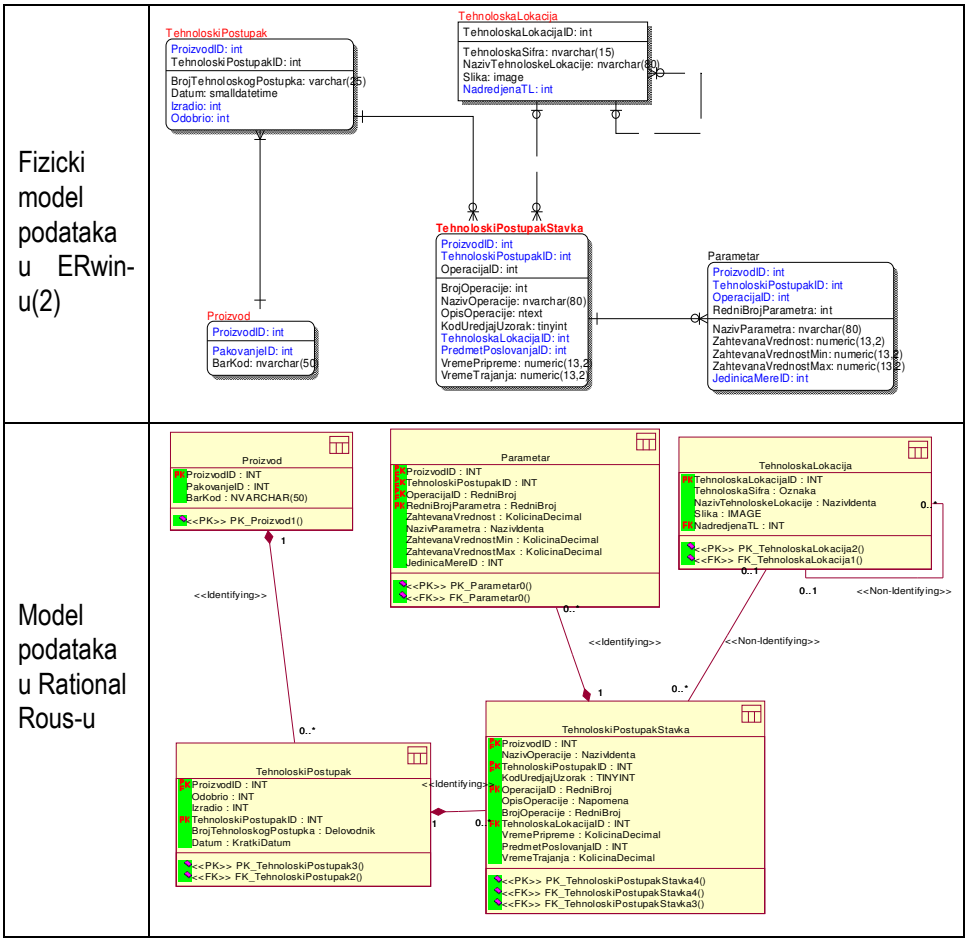
Neki atributi koji će se kasnije pojaviti u modelu podataka nedostaju a to je stoga što se ovde ne radi o tabelama i prenošenju stranih ključeva, već o softverskim klasama kod kojih se to rešava mogućnošću referenciranja.

Fizički model podataka TehnIS

Definisanje fizičkom modela podataka tj. implementacija klasa i njihovih atributa u tabele i kolone nekog SUBP, korišćenjem RationalRous-a, definisali smo na osnovu ERwin fizičkog modela podataka(2) postupkom reverse enginering. Prvo smo u ERwinu napravili skript fajl a potomo pozvali u RationalRous-u Tools-Data Modeler-Reverse Engeenering.

Šeme logičke baze podataka obuhvata poseban skup podataka (odgovarajući rečnik podataka) sa odgovarajućom semantikom i vezama među elementima baze podataka. Fizički, ove veze su smeštene u bazi podataka, za kasniju upotrebu.

Na sledećoj slici prikazana je šema baze podataka dobijena ovim postupkom.



Slika 5.6. Fizički model podataka TehnIS

Zaključak

Na osnovu definisanih dijagrama pristupa se izradi dva sledeća koraka. U prvom koraku prelazi se na definisanje server strane generisanjem baze podataka kao što je pokazano za ovaj primer u(2). Drugi korak podrazumeva izradu klijent strane korišćenjem MS Access-om kao što je pokazano u (2) i (3).

Literatura

1 Veljović A. Objektno modeliranje informacionih sistema, Fakultet za poslovne studije, MEGATREND Univerzitet, 2003. godina

2. Veljović A. Praktikum iz analize informacionih sistema, Fakultet za poslovne studije, MEGATREND Univerzitet, 2005. godina

3. Veljović A. i drugi, Projekat TehnIS, Sojaprotein Bečej, 2001. godina.

6. Poslovi cirkulacije u biblioteci

Uvod

Na primeru cirkulacije u biblioteci, detaljno će se prikazati sve faze objektnog modelranja informacionog sistema definisanog u (1). Ovaj primer predstavlja svojevrsan vodič kroz definisanje svih elemenata projekta razvoja informacionog sistema korišćenjem objektno orijentisanih CASE alata (Rational Rous). Osnove vezane za poslove cirkulacije u biblioteci date su u (2).

Osnova za praćenje faza OO projektovanja IS čine sledeće faze:

- Definisanje zahteva
    - Izrada dijagrama slučajeva upotrebe
    - Izrada dijagrama aktivnosti
- Objektno orijentisana analiza
    - Izrada konceptualnog modela
    - Izrada dijagrama sekvenci
    - Definisanje ugovora o izvrsenju operacija
- Objektno orijentisan dizajn
    - Izrada dijagrama saradnje
    - Izrada potpunih dijagrama klasa
    - Izrada dijagrama stanja
    - Definisanje paketa
- Implementacija
    - Izrada aplikacije
    - Definisanje tehnologije aplikativne i mrezne arhitekture

Predmet razmatranja ovog primera nije testiranje, uvodjenje i odrzavanje.

Definisanje zahteva za poslove cirkulacije u biblioteci

Osnovne pretpostavke vezane za definisanje zahteva poslova cirkulacije u biblioteci dati su u (2) i treba da bude veza sa elementima UML definisanim preko dijagrama aktivnosti i dijagrama slučajeva upotrebe.

Model poslovnih procesa (ili fizički model) je vezan za odgovarajuće organizaciono tehnološko okruženje i podrazumeva detaljan opis poslovnih procesa kao sekvence aktivnosti. Imajući ovo u vidu model poslovnih procesa podložan je češćim izmenama zbog promene organizacije i tehnologije obavljanja posla.

Ovde se pod poslovnim procesom definiše sekvenca (nit) aktivnosti kojima se ostvaruje neki cilj sistema ili zadovoljava zahtev korisnika.

Pod modeliranjem poslovnih procesa podrazumevaju se metode i alati koji se koriste za opis poslovnih procesa tj. definišu se:

- aktivnosti u svakom poslu kao i njegov redosled i uslovi pod kojima se odvija
- radna mesta (procesori) na kojima se aktivnosti odvijaju kao i
- dokumenta koji su ulaz, odnosno izlaz iz svake aktivnosti.

Izrada modela poslovnih procesa cirkulacije u biblioteci definisane su sledećim aktivnostima:

- Razvoj dijagrama slučajeva upotrebe za poslove cirkulacije u biblioteci
- Razvoj dijagrama aktivnosti za poslove cirkulacije u biblioteci

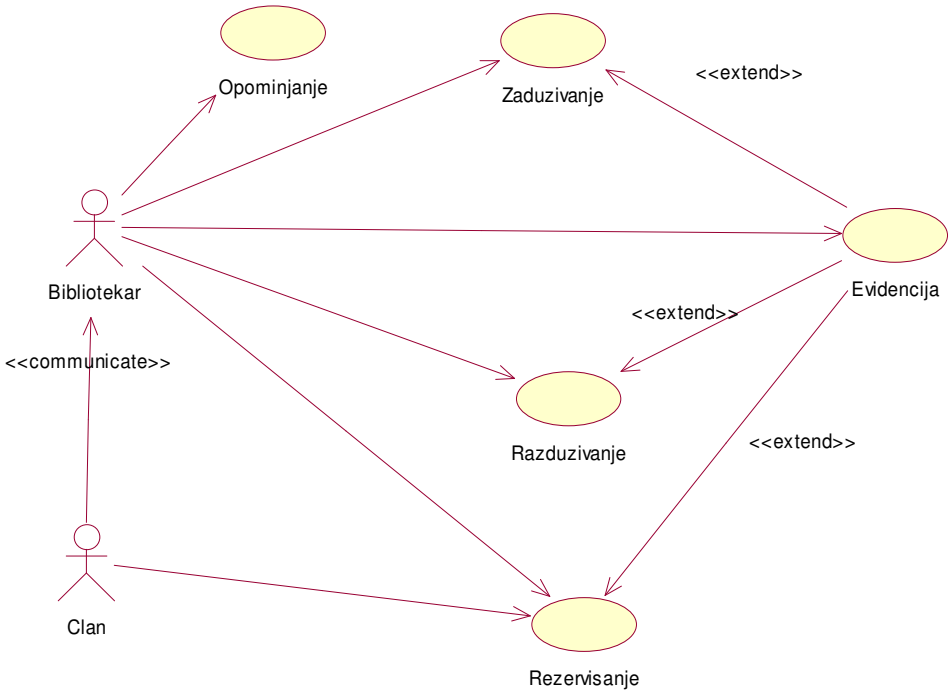
U daljem tekstu detaljno će se obrazložiti gore definisane aktivnosti.

Izrada dijagrama slučajeva upotrebe za poslove cirkulacije u biblioteci

U daljem radu koristićemo CASE alata RationalRose koji u potpunosti podržava UML notaciju za modelovanje dijagrama slučaja korišćenja. Model se ne sastoji samo od dijagrama već i od detaljnog opisa slučaja upotrebe.

Za opisivanje slučajeva upotrebe moraju se ispoštovati odgovorajući sadržaji koji sadrže kratak opis, učesnike, uslove koji se moraju zadovoljiti pre izvršenja, opis, izuzeci, i uslovi koji moraju biti zadovoljeni posle izvršavanja.

Slede dijagrami slučajeva upotrebe i tekstualni opis slučajeva upotrebe clirkulacije u biblioteci.



Slika 6.1. Dijagram slučaja upotrebe Cirkulacija

Slučaj upotrebe: Evidencija

Kratak opis: Vođenje podataka o članovima cirkulacije u biblioteci

Učesnici: Bibliotekar

Uslovi koji moraju biti zadovoljeni pre izvršavanja: Uredan spisak članova za upis sa tačnim ličnim podacima.

Opis: Svaka biblioteka mora imati ažurnu evidenciju o svojim članovima. Tu se vode lični podaci kao što je jedinstveni identifikator člana (ovde usvojeno JMBG), ime, prezime, telefon, radno mesto itd. Unos većine podataka sem pojedinih (kao broj telefona) je obavezan. Jedan član može samo jednom biti zaveden u evidenciju cirkulacije u biblioteci, i naravno iz cirkulacije u biblioteci se može ispisati samo član koji je već učlanjen.

Izuzeci: Za člana koji se upisuje postoji proverava se da li je tačno unesen JMBG. Ako član koji se ispisuje ne postoji proverava se tačnost unesenih podataka.

Uslovi koji moraju biti zadovoljeni posle izvršavanja: Uredna i tačna evidencija zapisa članova cirkulacije u biblioteci .

Slučaj upotrebe: Zaduživanje

Kratak opis: Članovi cirkulacije u biblioteci izuzimaju naslove iz bibliotečkog fonda.

Učesnici: Bibliotekar

Uslovi koji moraju biti zadovoljeni pre izvršavanja: Član i naslov postoje; zaduženje je evidentirano.

Opis: Član traži naslov i ako nije na zaduženju zadužuje ga na period od 30 dana.

Izuzeci: Za naslov koji je na pozajmici može se samo izvršiti rezervisati. Za naslov koji ne postoji proverava se da li je tačno unesen kriterijum pretrage. Ako takvo zaduženje ne postoji proverava se da li je knjiga rashodovana.

Uslovi koji moraju biti zadovoljeni posle izvršavanja: Tačna evidencija svih zaduženja koja su tekuća i još nisu namirena.

Slučaj upotrebe: Razduživanje

Kratak opis: Članovi cirkulacije u biblioteci vraćaju naslove iz bibliotečkog fonda.

Učesnici: Bibliotekar

Uslovi koji moraju biti zadovoljeni pre izvršavanja: Član i naslov postoje; zaduženje je evidentirano.

Opis: Pri razduživanju identifikuje se zaduženje člana i briše se zapis o njemu.

Slučaj upotrebe: Opominjanje

Kratak opis: Članovima koji su zadržali naslov duže od 30 dana šalje se opomena.

Učesnici: Bibliotekar

Uslovi koji moraju biti zadovoljeni pre izvršavanja: Tačan sistemski datum i tačni podaci o datumu zaduženja

Opis: Svakodnevno pri startovanju aplikacije prolaze se svi zapisi o zaduženjima i na osnovu datuma zaduženja i sistemskog datuma utvrđuju se prekoračenja za koja podsistem nudi mogućnost slanja opomene. Opomena se ne mora poslati kao što i sam bibliotekar može pokrenuti filtriranje zaduženja u potrazi za vremenskim prekoračenjima. Opomena se štampa i šalje članu.

Izuzeci: (štampač ne štampa opomenu) proveriti da li u štampaču ima papira; (Podaci na opomeni nisu kompletni) proveriti da li se za datog člana i naslov vode svi relevantni podaci.

Uslovi koji moraju biti zadovoljeni posle izvršavanja: Svi zapisi iz tabele zaduženja koji su prekoračili vremenski rok vraćanja naslova, smešteni su u tabelu opomena.

Slučaj upotrebe: Rezervisanje

Kratak opis: Javljanje člana na listu čekanja za naslov koji je već zadužen.

Učesnici: Bibliotekar, Član

Uslovi koji moraju biti zadovoljeni pre izvršavanja: Naslov koji se zadužuje postoji u vidu zapisa.

Opis: Član pretražuje naslove i kada nađe odgovarajući a on je već na zaduženju kod drugog člana javlja se na listu čekanja za taj naslov. Sada nakon vraćanja naslova u biblioteku on nemože biti zadužen niti od jednog člana osim od onog koji ga je rezervisao i to po FIFO(First In Firs Out) algoritmu čekanja. Do sada se ta funkcija vršila kroz komunikaciju člana sa bibliotekarom, dok bi uvođenjem novog tehnološkog okruženja tu funkciji mogli da vrše i članovi samostalno.

Izuzeci: (naslov ne postoji) pogledati da li su dobro uneti podaci za pretragu

Uslovi koji moraju biti zadovoljeni posle izvršavanja: Tačno formiran redosled članova po datumu rezervacije.

Izrada dijagrami aktivnosti za poslove cirkulacije u biblioteci

Zbog boljeg razumevanja korisničkih zahteva, dijagramima aktivnosti će se opisati mehanizam izvršavanja slučaja upotrebe koji su prethodno opisani.

Dijagram aktivnosti definisan je:

- plivačkim stazama
- stanjem i tranzicijom

Plivačke staze ili procesori su logičke celine nastali na osnovu definisane organizacije biblioteke. U plivačke staze se navode učesnici, aktivnosti koje dati procesor obavlja i dodeljuju odgovornosti za izvršenje akcija u pojedinim logičkim celinama.

Dakle plivačke staze (swimlanes) specificiraju odgovornosti za delove celokupne aktivnosti i nemaju neku duboku semantiku. Staza obično reprezentuje neki entitet realnog sveta. *Stanja* pripadaju stazama, a *tranzicije* mogu prelaziti iz jedne staze u drugu.

Čvorovi grafa predstavljaju aktivna stanja posla (aktivnosti), a neimenovane linije su tranzicije. Stanje akcija i aktivnosti tako se rasporedjuju da se vidi odgovornost objekata ili učesnika u sistemu za njegovo izvršenje.

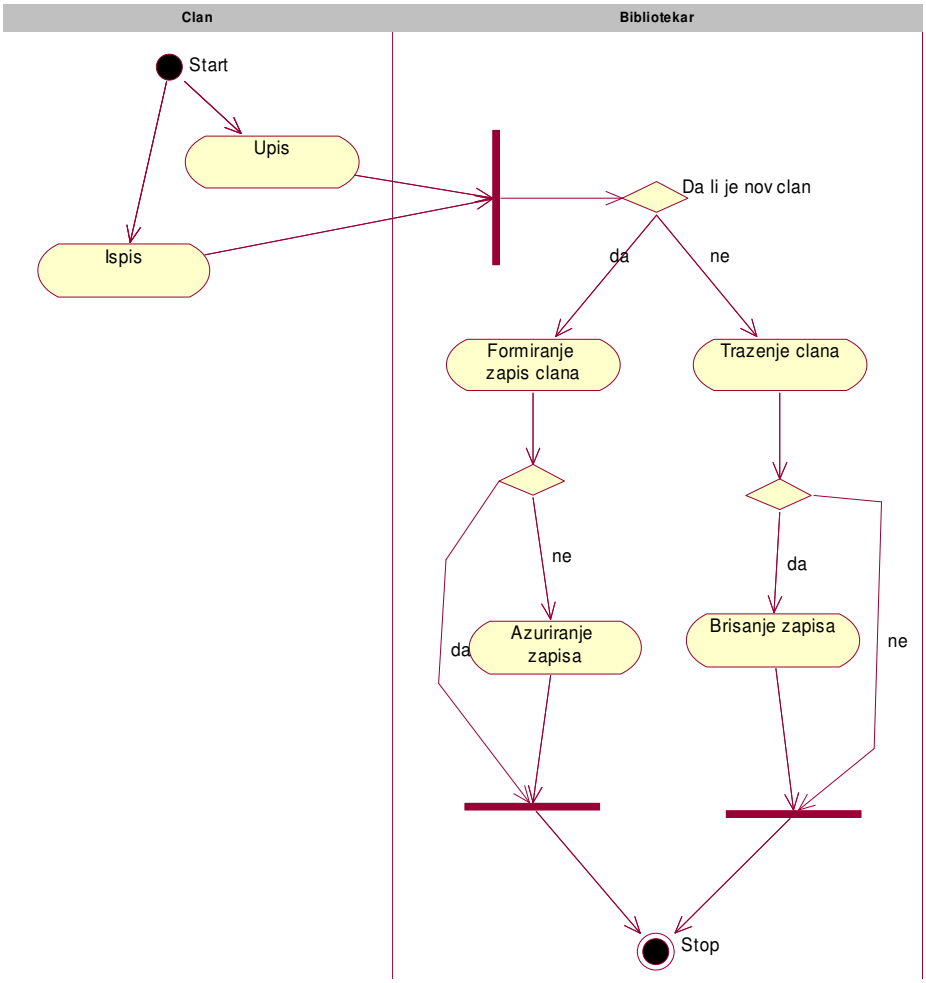
Slede dijagrami aktivnosti za pojedine slučajeve upotrebe.

Dijagram aktivnosti za vođenje evidencije članova

Svaki dijagram aktivnosti, počinje startnom aktivnošću koja se predstavlja ispunjenim crnim krugom. Imajući u vidu da je već rečeno da slučaj upotrebe evidencija članova biblioteke podrazumeva i upis i ispis članova tako će i dalji tok aktivnosti u ovom dijagramu zavisiti od aktivnosti koje će član obavljati u svom domenu (plivačkoj stazi). Drugim rečima u zavisnosti da li se član upisuje ili ispisuje dolaziće do upisa odnosno ispisa iz evidencije članova biblioteke.

U zavisnosti da li se radi o upisu ili ispisu člana, bibliotekar će vršiti ili formiranje zapisa korisnika ili njegovo pretraživanje. U oba konkurentna toka postoje dva identična stanja odluke koja postavljaju sigurnosno pitanje: "Da li je nov član?". Time se obezbeđuje da se ne dupliraju upisi istog člana biblioteke, a u drugoj grani proverava osnovanost brisanja zapisa člana koji mora da postoji da bi uopšte bio izbrisan. Dijagram se naravno završava oznakom krajnjeg stanja koje se predstavlja krugom koji nije do kraja ispunjen.

Na sledećoj slici prikazan je dijagram aktivnosti za vođenje evidencije članova.



Slika 6.2. Dijagram aktivnosti za vođenje evidencije članova

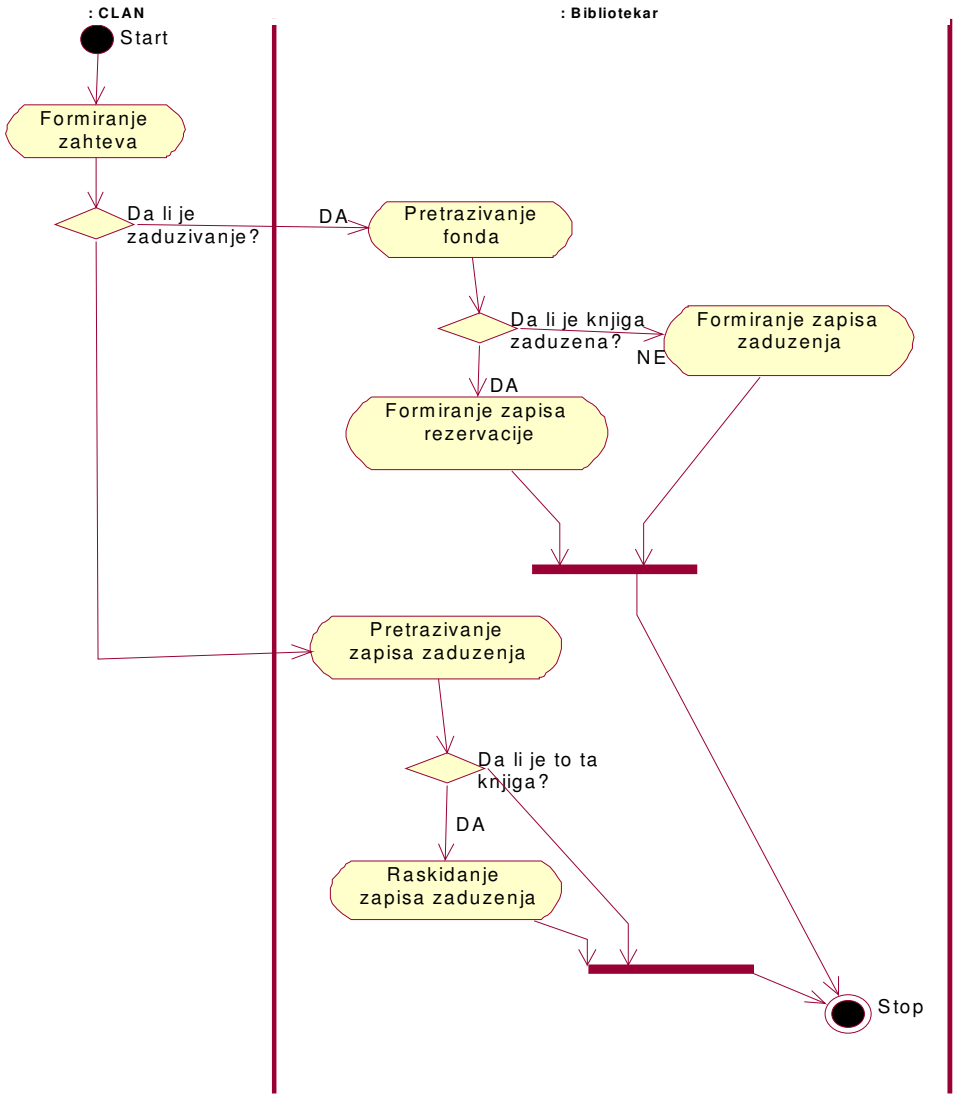
Dijagram aktivnosti za zaduživanje i razduživanje

Na sledećoj slici prikazan je dijagram aktivnosti za osnovnu i najvažniju funkciju bibliotečkog poslovanja, opsluživanje članova naslovima iz fonda biblioteke. Dakle, radi se o zaduživanju i razduživanju naslova iz biblioteke. I ovde se susrećemo sa domenima (plivačkim linijama) odgovarajućih učesnika u korespodenciji.

Član formira zahtev (bilo da želi da zaduži ili razduži knjigu), koji nije formalizovan već se ovde prosto misli na izražavanje želje člana za obavljanjem odgovarajuće operacije. Informacije koje će on prosleđivati bibliotekaru razlikovaće se u zavisnosti od toga da li se radi zaduživanju ili razduživanju naslova, ali to nije aspekt koji razmatra dijagram aktivnosti.

Bibliotekar će nakon toga izvršavati bilo pretraživanje naslova koji se želi zadužiti bilo zapisa o zaduženju prvenstveno orjentisanom ka ličnim podacima člana biblioteke.

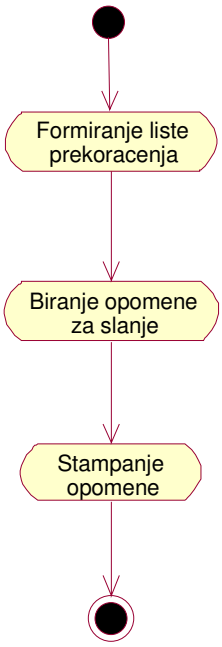
Pratimo za trenutak granu zaduživanja. Nailazimo na stanje odluke kome je uslov: "Da li je naslov zadužen?". U slučaju da nije, na osnovu svih raspoloživih podataka vrši se formiranje zapisa zaduženja. U suprotnom slučaju vrši se rezervacija naslova. Vratimo se grani razduženja. U tom slučaju vrši se pretraživanje zapisa zaduženja na osnovu ličnih podataka člana biblioteke. Drugim rečima proverava se da li se kod tog člana nalazi na zaduženju knjiga koja je podneta na uvid bibliotekaru. Samo u slučaju da je rezultat takve pretrage pozitivan izvršiće se raskid zapisa zaduženja. U suprotnom takva aktivnost se neće desiti i stanje zaduženja ostaće nepromenjeno.



Slika 6.3. Dijagram aktivnosti za zaduživanje i razduživanje

Dijagram aktivnosti opominjanje

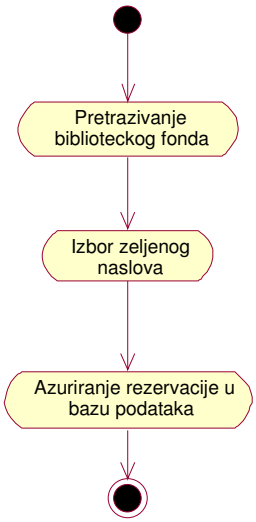
Na sledećoj slici dat je dijagram aktivnosti za slučaj upotrebe opominjanja korisnika. Sastoji se od tri aktivnosti koje se odvijaju linearno jedna za drugom. Da bi se izdala opomena nekom članu biblioteke potrebno je pretražiti sva zaduženja i naći sva ona koja nisu regulisana na vreme. Tako se formira lista prekoračenja. S obzirom da je po pravilu takvih zaduženja više od jednog, potrebno je izabrati jedno od njih, jer se u jednom momentu može štampati samo jedna opomena. Nakon toga sledi aktivnost štampanja opomene na osnovu izabranog prekoračenja.



Slika 6.4. Dijagram aktivnosti opominjanje korisnika

Dijagram aktivnosti za rezervisanje

Ni ovaj dijagram kao ni prethodni ne karakterišu tzv. plivačke linije jer su sve aktivnosti koje su obuhvaćene dijagramom u zoni odgovornosti bibliotekara. Doduše ostavlja se verovatnoća (koja će kasnije biti detaljnije razrađena) da rezervaciju preko odvojenih terminala vrši i sam član biblioteke. Aktivnosti koje se izvršavaju u sastavu slučaja upoterebe rezervisanja naslova su pretraživanje bibliotečkog fonda, izbor željenog naslova i ažuriranje rezervacije u bazu podataka. U praksi se retko dešava da član dođe sa formiranom idejom o tačnom naslovu koji želi da izuzme iz biblioteke. Najčešće član ima samo neku širu ili užu oblast interesovanja ili neki parcijalni podatak o naslovu. U tu svrhu su predviđeni različiti kriterijumi pretrage koji najčešće daju više rezultata pretrage. Iz dobijene liste potrebno je odabrati neki od naslova i izvršiti rezervaciju. Kraj rezervisanja se ogleda u aktivnosti ažuriranja rezervacije u bazu podataka. Nakon te aktivnosti naslov je rezervisan i može ga zadužiti samo član koji ga je rezervisao. Naravno, ta rezervacija ne može stojati večno i pitanje je politike poslovanja biblioteke koliki je taj dozvoljeni period između dana rezervisanja i dana zaduženja. Period o kome se govori ne opisuje se dijagramom aktivnosti.



Slika 6.5. Dijagram aktivnosti za rezervisanje

Objektno orjentisana analiza za poslove cirkulacije u biblioteci

Objektno orijentisana analiza poslova cirkulacije u biblioteci koristi se za definisanje ključnih koncepata (ključnih apstrakcija) i veza izmedju koncepata (mehanizama) vezanih za domen problema.

U okviru ove faze definisaće se:

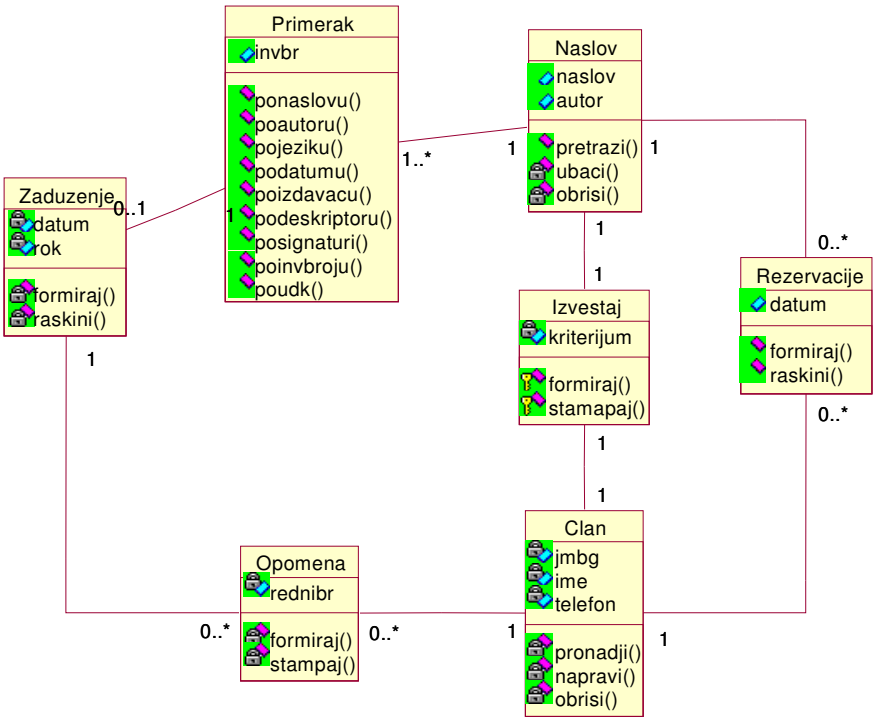
- Izrada konceptualnog modela
- Izrada dijagrama sekvenci
- Definisanje ugovora o izvršenju operacije

U daljem tekstu detaljno će se opisati gore definisane aktivnosti.

Izrada konceptualnog modela za poslove cirkulacije u biblioteci

Suština konceptualnog modela je u pronalaženju klasa u domenu problema kojim se bavimo. To je svojevrsno razmatranje realnog sistema koji stoji pred nama.

Klase u konceptualnom modelu su samo rezultat pokušaja da se sagledavanjem realnog sistema prepoznaju neki entiteti koji bi mogli da konkurišu za softverske klasu, te da se prepoznaju neke veze i njihova kardinalnost. Konceptualni model će morati da pretrpi krupne promene kako bi kroz ostatak faze analize i faze dizajna rezultovao dijagramom klasa.



Slika 6.6. Konceptualni model cirkulacije u biblioteci

Dakle, identifikovani su sledeći koncepti u konkretnom problemu realnog sistema: Član, Naslov, Zaduženje, Primerak, Izveštaj, Opomena i Rezervacija.

U daljem tekstu detaljno će se specificirati svaki pojedinačni koncept.

Naslov je klasa koja predstavlja apastrakciju stavke bibliotečkog fonda, skupa svih knjiga u biblioteci.

Ovaj koncept sastoji se od sledećih javnih atributa:

- naslov: je niz znakova koji predstavlja ime naslova
- autor: atribut znakovnog tipa koji čuva ime i prezime osobe koja je napisala konkretni naslov.

Primerak je koncept realnog sistema gde je moguće i dešava se da jedan naslov ima više primeraka. Razdvajanje koncepata naslova i primerka je potrebno zbog različitih operacija koje se odvijaju nad instancama ovih klasa.

Ovaj koncept sastoji se od sledećeg javnog atributa:

- invbr: je jedinstveni identifikator svakog primerka naslova

Član je koncept koji predstavlja svakog legitimnog korisnika bibliotečkog fonda.

Ovaj koncept sastoji se od sledećih privatnih atributa:

- jmbg: je jedinstveni matični broj građanina(člana) koji je ovde na neki način i članski broj
- ime: je ime i prezime člana biblioteke

- telefon : je atribut koji čuva podatak o telefonskom broju člana biblioteke.

Zaduženje je klasa-koncept koji predstavlja apstrakciju veze koju mogu da ostvare između sebe jedan član i jedan primerak.

Ovaj koncept sastoji se od sledećih privatnih atributa:

- datum: je podatak član koji sadrži datum formiranja konkretnog zaduženja.
- rok: može ili biti broj dana na koji se daje primerak na čitanje ili konkretan datum do kojeg se izdaje primerak.

Opomena je apstrakcija odnosa koji može postojati između člana i njegovog zaduženja, u slučaju da ne razduži primerak do datog mu roka.

Ovaj koncept sastoji se od sledećih privatnih atributa:

- rednibr: je celobrojna vrednost rednog broja konkretne opomene za konkretnog člana.

Rezervacija je koncept koji realizuje apstrakciju odnosa koji može postojati između jednog naslova i jednog člana

Ovaj koncept sastoji se od jednog javnog atributa:

- datum: je datum kada se naslov rezervisao

Izveštaj je koncept realnog sistema koji predstavlja dokument izveštavanja o korisniku i naslova.

Ovaj koncept sastoji se od sledećih privatnih atributa:

- kriterijum: je argument koji govori o tome da li se radi o izveštaju o knjizi ili o izveštaju o naslovu.

Može se reći da svaka koncept u konceptualnom modelu opisan je dodeljenim atributima i nekim elementarnim operacijama. Izuzetno je važno odvojiti koncept naslova od koncepta primerka. Naime dok je naslov jedan, primeraka može biti više (u slučaju naslov udžbenika redovno ima više primeraka), na šta ukazuje i kardinalnost veze između ova dva koncepta. Logično je onda što se naslov karakteriše atributima kao što su ime naslova i autora, a primerak inventarnim brojem koji ga jedinstveno razlikuje od ostalih primeraka istog naslova. Treba primetiti da koncept rezervacije povezuje koncepte člana i naslova, dok su koncepti člana i primerka povezani asocijativnim konceptima zaduženja i opomena.

Upravo se ovde vidi suština konceptualnog modela-skeniranje realnog sistema u potrazi za klasama objekata i značenjem njihovih međusobnih veza. Idući tim putem došlo se do toga da član rezerviše naslov bez obzira na primerak (jer je to u tom slučaju nebitno), dok se član zadužuje sa konkretnim primerkom. Dakle, konceptualni model, opisuje relaciju koncepata člana i primerka, koja je sama srž funkcionisanja realnog sistema bibliotečkog poslovanja. Tako primerak sa jedne strane može biti u jednom vremenskom momentu zadužen jednom ili nijednom, a jedno zaduženje se može voditi za samo jedan primerak. Idući dalje trasom veze koncepata član i primerak može se videti da je po jednom zaduženju moguća ili nijedna ili više

opomena, a jedna opomena može biti izdata za jedno i samo jedno zaduženje. Istovetna je i kardinalnost veze opomena-član, gde član može imati nijednu ili više opomena, a jedna opomena može biti izdata za jednog i samo jednog člana. Poslednji na ovom dijagramu je koncept izveštaja, koji je kao što se vidi zajednički i za člana i za naslov. Veze na oba pravca su jednostruke, zato što u realnom sistemu izveštaj može u jednom trenutku biti vezan za jednog i samo jednog člana i jedan naslov, a važi i obrnuto.

Izrada dijagrama sekvenci za poslove cirkulacije u biblioteci

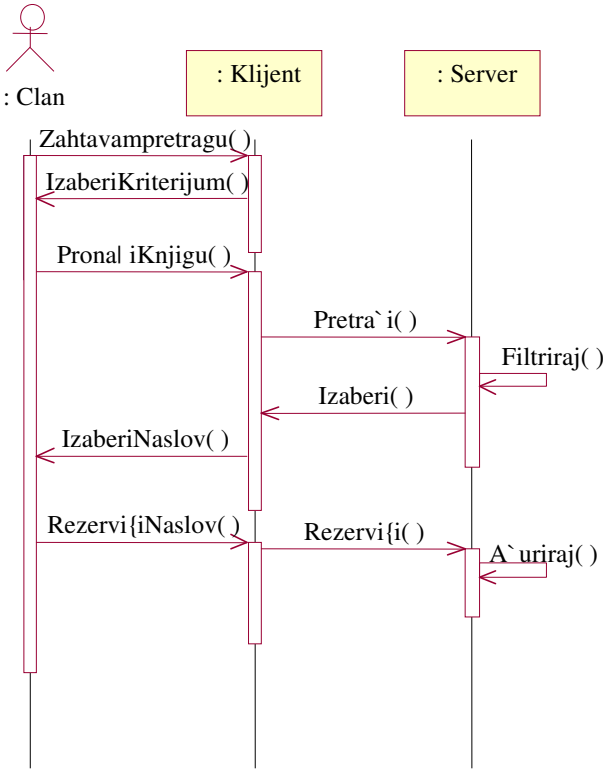
Dijagrami sekvenci spadaju u grupu interakcionih dijagrama koji služe za opis dinamičkog aspekta modela. Pored sekvencijalnih u ovu grupu dijagrama spadaju i dijagrami saradnje. Naziv potiče od tuda što je u središtu pažnje ovih dijagrama međusobna komunikacija, interakcija između objekata različitih klasa. Obe vrste dijagrama (i sekvencijalni i kolaboracioni) semantički su jednaki i moguće je vršiti međusobnu transformaciju, zato što nose iste informacije. Jedino u čemu se razlikuju je ugao gledanja na informacije koje nose. Kod sekvencijalnih dijagrama naglasak je na vremenski redosled odvijanja poruka između objekata različitih klasa, dok kolaboracioni dijagrami naglašavaju strukturu veza između objekata u interakciji. Dakle, sekvencijalni dijagrami se crtaju na vremenskoj osi, i predstavljaju specifikaciju vremenski zahteva u pogledu šta sistem treba da radi u realnom vremenu. Zato će za razliku od svog sabrata iz grupe interakcionih dijagrama (kolaboracioni dijagrami koji će biti korišćeni u fazi dizajna problema) biti iskorišćeni ovde za opis odigravanja događaja pojedinih slučaja upotrebe. Vremenskim redosledom poruka u sekvencijalnom dijagramu opisaće se logika odvijanja osnovnih funkcija poslova cirkulacije u biblioteci datih slučajevima upotrebe u prethodnoj fazi *definisanja zahteva*. Dakle, videće se šta podsistem treba da radi u realnom vremenu da bi obavio svoju funkciju predstavljenu slučajom upotrebe.

Sekvencijalni dijagram za rezervisanje naslova

Da bi član biblioteke, od kojeg i počinje interakcija objekata, rezervisao neki naslov on mora pretražiti naslove koji postoje u bibliotečkom fondu. Zato je prva poruka koju šalje član računaru klijentu ona koja zahteva pretragu. S obzirom da se naslov može tražiti po više kriterijuma kao odgovor na prethodnu poruku klijent će tražiti od člana da izabere kriterijum po kojem će pretraživati naslov (po jeziku, izdavaču, datumu unosa, signaturi, inventarnom broju, deskriptoru, naslovu, autoru). Odgovor klijenta je izbor jednog od navedenih kriterijuma i poruka klijentu da pretraži naslov. Klijent, čija je uloga u funkcionisanju podsistema posrednička, kao odgovor na tu poruku jednostavno prosleđuje zahtev sa odgovarajućim parametrima serveru. Server, pak po prijemu poruke vrši filtriranje svoje baze zapisa o naslovima po dobijenim parametrima. Rezultat takve aktivnosti je raznolik. Može se po zadatom kriterijumu dobiti nijedan, jedan ili više naslova koji odgovaraju zadatom kriterijumu. Imajući u vidu da se može rezervisati samo jedan naslov biće potrebno da se izabere jedan sa dobijene liste. Zato server šalje klijentu poruku kojom se zahteva izbor, a klijent taj zahtev članu u vidu liste sa koje se mora izabrati jedan od naslova da

bi se izvršila rezervacija. Član se odlučuje za jedan od naslova i šalje poruku o rezervaciji klijentu. On tu poruku prosleđuje sa odgovarajućim parametrima serveru gde se vrši ažuriranje rezervacije. Sada je obavljena rezervacija željenog naslova. Naravno da jedan član može rezervisati više naslova ali se u okviru jednog rezervisanja može rezervisati jedan i samo jedan naslov.

Na sledećoj slici prikazan je sekvencijalni dijagram za rezervisanje naslova.



Slika 6.7. Sekvencijalni dijagram za rezervisanje naslova

Sekvencijalni dijagram za izveštavanje i backup-ovanje

Za bolje razumevanje dijagrama sekvenci prikazanog na sledećoj slici potrebno je duž njene leve vertikalne ivice zamisliti vremesku osu, kao bi se imala potpuna predstava odvijanja poruka u realnom vremenu. Treba reći da imena poruka nisu obavezujuća te da je za funkcionisanje sistema bitno da za objekat koji prima poruku ona ima smisao imena koji nosi. Moguće je bilo razbiti na tri odvojena dela koji bi respektivno prikazivali sekvence poruka za izveštavanje o

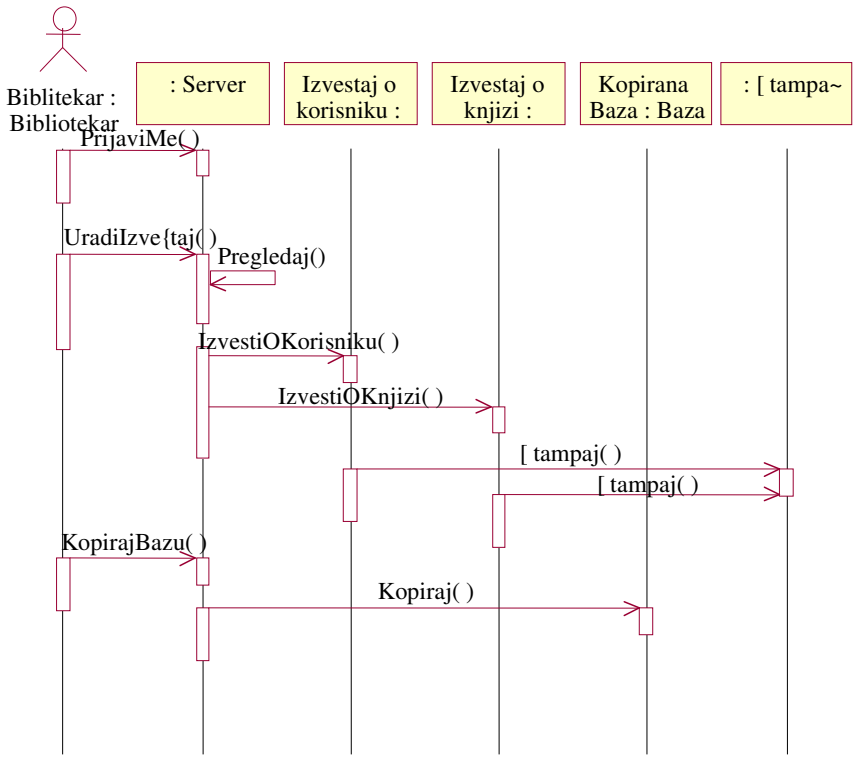
korisniku, izveštavanje o naslovu i backup-ovanje, ali to ovde nije učinjeno zbog izbegavanja nepotrebnog gomilanja, a istovremeno se ne gubi ništa na opisu funkcionalnosti.

Dakle, da bi se izvršila funkcija izveštavanja potrebno je da prethodno da objekat klase bibliotekar pošalje serveru poruku kojom zahteva svoju identifikaciju i početak rada. Nakon toga šalje poruku kojom zahteva kreiranje izveštaja. Posle dobijane poruke server vrši pregled odgovarajućih zapisa koji sadrže podatke relevantne za izveštaj.

Dalje su moguća dva događaja ili da server pošalje poruku kojm se zahteva pravljenje izveštaja o korisniku ili izveštaja o knjizi. Odatle se od objekata izveštaj o knjizi i izveštaj o korisniku prema primerku klase štampača šalje se od objekta poruka kojom se zahteva štampanje odgovarajućeg izveštaja.

Imajući u vidu dijagram koji je dat primećuje se da je vremenski redosled poruka *izvesti o korisniku* i *izvesti o knjizi nebitan*. Važno je samo da poruka *štampaj* sledi iza ovih poruka, a stvar je bibliotekara da li će prvo želeti da se obavesti o knjizi ili o korisniku(članu). Na samom kraju vertikalne vremenske ose prikazane su poruke koje se razmenjuju između objekata u funkciji backup-ovanja koja podrazumeva pravljenje rezervne baze podataka. Bibliotekar šalje poruku serveru za kopiranje podataka, koju on prosleđuje objektu kopirana baza. Naravno da se i ova sekvenca poruka vezana za backup-ovanje mogla u vremenu desiti i ranije ali je to samo stvar izbora primerka klase bibliotekar da li će prvo želeti da napravi izveštaj ili rezervnu kopiju podataka iz podsistema bibliotečkog poslovanja.

Na sledećoj slici prikazan je sekvencijalni dijagram za izveštavanje i backup-ovanje

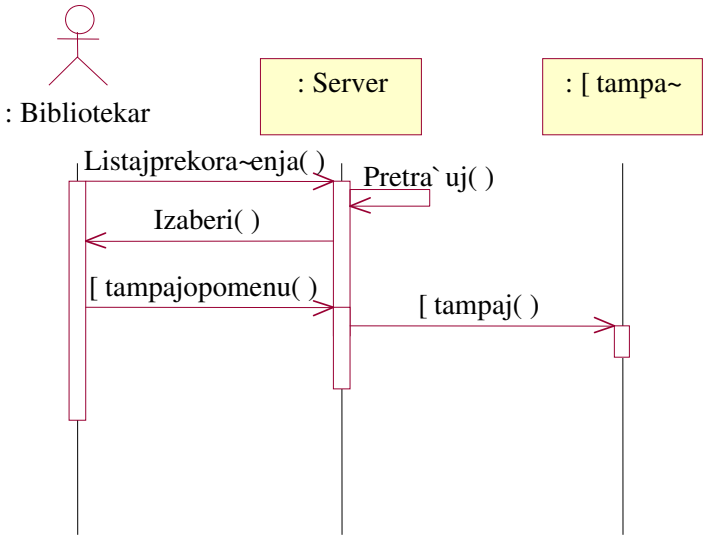


Slika 6.8. Sekvencijalni dijagram za izveštavanje i backup-ovanje

Sekvencijalni dijagram za opominjanje članova

Na sledećoj slici prikazana je vremenska sekvenca poruka koje razmenjuju objekti prilikom ostvarenja funkcije opominjanja. Potrebno je da na početku objekat klase bibliotekar pošalje serveru poruku za listanje prekoračenja roka zaduženja. Ta poruka izavaće pretraživanje na serveru svih zaduženja koja nisu raskinuta do roka koji je postavljen. Rezultat te pretrage u realnom sistemu je po pravilu spisak koji sadrži više od jednog prekoračenog zaduženja. Stoga će svojevrsna povratna poruka servera ka bibliotekaru biti da se izbere neko od tih prekoračenja kako bi se mogla štampati opomena. Ta poruka nije SW već je će prevashodno biti u vidu ekranskog prikaza spiska prekoračenja koji će prisiliti bibliotekara da izabere neke od njih kako bi se mogla štampati opomena. Sada bibliotekar kao reakciju na poruku servera da izabere neko od prekoračenja šalje poruku da se štampa jedna konkretna opomena. Reakcija servera na tu poruku je njeno prosto prosleđivanje štampaču.

Na sledećoj slici prikazaće se dijagram sekvenci za rezervisanje naslova i to za onaj njegov modalitet koji se oslanja na mrežnu arhitekturu podsistema. Razlog za to je nešto složenija saradnja između objekata u tom modu rezervisanja, koji je posledica dodavanja još jednog nivoa posredovanja u pristupu podsistemu (klijenta preko kojeg se pristupa podacima na serveru).



Slika 6.9. Sekvencijalni dijagram za opominjanje članova

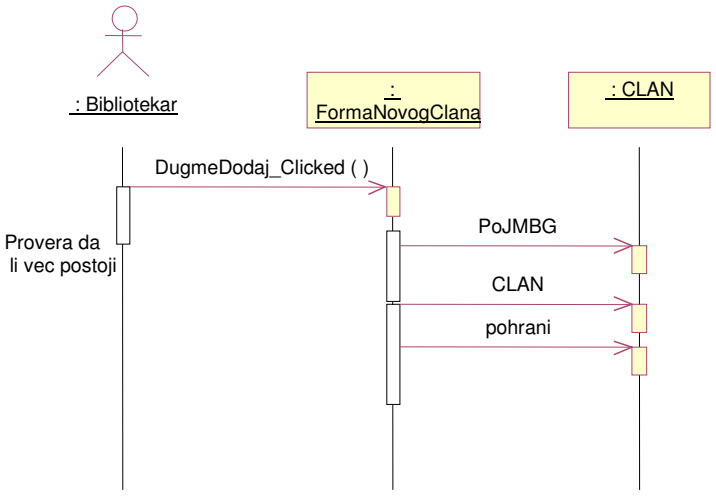
Izrada sekvencijalnih dijagrama interfejsa

Pored logike osnovnih funkcija koje treba da obavlja informacioni podsistem poslova cirkulacije u bibliotečkom poslovanju, moguće je sekvencijalnim dijagramima opisati i vremenski redosled razmene poruka između različitih formi (maski) aplikacije u toku izvršavanja specifičnih funkcija podsistema. Činjenica je da bi se svi dijagrami vezani za modelovanje korisničkog interfejsa podsistema mogli svrstati u objektno orjentisani dizajn, s obzirom da se radi o rešenju (kako sistem treba da obavlja osnovne funkcije posredstvom interfejsa). Svrastavanje ovakvih sekvencijalnih dijagrama u poglavlje posvećeno objektno orjentisanoj analizi treba shvatiti uslovno, te da su tu dati radi neprekidnosti izlaganja o sekvencijalnim dijagramima. Na dijagramima pojaviće se objekti (primerci) nekih novih klasa koje će kasnije podrobnije biti dati u dijagramima klasa interfejsa. Te klase uglavnom predstavljaju različite forme (maske) koje še se pojavljivati u aplikaciji.

Sekvencijalni dijagram interfejsa dodavanja člana

Prilikom upisa novog člana u biblioteku pred bibliotekarem koji obavlja tu funkciju nalazi se forma za unos novog člana. Da bi uneo podatke za novog člana on mora da pritisne dugme Dodaj, koje se nalazi na formi. Potom se vrši provera da li takav član već postoji i ta provera se vrši na osnovu JMBG, pošto nigde pa ni u podsistemu biblioteke se nemogu naći dve osobe istog matičnog broja. Jedna osoba nemože dva puta biti upisan u biblioteku. Tek potom se iz forme novog člana šalje poruka koja je zapravo konstruktor klase član. Tako formiran objekat se smešta i čuva u nekoj tabeli baze podataka koja se koristi.

Na sledećoj slici prikazan je sekvencijalni dijagram interfejsa dodavanja člana.



Slika 6.10. Sekvencijalni dijagram interfejsa dodavanja člana

Sekvencijalni dijagram interfejsa zaduživanja

I na ovom dijagramu prepliću se objekti klasa koji pretstavljaju korisnički interfejs i klasa koje opisuju logiku rada baze podataka. Međutim, to je neminovno jer je posledica pokretanja kontrola

koje se nalaze na maskama interfejsa kreiranje objekata softverskih klasa koji čuvaju trenutno stanje podsistema u pogledu materijalnog zaduženja. Na slici A.6.12. prikazan je sekvencijalni dijagram interfejsa zaduživanja.

Sekvencijalni dijagram interfejsa razduživanja

Kod razduživanja bibliotekar radi na formi za razduživanje, odakle je redosled radnji sledeći. Naslov se traži po inventarnom broju i tu se koristi forma za pretragu naslova. Zatim se pregledaju primerci tog naslova i kada se nađe primerak naslova, koji se razdužuje, te se na osnovu njega traži zaduženje. Kada se nađe zaduženje traži se i pribavlja član koji ga duži. Na osnovu tako uspostavljene veze i provere vrći se raskidanje zaduženja i njegovo uklanjanje iz tabele zaduženja. Na slici 6.13. prikazan je sekvencijalni dijagram interfejsa razduživanja

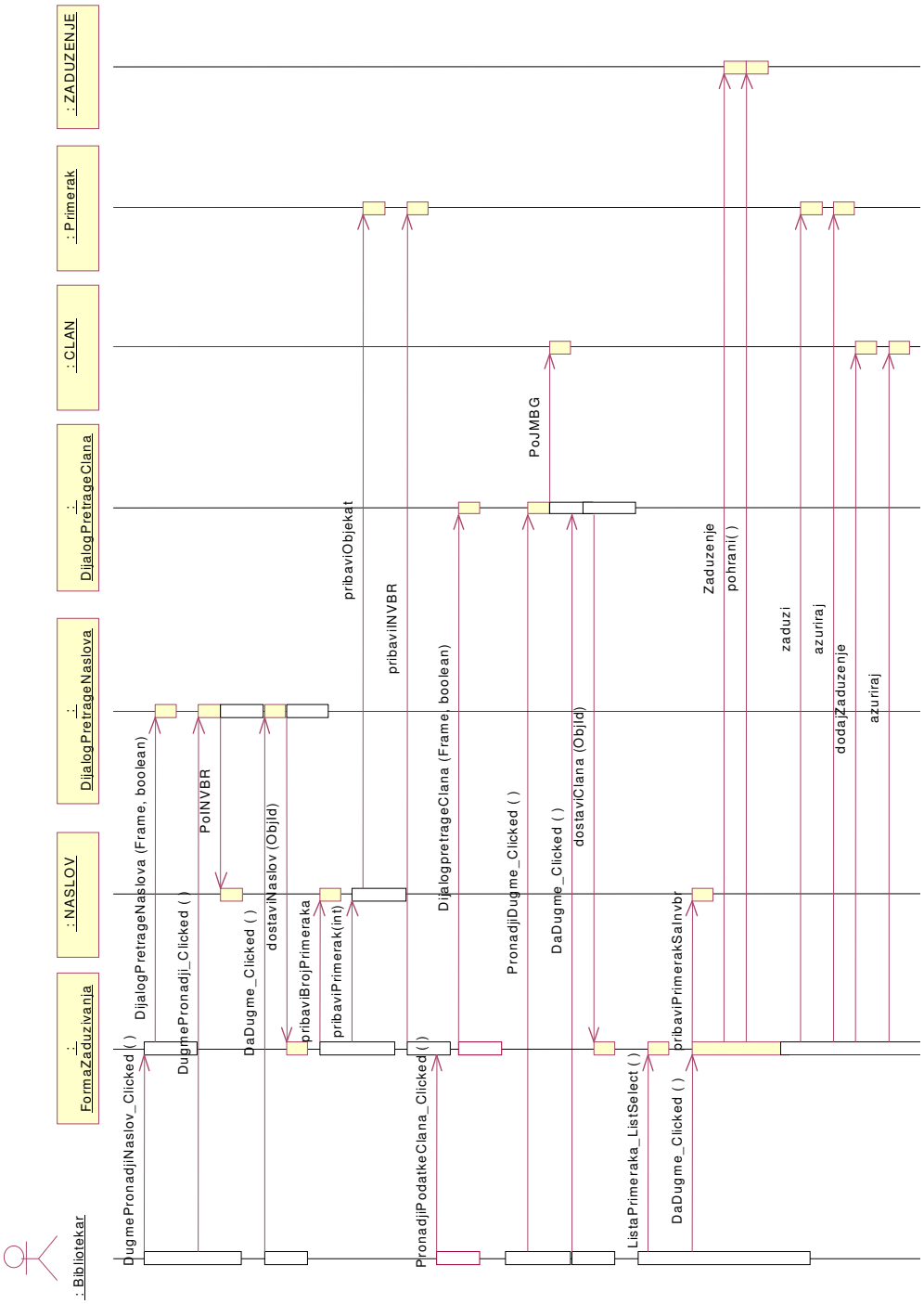
Sekvencijalni dijagram rezervisanja

Pri rezervisanju prvo se sa forme za rezervisanja poziva forma za pretraživanje naslova koji še se rezervisati. Naslov se pretražuje po inventarnom broju. Doduše može se naslov pretraživati i po drugim kriterijumima ali će samo ovakvo pretraživanje dati jedan i samo jedan naslov. Potom se preko forme za pretragu članova pretražuje član koji će rezervisati naslov. Kako je postavljeno u ovom dijagramu ovo pretraživanje člana se vrši po matičnom broju. Kao i kod naslova i člana moguće je pretraživati po još nekim pa i zbirnim kriterijumima, ali samo pretraživanje po matičnom broju daje jednoznačan rezultat pretrage. Nakon što se tako pronađe odgovarajući član i naslov za rezervaciju, porukom koja je zapravo konstruktor klase rezervacije kreira se objekat klase rezervacija. Taj objekat se pohranjuje i stanje sistema uvećano za jednu rezervaciju se uvećava. Na slici prik 6.14. prikazan je sekvencijalni dijagram rezervisanja.

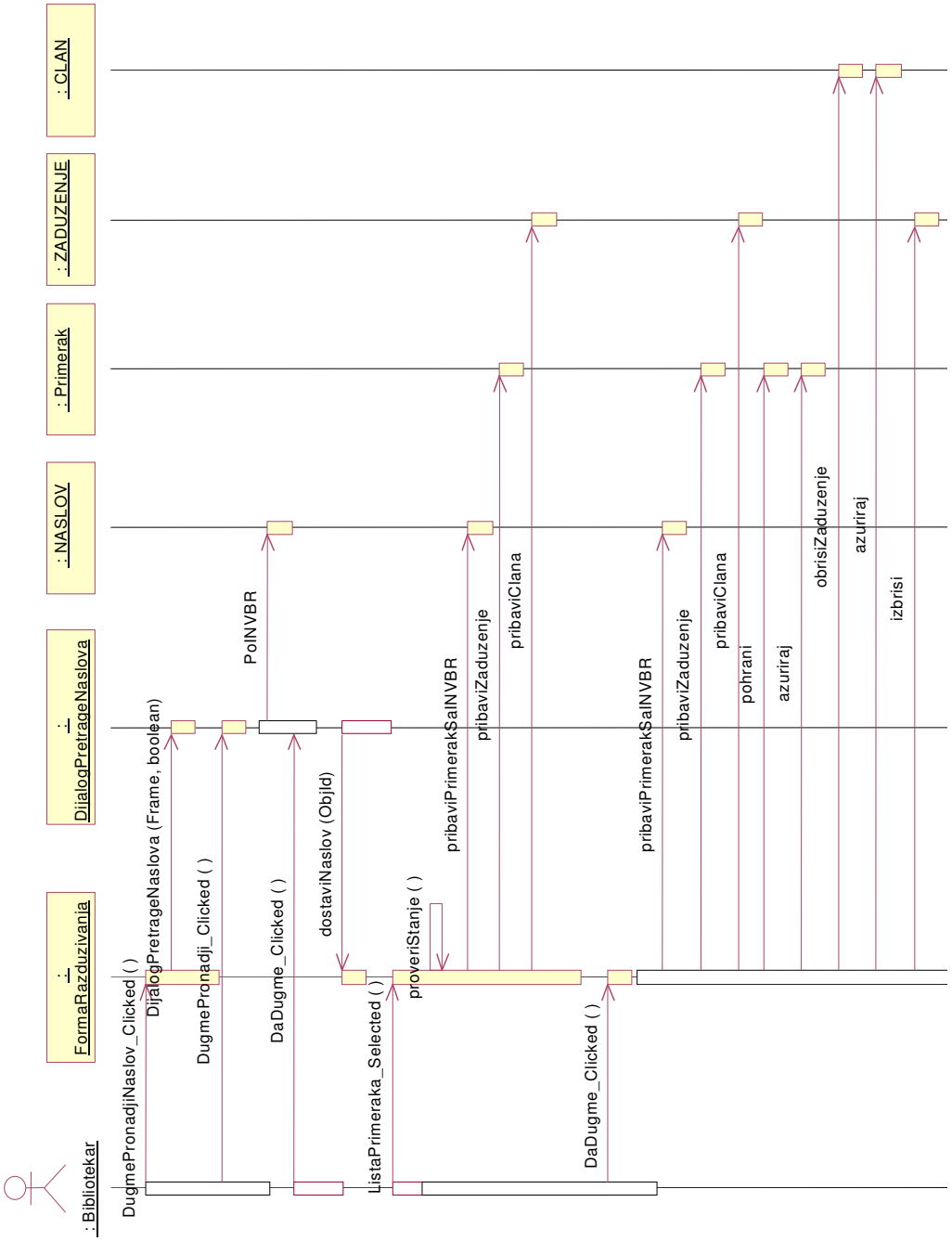
Sekvencijalni dijagram interfejsa brisanja rezervacije

Na sledećj slici daje se opis dešavanja prilikom brisanja postojeće rezervacije. Bibliotekar komunicira sa odgovarajućom formom za brisanje rezervacije i dijalogom za pretragu naslova. Na osnovu dobijenih rezultata vrše se intervencije nad stanjima objekata klasa naslov, član i rezervacija.

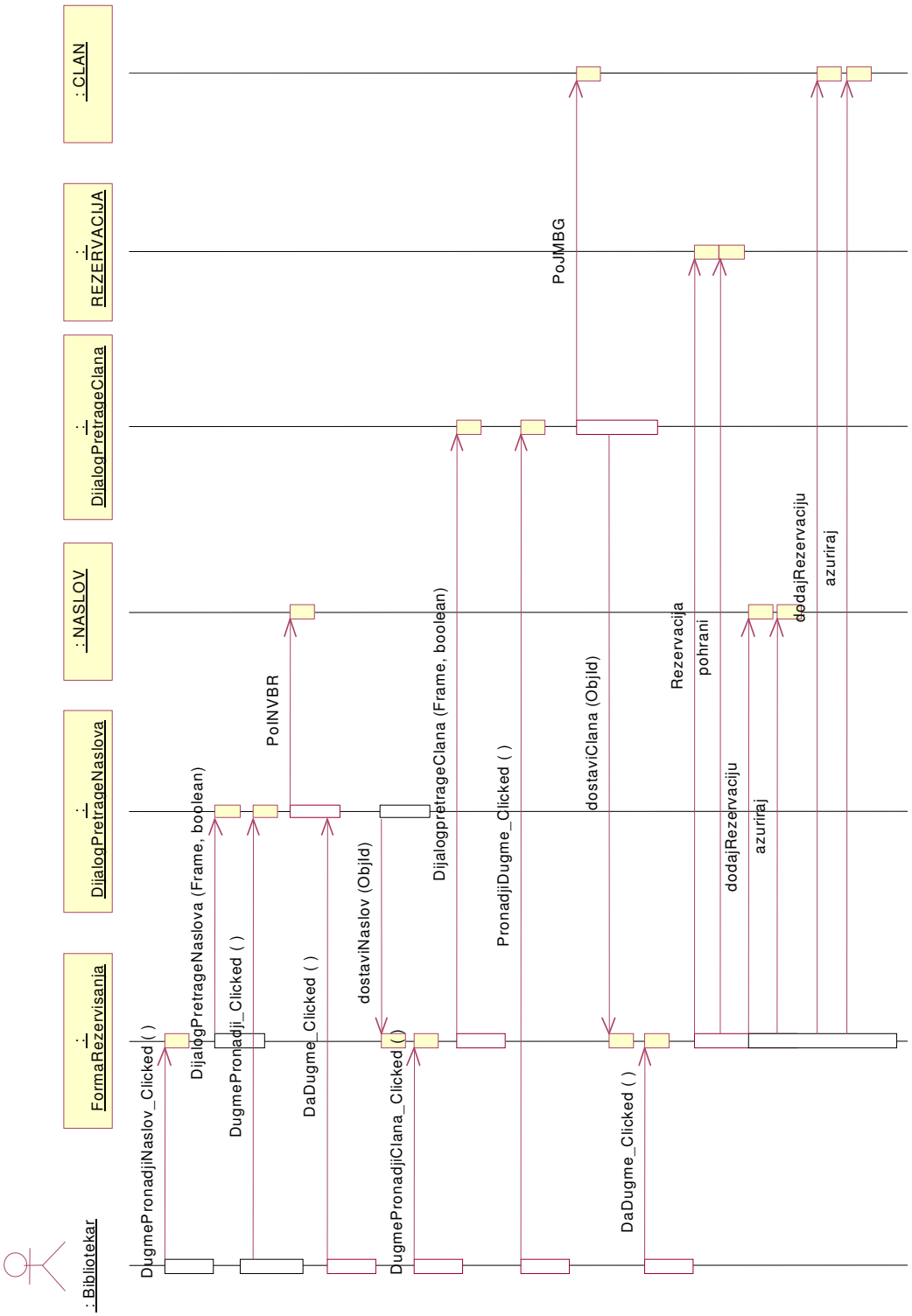
Ovakvim sekvencijalnim dijagramima moguće je opisati sve što se dešava na svim formama aplikacije bibliotečkog poslovanja. Smatrajući da su dosadašnjim primerima u dovoljnoj meri opisani mehanizmi bibliotečkog poslovanja i semantika i notacija sekvencijalnih dijagrama prostali sekvencijalni dijagrami koji su izrađeni za potrebe ovog projekta biće dati samo potpisani bez tekstualnog opisa. Radi se o dijagramima koji se odnose na funkcionisanje interfejsa za održavanje ažurnosti sistema. Preko korisničkog interfejsa koji je pomenut vrši dodavanje novih naslova, članova i primeraka ali i njihovo brisanje iz podsistema bibliotečkog poslovanja. Na slici 6.15. prikazan je sekvencijalni dijagram interfejsa brisanja rezervacije.



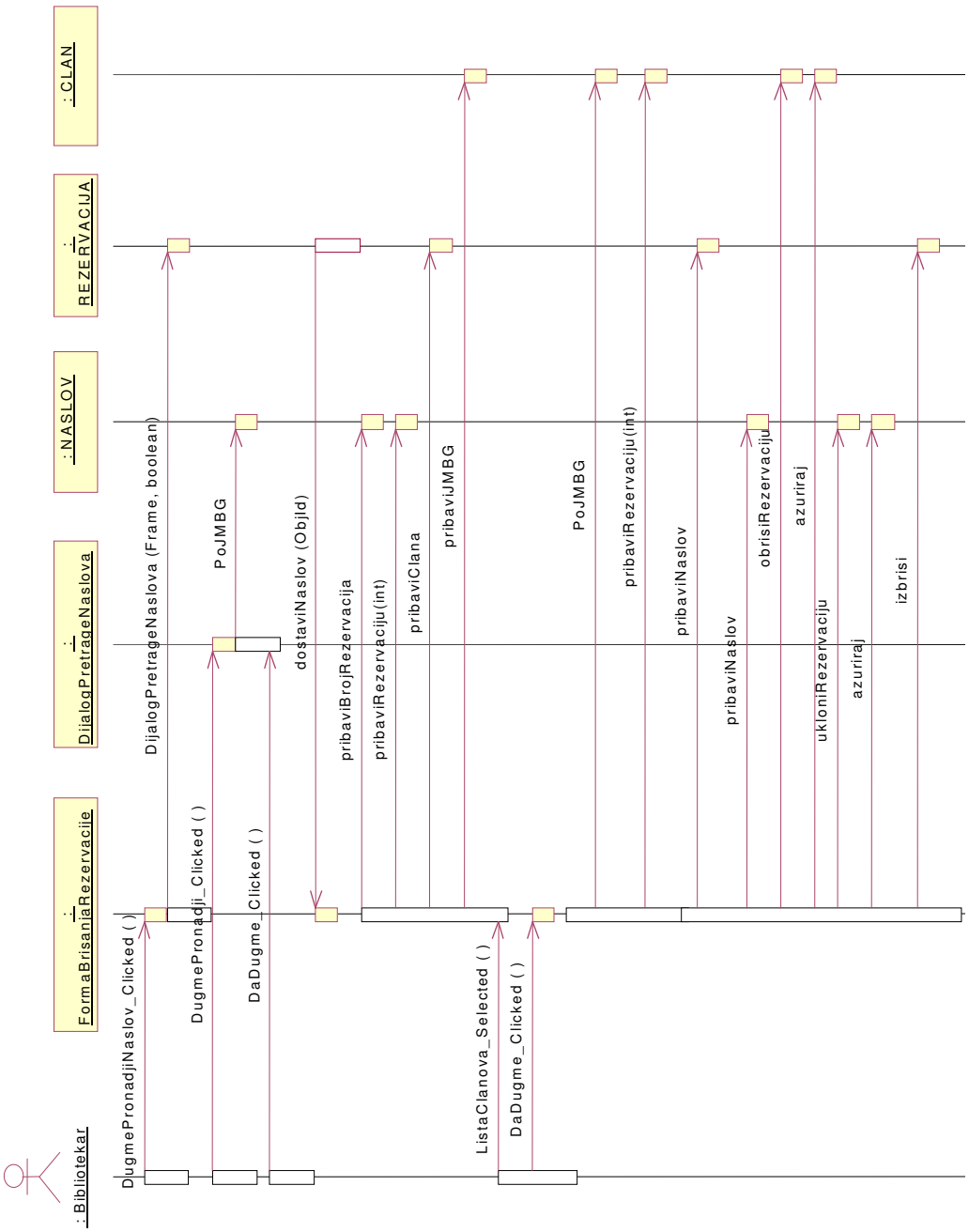
Slika 6.12. Sekvencijalni dijagram interfejsa zaduživanja



Slika 6.13. Sekvencijalni dijagram interfejsa razduživanja



Slika 6.14. Sekvencijalni dijagram rezervisanja



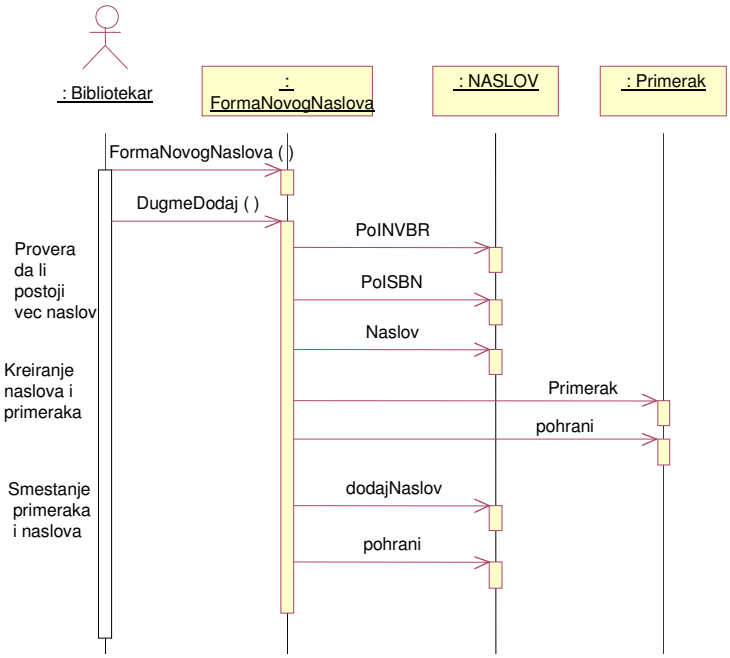
Slika 6.15. Sekvencijalni dijagram interfejsa brisanja rezervacije

Sekvencijalni dijagram interfejsa dodavanja naslova

Da bi se dodao novi naslov u evidenciju bibliotekar mora da pozivom konstruktora forme novog naslova otvori masku za unos podataka novog naslova. Kada se unesu svi podaci šalje se poruka izazvana događajem pritiska na da dugme. Potom se proverava da li takav naslov postoji imajući u vidu upisani inventarni ili ISBN broj. Ako takav naslov ne postoji poziva se konstruktor objekta klase naslov čije se stanje postavlja u zavisnosti od podataka novog člana. U zavisnosti od broja primeraka novog naslova određeni broj puta se poziva i konstruktor objekta klase primerka. Tek nakon toga se objekat novog naslova zaista i menmoriše u bazu podataka.

Kao što postoji vremenski redosled razmene poruka između objekata klasa interfejsa u slučaju dodavanja naslova, tako vremenski redosled poruka postoji i u slučaju brisanja naslova iz evidencije.

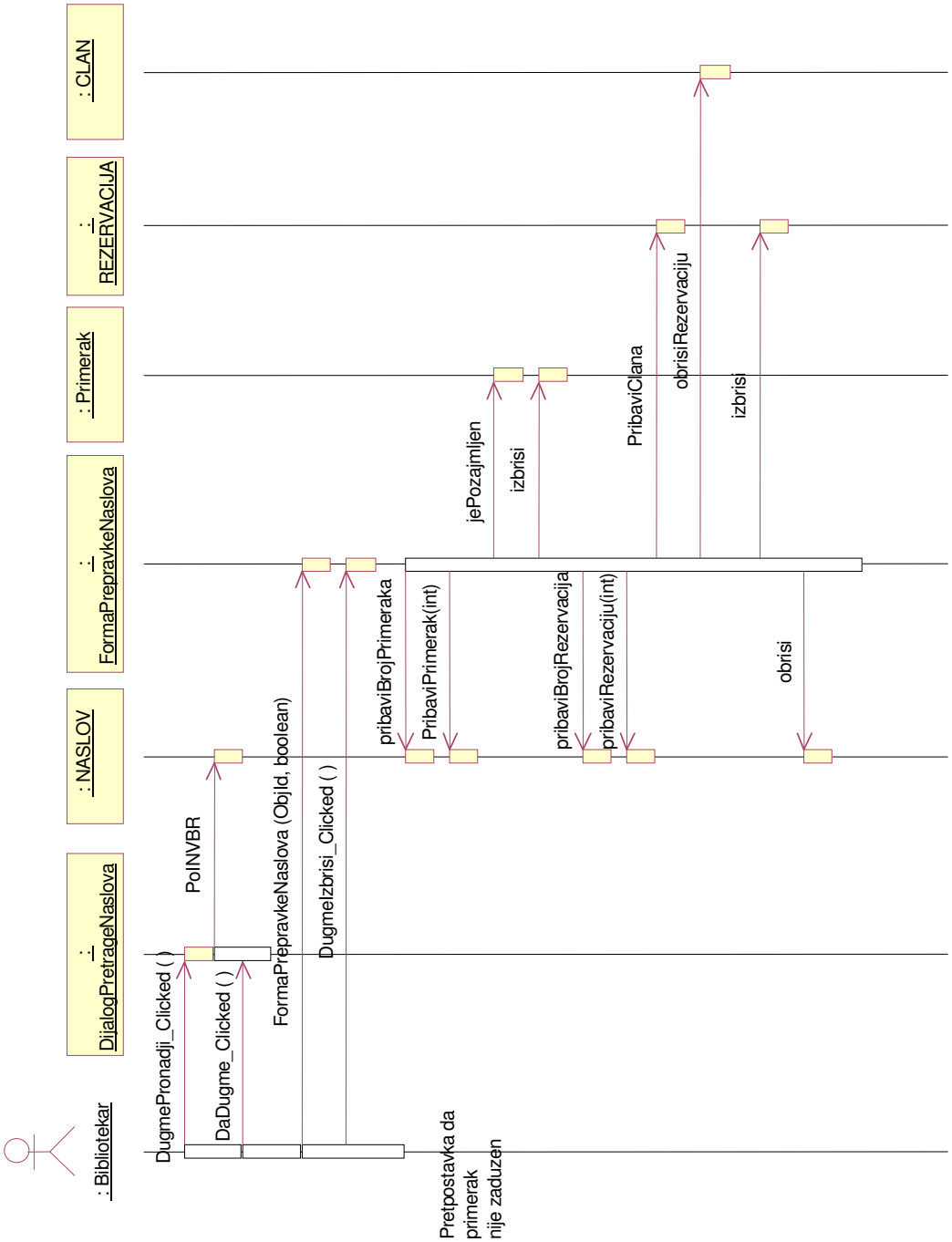
Na sledećoj prikazan je sekvencijalni dijagram interfejsa dodavanja naslova.



Slika 6.16. Sekvencijalni dijagram interfejsa dodavanja naslova

Sekvencijalni dijagram interfejsa brisanje naslova

Prvo se naslov koji se briše mora pronaći i zato se prvo poziva konstruktor klase dijaloga pretrage naslova. U taj dijalog se unosi kriterijum pretraživanja pa se u zavisnosti od njega naslov pretražuje (ovde je uzeto primera radi pretraživanje po inventarnom broju). Kada se pronađe naslov koji treba da se izbriše otvara se forma prepravke naslova, jer je i brisanje neka vrsta prepravke. Naslov se potom fizički briše i pribavljaju se njegovi primerci, proverava se da li su primerci zaduženi , ako nisu i ti se objekti brišu a bez ikakvog ograničenja uklanjaju se i sve rezervacije tog naslova. Na sledećoj slici prikazan je sekvencijalni dijagram interfejsa brisanje naslova.



Slika 6.17. Sekvencijalni dijagram interfejsa brisanje naslova

Sekvencijalni dijagram interfejsa dodavanja primerka

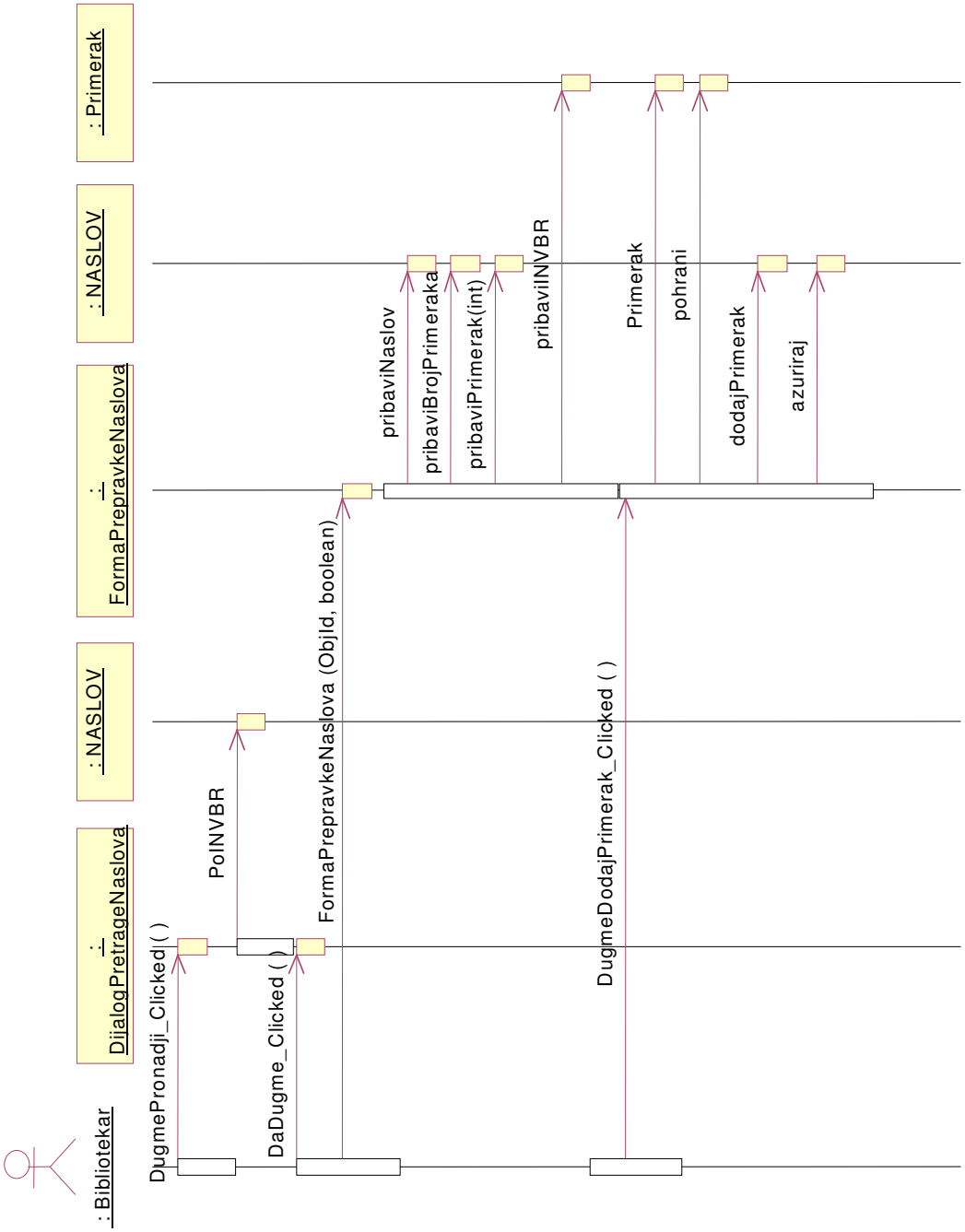
U realnom sistemu je za očekivati takvu situaciju da će za već zavedeni naslov pristizati novi primerci. Prvo se mora pronaći naslov za koji se dodaju primerci i zato se inicira forma pretrage naslova. Naslov se potom pretražuje po nekom kriterijumu (recimo inventarni broj) i za pronađeni naslov se otvara forma prepravke naslova. Iz ove forme se dolazi do podatka o postojećem broju primeraka i ostalih činjenica od značaja za dodavanje novog primerka naslova.

Na slici 6.18. prikazan je sekvencijalni dijagram interfejsa dodavanja primerka.

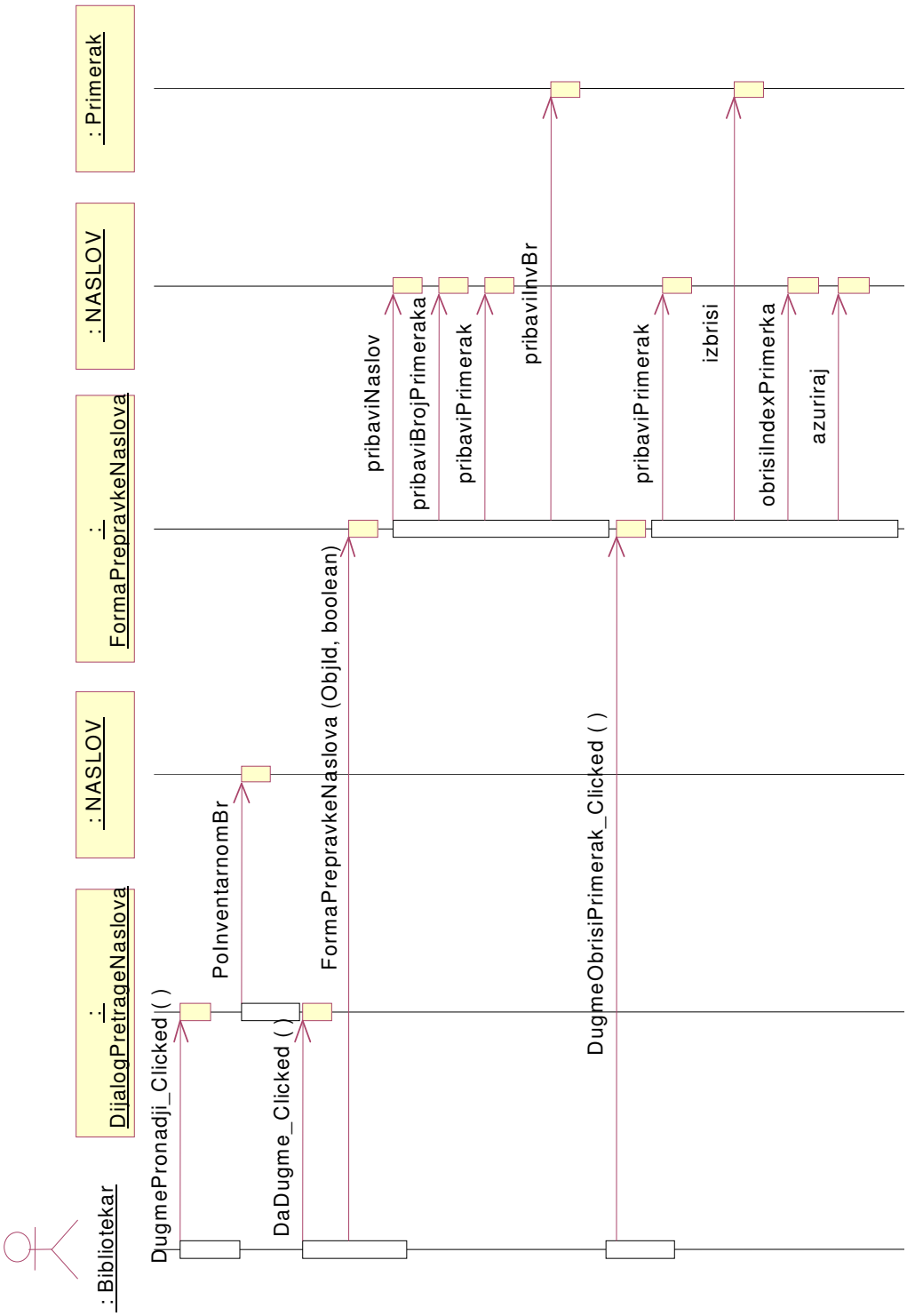
Sekvencijalni dijagram interfejsa brisanja primerka

Inverzna operacija od gore opisane je naravno, operacija brisanja primerka naslova. Veoma je sličan scenario odvijanja razmene poruka između objekata klasa u interakciji sa onim opisanim u prethodnim slučajevima. Mora se otvoriti forma pretrage naslova kako bi se našao naslov kojem se brišu primerci. Za pronađeni naslov se otvara dijalog prepravke naslova, iz nje se pribavljaju postojeći primerci naslova i odabrani se uklanjaju iz evidencije bibliotečkog podsistema.

Na slici 6.19. prikazan je sekvencijalni dijagram interfejsa brisanja primerka.



Slika 6.18. Sekvencijalni dijagram interfejsa dodavanja primerka

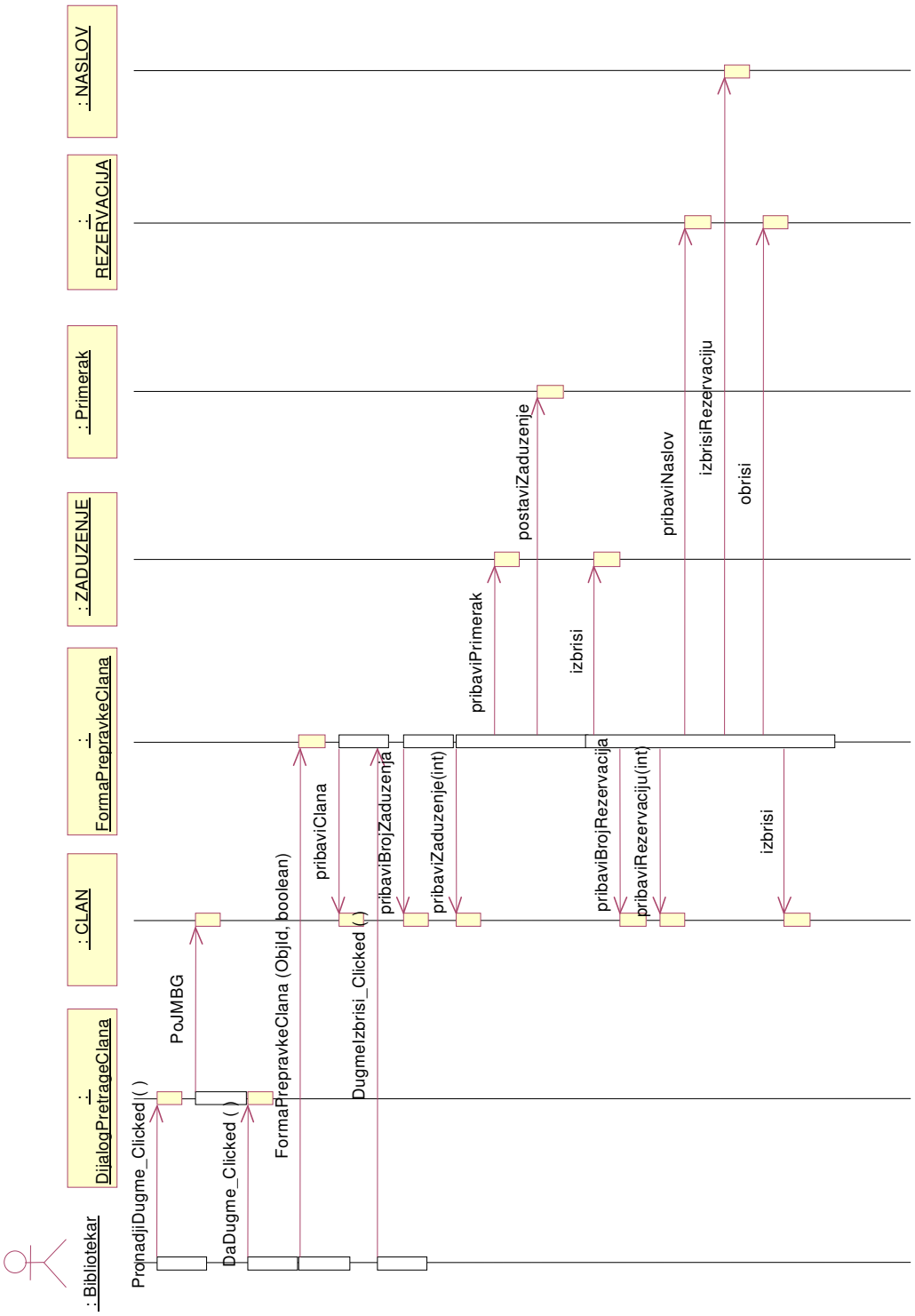


Slika 6.19. Sekvencijalni dijagram interfejsa brisanja primerka

Sekvencijalni dijagram interfejsa brisanja člana

Na kraju je ostao da se opiše još sekvencijalni dijagram interfejsa za brisanje člana biblioteke. Član čiji se podaci uklanjaju iz podsistema mora se prvo pronaći i zato se prvo radi sa dijalogom za pretragu člana. Iz njega se član pretražuje po zadatim kriterijumima (najčešće po jedinstvenom matičnom broju građanina). Za pronađenog člana biblioteke otvara se forma prepravke člana, s obzirom da je i brisanje svojevrsna promena stanja objekta klase člana. Za člana se pribavlja spisak primeraka naslova koje duži kako i spisak njegovih rezervacija. U načelu je zabranjeno izbrisati člana koji nije u potpunosti razdužen sa primercima iz bibliotečkog fonda, a to zavisi od usvojene politike poslovanja koja je podložna ponekad i isuviše čestim promenama. Bilo kako bilo, rezervacije koje je ostvario član u toku svog životnog veka u bibliotečkom podsistemu se uklanjaju bez ikakvih ograničenja. Indikativno je i sa stanovišta programerske logike očekivano da se prvo pozivaju destruktori sa članom povezanih objekata klasa zaduženja i rezervacija. pre nego što se pozove i destruktor konkretnog objekta klase člana biblioteke.

Na sledećoj slici prikazan je sekvencijalni dijagram interfejsa brisanja člana.



Slika 6.20. Sekvencijalni dijagram interfejsa brisanja clana

Definisanje ugovora o izvršenju operacija

Ugovori o izvršenju operacija definišu efekte koje operacije treba da imaju po stanje sistema. Uobičajno je predstavljanje operacija u vidu formalnog izlaganja opisa stajna u kojima se sistem nalazi pre i posle izvršenja operacija. Ugovore je moguće definisati na veoma niskom nivou, pojedinačnih metoda svake softverske klase, ali i na nivou uopštenih sistemskih operacija. S obzirom da se razvoj informacionog sistema bibliotečkog poslovanja još uvek nalazi u fazi analize kada treba dati ogovor na pitanje šta sistem treba da radi, ovde će biti specificirani neki ugovori o izvršenju operacija sistema, posmatranog kao celina. Dakle radi se o formalnim opisima stanja sistema nakon izvršenja operacija čija je realizacija u neku ruku data prethodnom izradom sekvencijalnih dijagrama.

Ime operacije: *Dodavanje člana*

Odgovornosti: Upisuje podatke novog člana u evidenciju podsistema

Izuzeci: Član tog matičnog broja već postoji ; Nisu uneti svi obavezni podaci člana

Preduslov: Šifarnici atributa člana su ažurni

Post uslov: Kreiran je novi objekat klase član, vezan je za određene reference kategorije članstva(student, zaposlen ili vanredni), te u zavisnosti od kategorije za identifikatore studentske klase, službe, jedinice ili ustanove.

Ime operacije: *Brisanje člana*

Odgovornosti: Uklanjanje zapisa podataka člana iz evidencije podsistema.

Izuzeci: Član ne postoji, član još uvek ima zaduženja

Preduslov: Član postoji i razdužio je sve naslove koje je imao na pozajmici

Post uslov: Uništen je konkretni primerak klase člana, i izbrisane su sve njegove rezervacije koje postoje u podsistemu.

Ime operacije: *Dodavanje naslova*

Odgovornosti: Unosi podatke novog naslova u evidenciju podsistema

Izuzeci: Naslov tog inventarnog broja ili signature već postoji; Nisu uneti svi obavezni podaci naslova.

Preduslov: Svi šifarnici atributa naslova su ažurni

Post uslov: Kreiran je novi objekat klase naslov, povezan je sa odgovarajućim identifikatorima jezika i udk oblasti.

Ime operacije: *Brisanje naslova*

Odgovornosti: Uklanja podatke naslova iz evidencije podsistema

Izuzeci: Naslov ne postoji

Preduslov: Naslov mora da postoji

Post uslov: Uništen je primerak klase naslov, i sve njegove rezervacije, zaduženja i opomena.

Ime operacije: *Zaduživanje*

Odgovornosti: Zadužuje člana biblioteke sa konkretnim primerkom naslova

Izuzeci: Član ili naslov ne postoji; svi primerci naslova su zaduženi ili rezervisani

Preduslov: Član i naslov postoje i ima slobodnih primeraka naslova

Post uslov: Kreiran je novi objekat klase zaduženje sa pripadajućim atributima

Ime operacije: *Razduživanje*

Odgovornosti: Razdužuje člana biblioteke sa primerakom naslova

Izuzeci: Ne poklapaju se inventarni brojevi primerka naslova koji je zadužen i primerka koji pokušava da se razduži

Preduslov: zaduženje postoji

Post uslov: Uništen je konkretni primerak klase zaduženja, i kreiran je odgovarajući objekat klase razduženja.

Ime operacije: *Rezervisanje*

Odgovornosti: Član rezerviše naslov koji mu nije trenutno dostupan ili ne želi da ga pozajmi odmah, već u narednom periodu.

Izuzeci: Član ili naslov ne postoji; član je već rezervisao naslov

Preduslov: Član koji rezerviše i naslov koji se rezerviše moraju da postoje.

Post uslov: Kreiran je odgovarajući objekat klase rezervacije

Ime operacije: *Brisanje rezervacije*

Odgovornosti: Ukida se rezervacija člana biblioteke u odnosu na neki naslov iz bibliotečkog fonda.

Izuzeci: Rezervacija ne postoji

Preduslov: Prošlo je određeno vreme nakon što je primerak naslova oslobođen a rezervacija nije prerasla u zaduženje

Post uslov: uklonjen je objekat klase rezervacije iz liste rezervacije. Redosled objekata u FIFO listi rezervacija je ažuriran.

Objektno orjentisan dizajn za poslove cirkulacije u biblioteci

U okviru Objektno orjentisan dizajn poslova u biblioteci treba da se odogovor na pitanje kako izvršiti proces logičkog i fizičkog dekomponovanja sistema na manje softverske celine i blokove, i pritom izvršiti specifikaciju statičkih i dinamičkih programskih celina.

U ovoj fazi razmatraće se:

- Izrada dijagrama saradnje poslova cirkulacije u biblioteci
- Izrada potpunog dijagrama klasa za poslove cirkulacije u biblioteci
- Izrada dijagrama stanja poslova cirkulacije u biblioteci
- Definisanje paketa

U daljem tekstu detaljno će se opisati gore definisane aktivnosti.

Izrada dijagrama saradnje poslova cirkulacije u biblioteci

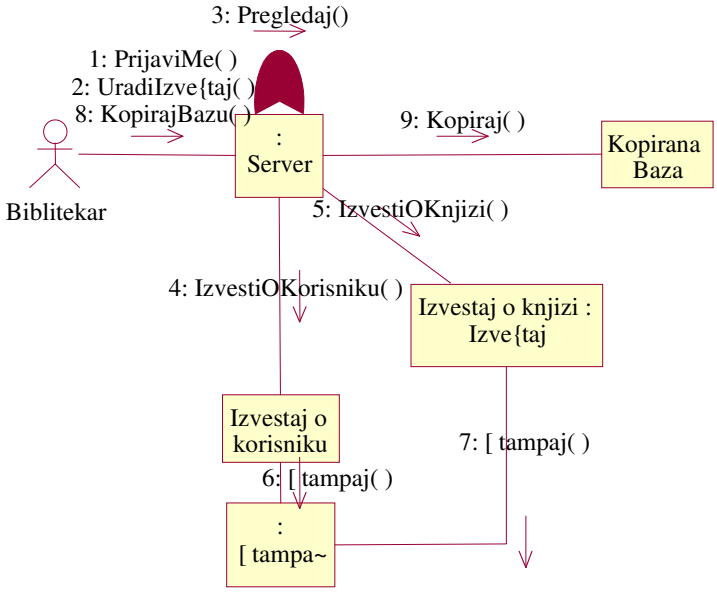
Kao što smo već rekli u poglavlju vezanom za objektno orjentisanu analizu i dijagrami saradnje spadaju u interakcione dijagrame. Dijagramima saradnje se modeluju tokovi kontrole po organizaciji. Takvo modelovanje naglašava strukturne odnose objekata u interakciji, koji razmenjuju poruke. U odnosu na dijagrame sekvenci dijagrami saradnje(kolaboracije) su mnogo zgodniji za opis više paralelnih tokova kontrole. Objekti su na dijagramu saradnje prikazani kao čvorovi nekog grafa između kojih teku veze kojima se kreću poruke. Značaj dijagrama saradnje je u tome što se njima na neki način predefinišu metode (funkcije članice) koje će se pojavljivati u dijagramu klasa. Svaka od operacija članica klase na neki način je satavljena od poruka koje razmenjuju objekti u saradnji. Na sledećoj slici je dat dijagram saradnje za izveštavanje i backup-ovanje.

Dijagram saradnje za izveštavanje i backup-ovanje

S obzirom da je smisao odvijanja poruka prethodnog dijagrama objašnjen na istom slučaju ali za sekvencijalan dijagram, ovde će se samo skrenuti pažnja na neke momente koji nisu mogli biti prikazani notacijom u vremenskom domenu. Ovde se može videti da su nakon slanja

odgovarajućih poruka (kopiraj bazu, uradi izveštaj) serveru moguća dva paralelna toka. Moguće je i kopirati i bazu i uraditi izveštaj i za korisnika i za naslov. Taj momenat nije mogao da dođe do izražaja kod opisa pomoću sekvencijalnog dijagrama. I na ovom dijagramu kao i na sekvencijalnom, a možda i još jasnije, postaje jasno da poruka štampaj može uslediti tek nakon odgovarajućih poruka za kreiranjem izveštaja. Obe poruke se stiču u jednom čvoru grafa, primerku klase štampač.

Na sledećoj slici prikazan je dijagram saradnje za izveštavanje i backup-ovanje.

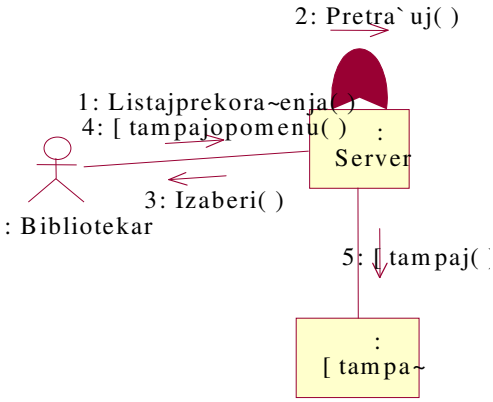


Slika 6.21. Dijagram saradnje za izveštavanje i backup-ovanje

Dijagram saradnje za opominjanje

Dijagram koji se nalazi na sledećoj slicic znatno je jednostavniji od prethodnog zato što ima manje objekta u kolaboraciji pa je samim tim i interakcija između njih prostija. Brojevi koji su dodati u odnosu na prethodni dijagram predstavljaju numeraciju poruka po vremenskom redosledu, koja je preuzeta iz odgovarajućeg sekvencijalnog dijagrama. Nema paralelnih tokova kontrole kao u prethodnom slučaju. Logika kojom se odvija slanje poruka opisana je u delu posvećenom sekvencijalnom dijagramu iste funkcije informacionog podsistema bibliotečkog poslovanja.

Na sledećoj slici prikazan je dijagram saradnje za opominjanje.

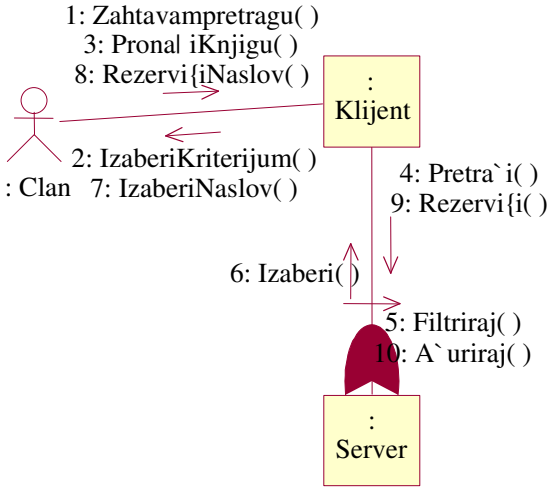


Slika 6.22. Dijagram saradnje za opominjanje

Dijagram saradnje za rezervisanje

Dijagram saradnje Na sledećoj slici ilustruje razmenu poruka između objekata klasa. Nema pralelnih tokova kontrole. Poruke se između objekata razmenjuju po redosledu brojeva koji ih numerišu i to na trasi član-klijent-server.

Član biblioteke preko računara razmenjuju poruke sa serverom gde se nalaze podaci neophodni za obavljanje funkcije rezervisanja naslova. Smisao odvijanja slanja poruka već je opisan u odgovarajućem sekvencijalnom dijagramu u poglavlju objektno orjentisane analize.



Slika 6.23. Dijagram saradnje za rezervisanje

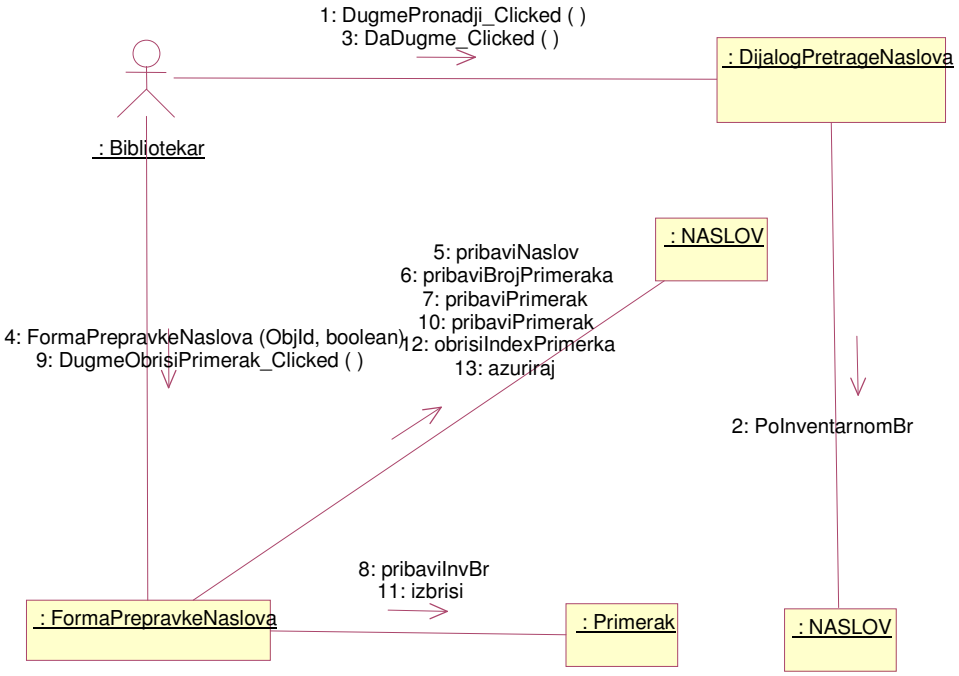
Izrada dijagrama saradnje interfejsa

Na isti način kao što su definisani sekvencijalni dijagrami za korisnički interfejs različitih funkcija podsistema, isto tako mogu biti urađeni i kolaboracioni dijagrami interfejsa. Činjenica da su već uradjeni sekvencijalni dijagrami istih pojava taj zadatak čini samo lakšim, zbog semantičke jednakosti ove dve podvrste interakcionih dijagrama. Jedina razlika, što je već naglašeno, je u pogledu na saradnju koju objekti interfejsnih klasa ostvaruju razmenjujući poruke. Za razliku od sekvencijalnih dijagrama klasa interfejsa kolaboracioni dijagrami interfejsa naglašavaju strukturu objekata koji komuniciraju i veza između njih. Pored očekivanih interfejsnih klasa ovde će se pojavljivati i neke poslovne klase, tako da ni ovi kao ni sekvencijalni dijagrami neće biti u potpunosti samo interfejsni. No to je i za očekivati jer granica između interfejsnog sloja i sloja logike aplikacije nije jasno povučena, pogotovu kada se uzmu u obzir interakcioni dijagrami, koji se zasnivaju na saradnji objekata različitih klasa pa makar one bile i iz različitih slojeva. Uostalom normalno je da obrasci i poslovni objekti međusobno razmenjuju poruke- na tome se zasniva svaka aplikacija u grafičkom okruženju.

Dijagram saradnje interfejsa brisanja primeraka

Prateći numeraciju koja označava redosled slanja poruka u komunikaciji koja se ostvaruje pri brisanju postojećeg primerka nekog naslova može se rekonstruisati saradnja intrfejsa i objekata poslovnih klasa. Bibliotekar koji vrši brisanje primerka pritiska dugme pronađi i time pretražuje naslov po inventarnom broju (mogući su i drugi kriterijumi pretrage). Izbor tog naslova se potvrđuje, nakon čega se otvara forma za prepravku naslova pozivom njenog konstruktora (brisanje primerka je zapravo menjanje jednog podatka naslova). Potom se pribavlja naslov sa svim pratećim podacima, od kojih je od posebnog interesa broj primeraka jednog naslova. Tako dobijeni primerak se uklanja a stanje ažurira. Time je izvršeno brisanje primerka naslova.

Na sledećoj slici prikazan je dijagram saradnje interfejsa brisanja primeraka

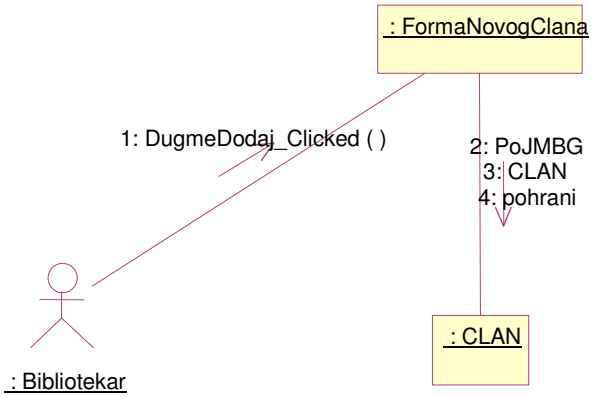


Slika 6.24. Dijagram saradnje interfejsa brisanja primeraka

Dijagram saradnje interfejsa dodavanje člana

Dijagram saradnje interfejsa za upis novog člana biblioteke je krajnje jednostavan kao i njegov parnjak iz grupe sekvencijalnih dijagrama. Na pritisak dugmeta na formi za dodavanje novog člana pre nego što se pozove konstruktor člana proverava se po matičnom broju da takav član već ne postoji. Jedna osoba može samo jednom biti učlanjena u biblioteku.

Na sledećoj slici prikazan je dijagram saradnje interfejsa dodavanje člana.

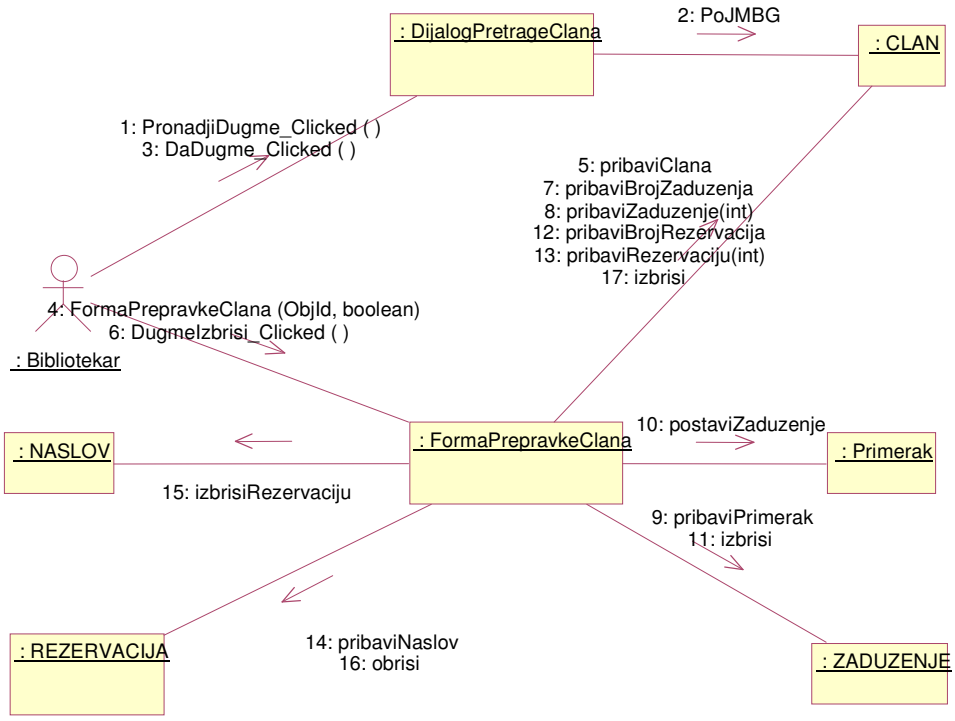


Slika 6.25.Dijagram saradnje interfejsa dodavanje člana

Dijagram saradnje interfejs brisanja člana

Da bi se član izbrisao mora se prvo pretražiti. Zato je u saradnji potrebna forma za pretragu člana, potom se poziva forma za prepravku člana jer je brisanje člana na neki način njegova prepravka. Interesantno je da se na osnovu numeracije može odmah reći da pre nego što se izbriše član pribavljaju se njegova zaduženja i njegove rezervacije pa se i one uklanjaju. Tek nakon brisanja rezervacija i zaduženja vrši se i uklanjanje objekta klase člana.

Na sledećoj slici prikazan ke dijagram saradnje interfejs brisanja člana.

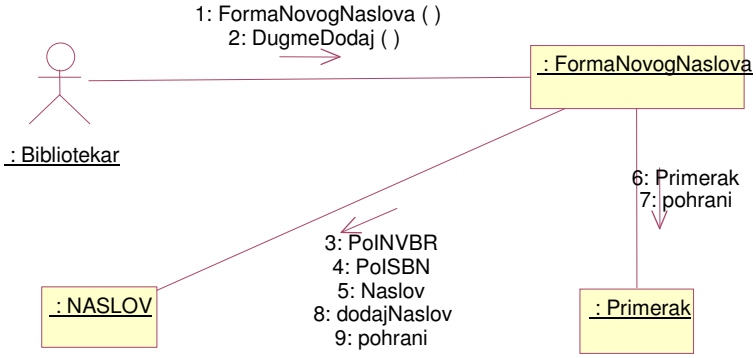


Slika 6.26. Dijagram saradnje interfejs brisanja člana

Dijagram saradnje interfejsa dodavanja naslova

U slučaju kada se uvodi novi naslov (pristigla je nova knjiga), bibliotekar sa sistemom komunicira preko forme novog naslova. I ovde se vrši standardna provera da ne postoji već naslov istog inventarnog broja. Interesantno je da se kreira objekat klase naslova pa nakon toga i odgovarajući primerci naslova u slučaju da naslov ima više primeraka. Tek nakon toga vrši se smeštanje novog naslova u podsistem bibliotečkog poslovanja na .

Na sledećoj slici prikazan je dijagram saradnje interfejsa dodavanja naslova.

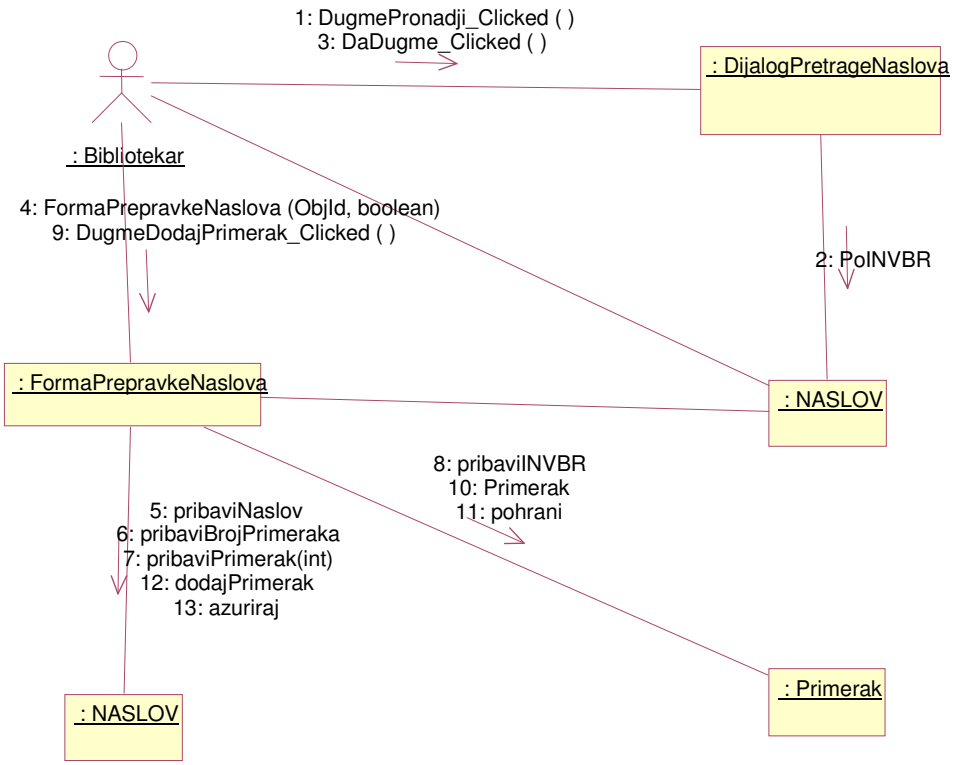


Slika 6.27. Dijagram saradnje interfejsa dodavanja naslova

Dijagram saradnje interfejsa dodavanja primerka

U poslovanju biblioteke dešava se da naknadno stignu novi primerci naslova koji već postoji zaveden u podsistemu. Tada je potrebno zavesti te primerke , a to nije isto kao unos novog naslova. Zato za tu operaciju postoji posebna forma prepravke naslova, jer je unošenje novih primeraka u naslov promena podataka naslova. No, pre toga potrebno je putem dijaloga za pretragu naslova pronaći naslov, bibliografsku jedinicu za koju se unose novi primerci. Naslov se pretražuje, po pravilu, po inventarnom broju, mada nije obavezno, ali je preporučljivo. Kada se pronađe odgovarajući naslov dobavlja se njegov broj primeraka kako bi se znalo koje redne brojeve dodeliti novopridošlim primercima. Tek tada se za poznate podatke poziva konstruktor primerka i nastaje novi objekat klase primerka.

Na sledećoj slici prikazan je dijagram saradnje interfejsa dodavanja primerka.

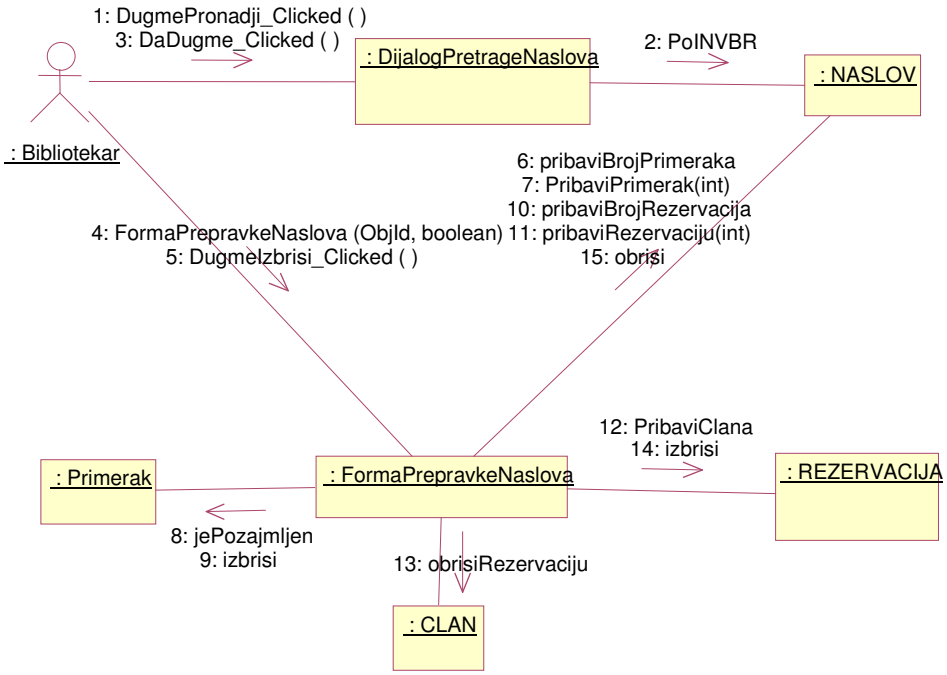


Slika 6.28. Dijagram saradnje interfejsa dodavanja primerka

Dijagram saradnje interfejsa brisanja naslova

U skladu sa usvojenom politikom poslovanja definisani uslovi kada može doći do rashodovanja nekog naslova. Definisanje tih uslova ne spada u domen informacionog sistema, on samo izvršava brisanje naslova kada se za to donese prethodna odluka. I ovu funkciju obavlja bibliotekar preko korisničkog interfejsa aplikacije. Naravno radi se sa maskom koja je dijalog pretrage naslova, koji se želi obrisati. Na osnovu dobijenog rezultata pretrage poziva se forma za prepravku naslova, pošto se brisanje naslova može smatrati njegovom svojevrsnom prepravkom. Za svaki naslov se pribavljaju primerci i rezervacije, kako bi se prvo oni uklonili pre nego što se obriše i sam naslov. Kada bi se obrisao naslov pre primeraka i rezervacija tada se primercima i rezervacijama ne bi više moglo pristupiti i oni bi za navek ostali u podsistemu i time vremenom zauzimali sve veći i veći memorijski prostor. Primećuje se da brisanju naslova ne prethodi na isti način i brisanje zaduženja, iz čega se može zaključiti da se naslov ne može rashodovati ukoliko je kod nekog člana na zaduženju.

Na sledećoj slici prikazan je dijagram saradnje interfejsa brisanja naslova.

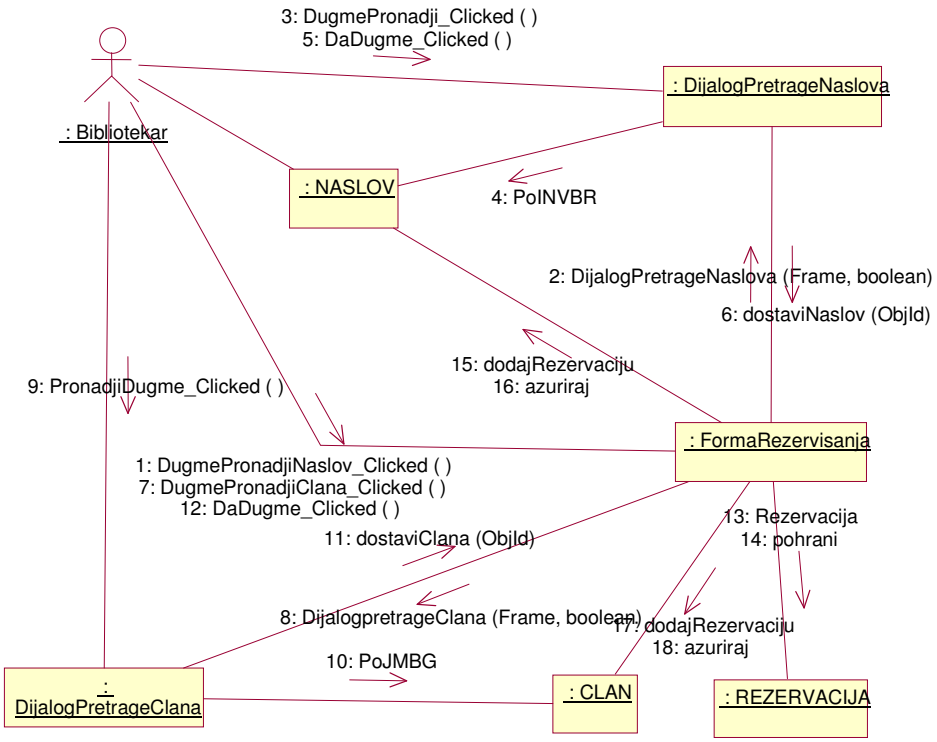


Slika 6.29. Dijagram saradnje interfejsa brisanja naslova

Dijagram saradnje interfejsa rezervisanja

Za uspostavljanje valjane rezervacije nekog naslova potrebni su validni podaci člana i naslova koj se združuju u rezervaciji. Zato se u dijagramu saradnjepojavljuju primerci klasa koje predstavljaju dijaloge za pretragu naslova i člana biblioteke. Pretrage se vrše u načelu po inventarno, odnosno matičnom broju, ali ne i obavezno. Veliku ulogu u obrazovanju i povećanju uslužnosti informacionog sistema trebalo bi da odigra formiranje stručnog kataloga po UDK klasifikaciji. Član bi onda na osnovu svog interesovanja mnogo bolje koristio fond biblioteke . Nakon što se obave konkurentni tokovi ka pretrazi naslova i člana, dobijemi podaci se združuju u jednoj formi rezervacije, odakle se šalju poruke ka tri strane. To se podaci o pohranjenoj rezervaciji smeštaju u objekte klasa član, naslov i rezervacija. Može se primetiti da u saradnji pri rezervisanju ne učestvuje klasa primerka, a razlog tome je, kao što je već pomenuto, što se ne rezerviše konkretan primerak već naslov bez obzira na pojavu.

Na sledećoj slici prikazan je dijagram saradnje interfejsa rezervisanja.

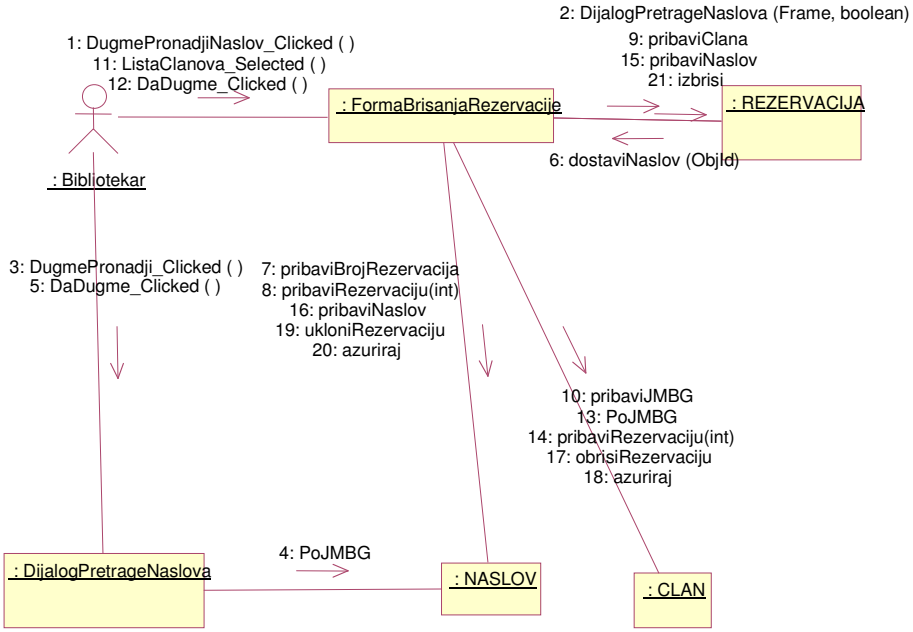


Slika 6.30. Dijagram saradnje interfejsa rezervisanja

Dijagram saradnje interfejsa brisanja rezervacije

Pošto je na sledećoj predstavljen dijagram saradnje za rezervisanje naslova, prirodno je da se predvidi i mogućnost sistema da obriše jednom postavljenu rezervaciju. Rezervacija se uklanja tako što se brišu podaci o njoj iz objekata klasa naslov, član i rezervacija.

Prvo se preko dijaloga za pretragu naslova pronađe naslov za koji se želi obrisati rezervacija i time se dobija lista članova koji su rezervisali taj naslov. Sa ponuđene liste bira se željeni član te se pribavlja njegov matični broj. Tako se dolazi do objekata iz koji se treba izbrisati podatak o postavljenoj rezervaciji. Jedino se odgovarajući objekat klase rezervacija briše u potpunosti, tačnije poziva se njegov destruktor.

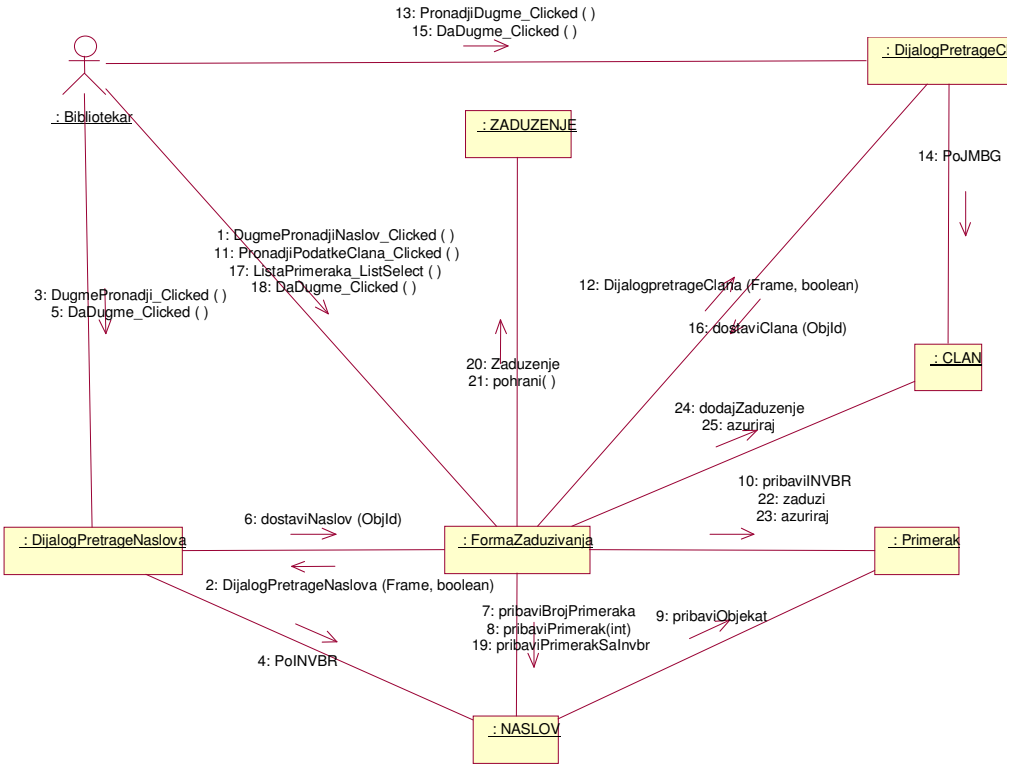


Slika 6.31. Dijagram saradnje interfejsa brisanja rezervacije

Dijagram saradnje interfejsa zaduživanja

U ovom dijagramu saradnje figuriraju objekti poslovnih klasa naslova, primerka, člana i zaduženja, ali i interfejsnih klasa koje predstavljaju forme aplikacije zadužene za pretraživanje naslova, člana i formiranje zaduženja. U početku se šalje poruka objektu klase forme zaduživanja da pronađe naslov, na šta se otvara forma za pretragu naslova. Sada se odvija sekvenca poruka koja je već viđena u ranijim dijagramima saradnje kada su se pretraživali naslovi. Za dobijeni naslov pribavlja se broj primeraka naslova i odlučuje se za neki konkretni, jer se zadužuje konkretan primerak naslova. Potom se objektu klase forme zaduživanja šalje poruka za pretragu člana koji će zadužiti primerak naslova. Na prijem te poruke otvara se dijalog pretrage člana odakle se pretražuje željeni član po matičnom broju. Kada su pribavljeni svi podaci potrebni za zaduženje naslova, poziva se konstruktor klase zaduženje i tako kreirani objekat se pohranjuje u informacioni podsistem. Sada je zaduženje formirano i ostalo je samo obezbediti inverzni proces, operaciju razduživanja zaduženog naslova. Sledi dijagram saradnje interfejsa za razduživanje zaduženog naslova.

Na sledećoj slici prikazan je dijagram saradnje interfejs zaduživanja.

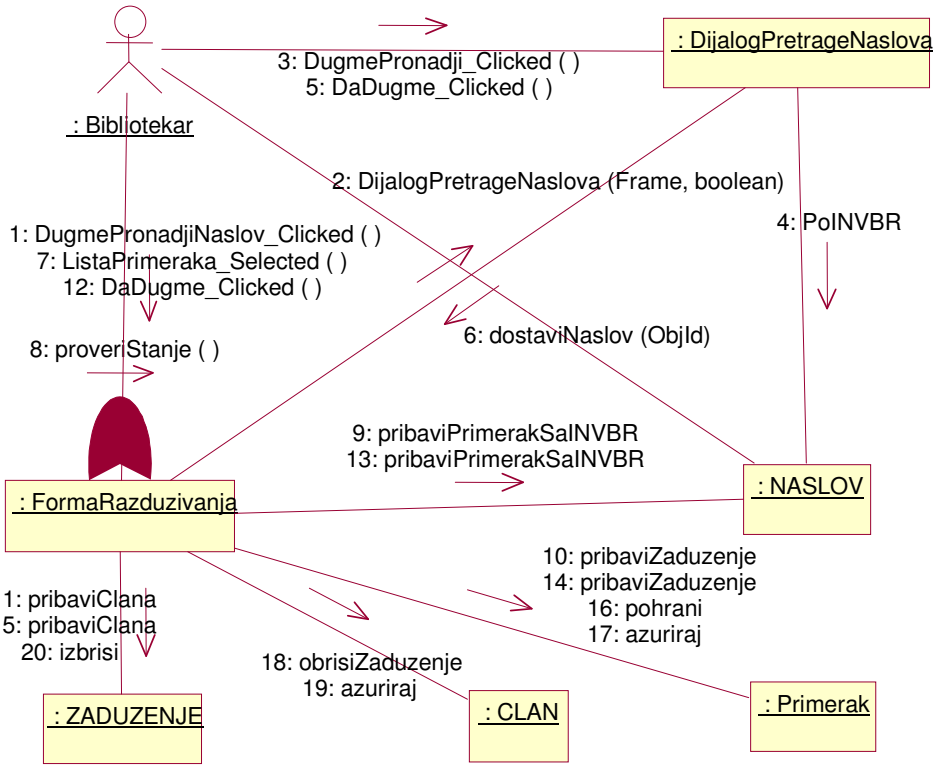


Slika 6.32. Dijagram saradnje interfejs zaduživanja

Dijagram saradnje interfejsa razduživanja

Razduživanjese izvodi slanjem poruka za traženje naslova na koju se otvara dijalog pretrage naslova. Mora se naći naslov koji se želi razdužiti. Naslov se pretražuje po inventarnom broju. Od naslova se uzima broj primeraka, i od ponuđenih primerak bira se primerak koji se želi razdužiti. Sa tog primerka uzima se podatak o zaduženju u kome se nalazi i informacija o članu koji duži dati primerak. Sada postoje svi neophodni podaci za brisanje zaduženja. Podaci o zaduženju se brišu i ažuriraju za objekte klasa član i primerak, dok se za objekat klase zaduženja poziva destruktor.

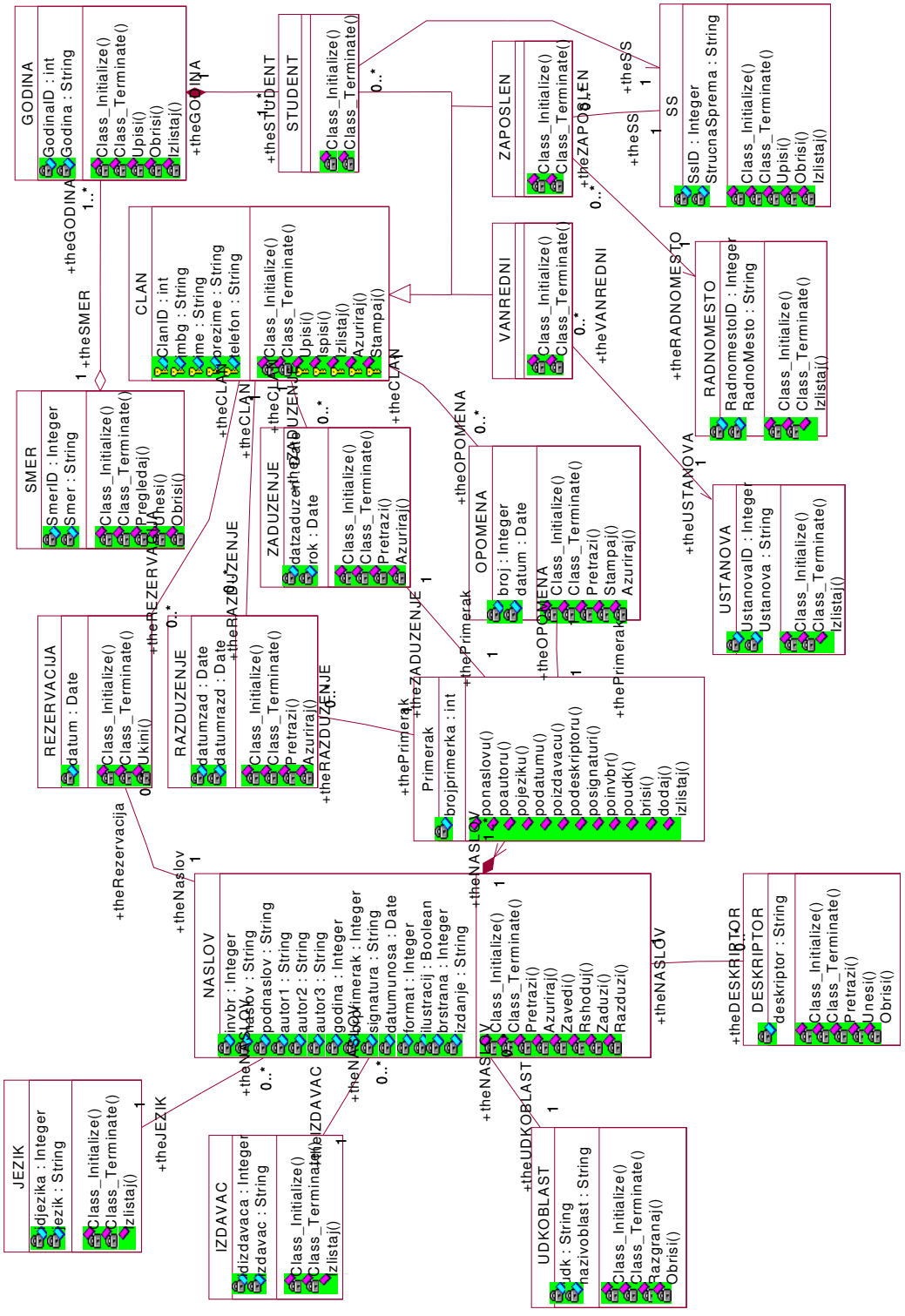
Na sledećoj slici prikazan je dijagram saradnje interfejsa razduživanja.



Slika 6.33. Dijagram saradnje interfejsa razduživanja

Izrada potpunih dijagrama klasa za poslove cirkulacije

Potpuni dijagram klasa ili kako se kraće kaže dijagram klasa se tako zove da bi se odvojio od konceptualnog modela koji se prikazuje kao dijagram klasa bez operacija ili sa onim najosnovnijim. Dijagram klase se sastoji iz klasa i veza između njih. Naziva se još i dijagram statičke strukture. Klasa predstavlja složeni tip, kolekciju, struturu koja se sastoji od više atributa (podata članova) i operacija (metoda, funkcija članica). Na sledećoj slici prikazan je potpuni dijagram klasa koji će bitiu daljem tekstu detaljno opisan.



Slika 6.34. Potpuni dijagram klasa poslova cirkulacije

Dijagram klasa koji je dat na predhodnoj slici opisuje logiku aplikacije i deo paketa istog imena koji predstavlja pogon sistema u njegovoj troslojnoj arhitekturi, o čemu će nešto kasnije biti govora. Odmah valja reći da sve klase imaju dve iste funkcije članice (metode), a to su Class_Initialize i Class_Terminate. Radi se o takozvanim konstruktorima i destruktorima klase koji se pozivaju uvek kada se kreira, odnosno uništava objekat date klase.

Na dijagramu klasa uočavaju se dve klase od ključnog značaja za problem koji se pokušava rešiti. To su klase NASLOV i CLAN.

Klasa NASLOV ima očekivane atribute (naslov, podnaslov, inventarni broj, imena autora, signatura, format, godina itd.) u odgovarajućim tipovima podataka. Neki atributi koji će se kasnije pojaviti u modelu entiteta i veza nedostaju a to je stoga što se ovde ne radi o tabelama i prenošenju stranih ključeva, već o softverskim klasama kod kojih se to rešava mogućnošću referenciranja. Klasa NASLOV takođe ima odgovarajuće metode pored već pomenutih konstruktora i destruktora. Nazivi tih funkcija jasno govore o njihovoj nameni. Važno je istaći da klasa NASLOV referencira klasu Primerak, iz raloga što je čest slučaj da jedan naslov ima više primeraka (s obzirom da veći deo fonda biblioteke čine udžbenici).

Što se tiče klase CLAN stvar je malo drugačija, jer se radi o klasi roditelja koja služi prvenstveno za izvođenje (generalizaciju) specifičnih klasa koje predstavljaju različite tipove člana koji mogu da postoje u poslovima cirkulacije. Zato klasa CLAN ima samo one atribute koji su zajednički za sve vrste članova (ime, prezime, telefon i JMBG). Ti atributi su svi protected nivoa zaštite kako bi mogli da im pristupe samo klase koje nasleđuju klasu člana (STUDENT, ZAPOSLEN, VANREDNI).

Poslovi cirkulacije odvijaju se između objekata ove dve klase i zato od njih polaze gotovo sve veze u dijagramu klasa. Razmotrićemo prirodu i smisao tih veza u ovakvom dijagramu.

Jedna vrsta tih veze koje se pomalo i ponavljaju je veza asocijacije ka nečemu što bi se u modelu objekti i veze moglo nazvati šifarnikom. Za ilustraciju ćemo uzeti vezu asocijacije klase NASLOV sa klasom UDKOBLAST. Ta veza zamenjuje atribut koji bi da je nema morao postojati u klasi NASLOV. Ovako veza asocijacije to rešava elegantije. Značenje veze asocijacije između bilo koje dve klase pa i ove je da objekti te dve klase mogu da referenciraju jedan drugi tj. to znači da objekat koji je u asocijaciji sa objektom druge klase ima informaciju o objektu te druge klase koju referencira.

Drugim rečima, naslov ima svest i zna udk broj i naziv udk oblasti kojoj pripada. Isto tako jedna udk oblast zna sve naslove koji je referenciraju i njihove atribute (to je potrebno da bi se mogli pretraživati naslovi po udk oblasti).

Takođe, ova veza ima i kardinalnost koja govori da jedan naslov može i mora da referencira samo jednu udk oblast, dok udk oblast može da postoji bez da je referencira bilo koji naslov.

U praksi to znači da će se popunjavati šifarnik udk oblasti, te da će takve oblasti moći imati u sebi svrstan nijedan ili više naslova, dok će naslov morati da ima jedan i samo jedan udk broj koji ga svrstava u stručnom katalogu. Odnos koji je opisan gotovo bez izuzetka se ponavlja kod svih klasa koje su svrstane u kategoriju šifarnika (IZDAVAC, JEZIK, SS, RADNO MESTO i

USTANOVA)

Druga kategorija klasa uslovno je nazvana grupom asocijativnih klasa. To su klase koje predstavljaju vezu klasa NASLOV i CLAN i dinamiku tog odnosa (REZERVACIJA, OPOMENA, ZADUZENJE I RAZDUZENJE). Ove klase su povezane i sa klasom NASLOV i sa klasom CLAN vezom asocijacije sličnom onoj koja je opisana u prethodnom paragrafu. Od navedene četiri "asocijativne" klase jedino klasa REZERVACIJA direktno referencira klasu NASLOV, dok ostale tri to čine preko tranzizentne klase Primerak. To je i logično jer se objekti klasa zaduženja, razduženja i opomene kreiraju za konkretni primerak naslova, a samo se rezervacija vrši za naslov bez obzira na primerak.

I navigabilnost tih veza je dvosmerna jer konkretan objekat zaduženja referencira objekat klase člana i ima sve informacije o njemu. Tako primerak klase člana referencira nijedno, jedno ili više zaduženja i ima sve informacije o tim zaduženjima. Slična je i situacija u vezi ovih klasa prema klasi Primerak. Kardinalnosti govori o mogućim stanjima dinamičkog odnosa naslova i člana. Primerak može imati nijedno ili jedno zaduženje (naslov može imati više primeraka), nijednu, jednu ili više rezervacija i opomena; dok jedno zaduženje, rezervacija, i opomena može ukazivati na jedan i samo jedan naslov. Istovetno za jednog člana može postojati nijedno,jedno ili više zaduženja, rezervacija i opomena; dok za jedno zaduženje, rezervaciju i opomenu može biti vezan samo jedan član.

Pored već navedenih tipova veza uočava se jedna koja je specifična i javlja se između klasa STUDENT i GODINA, ali i GODINA i SMER. To su takozvane veze agregacije koje reprezentuju odnos celina - deo. Radi se o dve podvrste veze agregacije : po vrednosti (kompozicija između klasa studenta i klase studentske klase) i po referenci (veza između studentske klase i službe). Smisao ovih veza je sledeći. Značenje agregacije po vrednosti: Primerak klase GODINA je vlasnik jednog ili više primeraka klase STUDENT i životni vek primerka klase STUDENT prestaje prestankom postojanja primerka klase GODINA čiji je deo.

Značenje agregacije po referenci je nešto drugačije. Jedan objekat klase GODINA može biti deo više primeraka klase SMER. Logiku veza agregacije koje su opisane slikovito dopunjuju i kardinalnosti koje su date.

Izrada dijagrama interfejs klasa

Pored dijagrama klasa kojima se opisuje logika rada aplikacije moguće je sistem još detaljnije modelovati izradom interfejsnih dijagrama klasa. To su dijagrami koji snimaju statičku strukturu izgleda korisničkog interfejsa aplikacije koja se projektuje. Dinamika klasa koje čine ove interfejs dijagrame specificirana je ranije odgovarajućim kolaboracionim i sekvencijalnim dijagramima.

Paket koji sadrži ove klase i dijagrame za razliku od prethodno opisanog dijagrama pripada gornjem sloju troslojne arhitekture softvera sistema- korisnički servis (User Services).

Ovde klase predstavljaju prozore (maske) aplikacije a veze između njih mogućnost prelaza iz jedne maske u drugu. Ti događaji nastupaju pritiskom na određene kontrole koje se nalaze na formama. Kompletni scenariji tih dešavanja dati su prethodno pomenutim interakcionom dijagramima za pojedine osnovne funkcije podsistema koje se obavljaju preko rada sa formama korisničkog interfejsa.

Slede dijagrami interfejs klasa aplikacije cirkulacije poslovanja biblioteke na uslovno podeljeni u tri grupe:

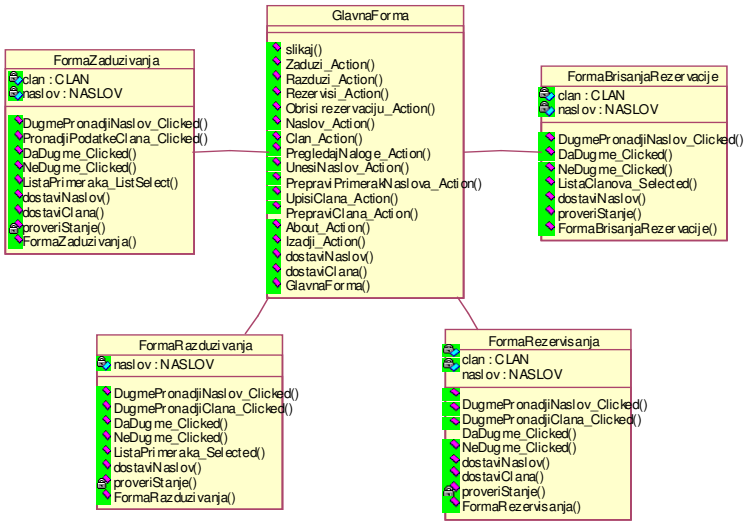
- osnovne funkcije poslovanja(zaduživanje, rezervisanje...),
- manipulisanje podacima(razne pretrage) i
- ažuriranje podataka.

Interfejs za osnovne funkcije poslova cirkulacije u biblioteci

Postoji jedna glavna forma iz koje se može, pritiskom na određene kontrole ulaziti u četiri forme gde se obavljaju četiri (u svakoj po jedna) osnovne funkcije koje treba da obavlja aplikacija bibliotečkog poslovanja: zaduživanje, razduživanje, rezervisanje i brisanje rezervacije. Prelazak iz glavne u ove ostale forme izvršavaju funkcije članice klase glavnog obrasca. Izuzimajući metodu koja je konstruktor ostale funkcije praktično otvaraju neku od maski aplikacije. One funkcije koje ne otvaraju neku od formi čije se klase nalaze na gornjem klasnom dijagramu, igraće tu ulogu u nekom od sledećih dijagrama.

Tako, metoda klase glavne forme Zaduzi_Action() otvara obrazac gde se vrši zaduživanje naslova, a koji je na dijagramu predstavljen klasom FormaZaduživanja. Ova klasa pored konstruktora ima svoje metode kojima se obavlja zaduživanje naslova, tačnije konkretnog njegovog primerka. Klasa takođe sadrži za razliku od klase glavne forme dva podatka člana tipa član i naslov. Ovi podaci članovi moraju biti prisutni jer funkcija zaduživanja vezuje podatke člana biblioteke i naslova iz bibliotečkog fonda.

Na sledećoj slici prikazan je dijagram klasa interfejsa osnovnih funkcija podsistema.



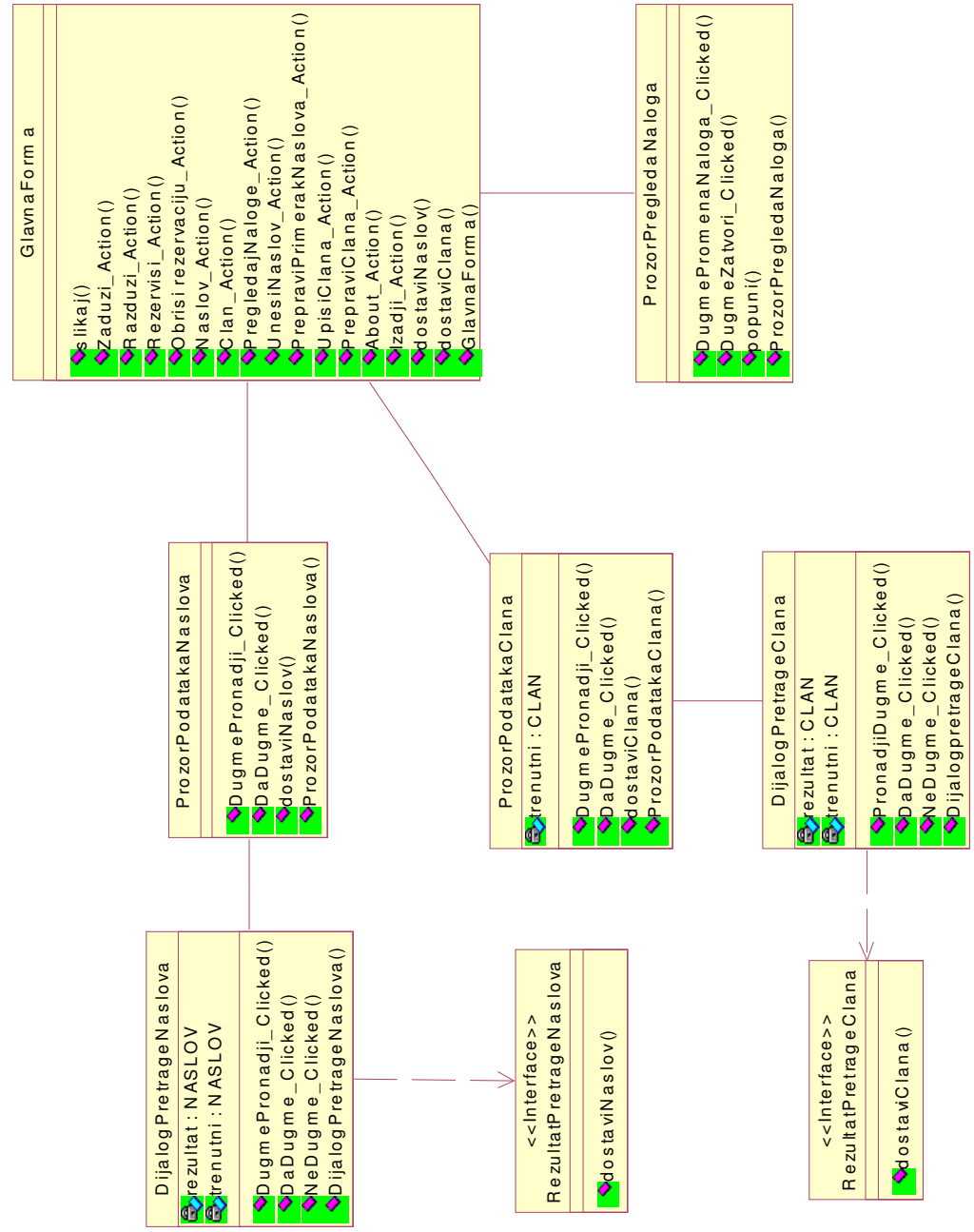
Slika 6.35. Dijagram klasa interfejsa osnovnih funkcija podsistema

Interfejs za manipulisanje podacima

Polazna klasa i na ovom kao i na prethodnom dijagramu je klasa glavne forme. Iz nje se korisnik podsistema može odlučiti da pregleda naslove, članove ili naloge za rad sa aplikacijom (ukoliko ima ovlašćenje za takav rad). Bilo za koju od mogućnosti da se korisnik odluči njoj će korespondirati neka maska.

Uzmimo recimo u razmatranje prozor pretrage člana. Klasa koja odgovara tom prozoru ima jedan podatak član tipa člana, koji sadrži podatke o članu biblioteke koji je trenutan. Pritisak na dugme pronađi, koje se nalazi na toj formi, otvara se sledeći obrazac koji je na dijagramu predstavljen klasom dijaloga pretrage člana. Pored već pomenutog podatka člana tipa član ova klasa ima još jedan podatak član istog tipa koji predstavlja rezultata pretrage. Istovremeno klasa realizuje interfejs rezultata pretrage koji dostavlja člana po zadatoj kategoriji. Istovetno je rešeno i odvijanje prelaza iz maske u masku u slučaju kada se korisnik aplikacije odlučuje za pretragu Naslova. U ovu grupu obrazaca spada i obrazac pregleda korisničkih naloga aplikacije sa svojom odgovarajućom klasom i pripadajućim metodima. U toj formi moguće je pregledanje, dopuna i prepravka postojećih naloga.

Na sledećoj slici prikazan je dijagram klasa interfejsa pretrage naslova i članova.



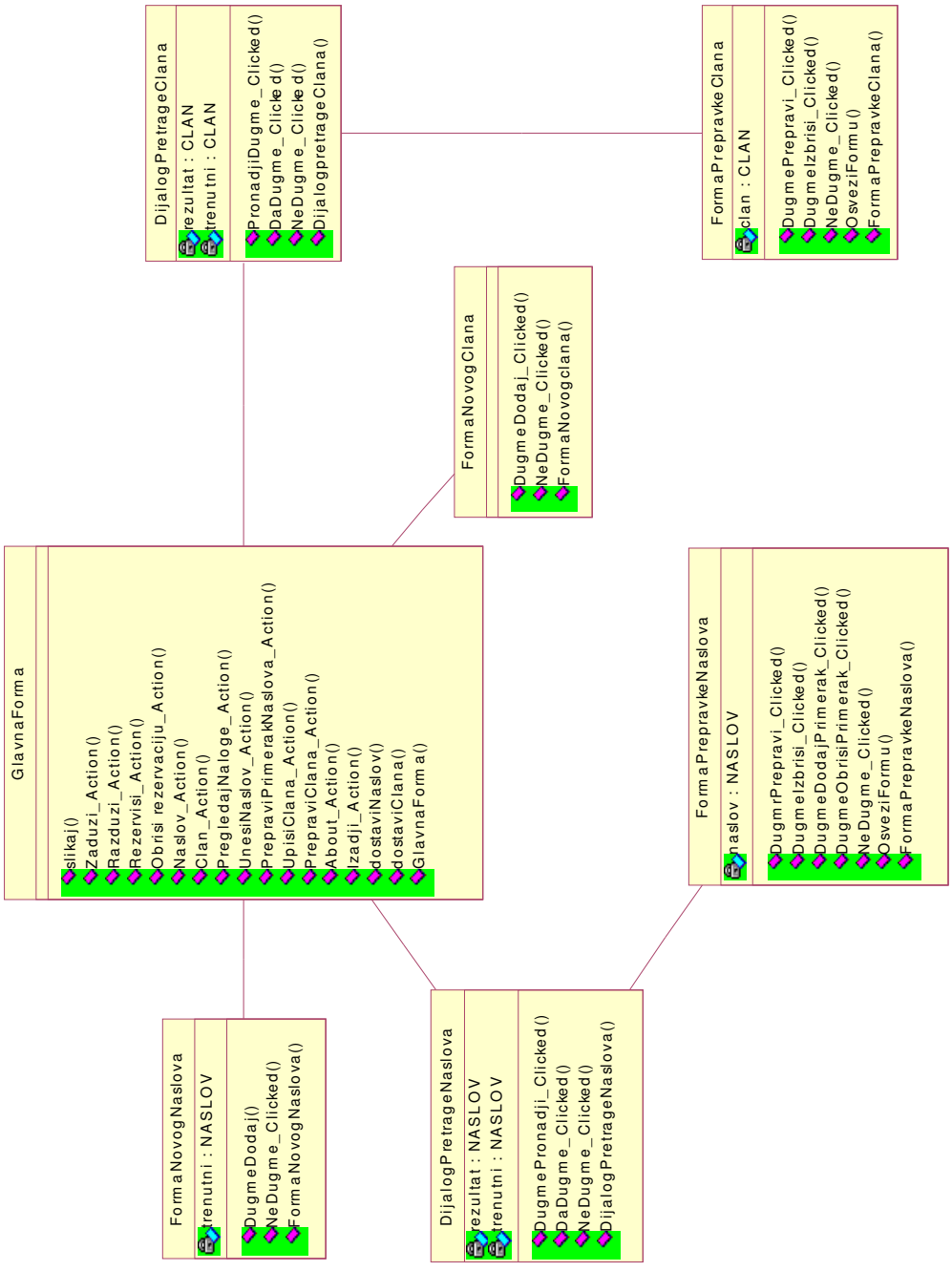
Slika 6.36. Dijagram klasa interfejsa pretrage naslova i članova.

Interfejsa održavanja ažurnosti podataka

I ovde sve počinje od jedne glavne forme, ali je svrha postojanja svih ostalih klasa koje predstavljaju obrasce aplikacije različita od prethodnih i sastoji se od održavanja tačnosti podataka u sistemu. U tu svrhu omogućene su maske (a u modelu i odgovarajuće klase koje ih predstavljaju) za unos podataka novih članova i novih naslova. Međutim, ažurnosti radi ponekad nije dovoljno samo unositi nove članove i naslove, već i intervenisati na postojećim kada se za to ukaže potreba. Zato postoje forme prvo za pretragu člana, odnosno naslova pa onda i za prepravku članova, odnosno naslova. Sve odgovornosti obrazaca omogućene su funkcijama članicama klasa koje predstavljaju te obrasce.

Na sledećoj slici prikazan je dijagram klasa interfejsa održavanja ažurnosti podataka.

Na kraju odeljka posvećenog izradi dijagrama klasa interfejsa valja nešto reći o vezama između klasa koje predstavljaju maske aplikacije. Radi se o vezama asocijacije, ali bez navigabilnosti. To znači, a u praksi tako i jeste, da maske koje su povezane tim vezama referenciraju jedna drugu, ili da mogu jedna drugu otvarati. Drugim rečima u toku rada sa aplikacijom prelaz iz jedne u drugu formu je izvodljiv. Način na koji to klase izvode pozivajući svoje funkcije članice, dinamički je opisan u poglavljima posvećenim kolaboracionim i sekvencijalnim dijagramima.



Slika 6.37. Dijagram klasa interfejsa održavanja ažurnosti podataka

Izrada dijagrama stanja za poslove cirkulacije u biblioteci

Dijagramom stanja se opisuju dinamičke karakteristike sistema. Dijagram stanja se pridružuje svakoj klasi čije je ponašanje značajno u opisu dinamike sistema. Njime se opisuje ponašanje instanci date klase, prikazuje prostor stanja (skup svih mogućih stanja) instanci klase i događaje koji iniciraju prelaz objekta iz jednog stanja u drugo kao i akcije koje se izvršavaju kao posledica promene stanja. Do promene stanja jednog objekta u sistemu dolazi kao odgovor na poruku koju šalje drugi objekat u sistemu. Dijagram stanja može opisivati dinamički model klase, paketa ili čitavog sistema. Ovde će se opisati prostor stanja objekata klasa karakterističnh za rešenje koje je ponuđeno. Pri opisu dijagrama klasa već su izdvojene neke karakteristične grupe klasa pa će se ta uslovno uzeta kategorizacija i ovde koristiti. Za svaku od kategorija daće se i opisati dijagram stanja po jedne klase koja je u tu grupu svrstana.

Radi se o prostorima stanja objekata klasa SS (stručna sprema), člana, zaduženja i naslova.

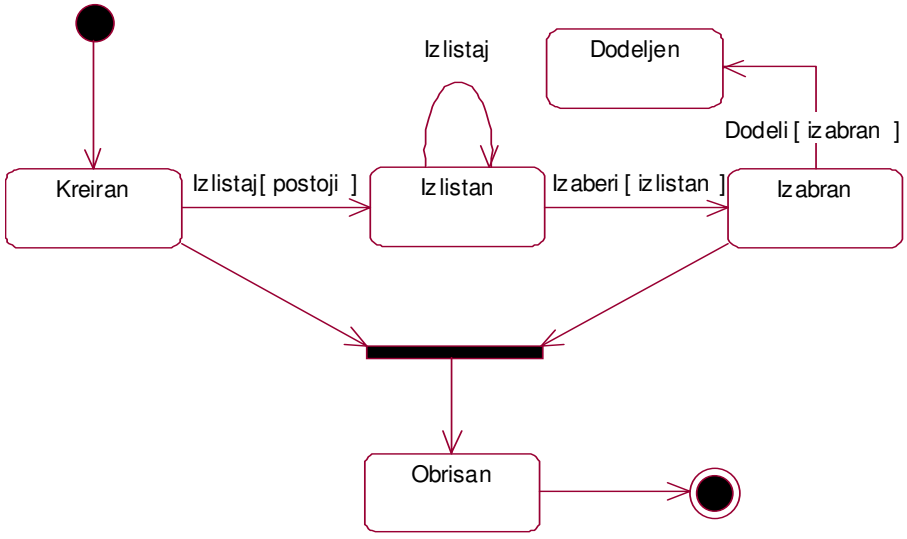
Dijagram stanja objekta klase SS

Jedan dijagram stanja, kao što je ovaj predstavlja prostor svih mogućih stanja u kojima može da se nađe jedan objekat posmatrane klase u toku svog životnog veka, od pozivanja konstruktora do pozivanja destruktora.

Dakle, nakon početnog stanja sledeće stanje u kom se može naći objekat klase SS je stanje kreiran. Odatle su mogući prelazi u dva stanja. Objekat može odmah biti izbrisan (to će se desiti kod pogrešnog unosa) ili može biti prihvaćen i korišćen, što će reći izlistan. Objekat klase SS se u aplikaciji može koristiti za dodelu strucne spreme članu ili za pretraživanje člana po stručnoj spremi. U svim tim slučajevima objekti klase SS se daju korisniku na uvid u vidu listinga.

Zato je sledeće stanje objekta izlistan. Razumljivo to mu se stanje u toku njegovog životnog veka može ponoviti neograničen broj puta, i to je razlog rekurzivnog prelaza stanja izlistan. U nekom od tih stanja izlistan objekat klase SS može biti izabran i to je novo sledeće moguće stanje. Kada je objekat izabran on se dodeljuje nekom objektu klase CLAN ili se uzima kao vrednost za pretragu. Tako je moguć i prelaz u stanje dodeljen. Međutim, iz stanja izabran moguć je i prelaz u stanje izbrisan, jer je to ono što se zapravo i dešava u formi održavanja šifarnika. Praktično s vremena na vreme se u sklopu održavanja ažurnosti šifarnika brišu činovi koji više ne važe ili unose novi. Tu bi i bio kraj životnog ciklusa objekta klase SS.

Na sledećoj slici prikazan je dijagram stanja objekta klase SS.



Slika 6.38. Dijagram stanja objekta klase SS

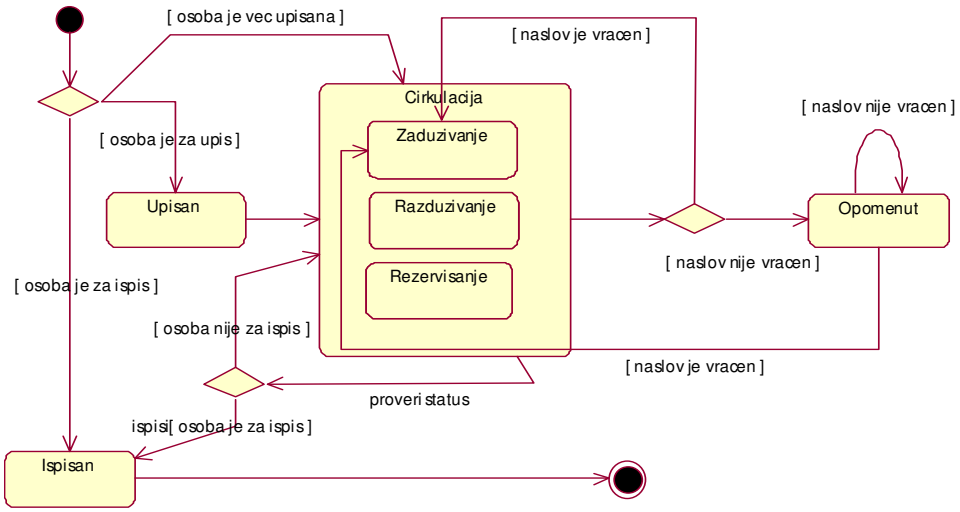
Dijagram stanja objekta klase CLAN

Na dijagramu stanja prikazanom na sledećoj slici su data sva moguća stanja u kojima se može naći objekat klase CLAN u toku svog životnog veka, od upisa do ispisa iz biblioteke. Ako objekat postoji (ako je CLAN već upisan) moguća su dva stanja, ili će se pozvati destruktor i CLAN će biti ispisan, ili će CLAN ući u cirkulaciju, gde može da obavlja sve funkcije kao što je zaduživanje, razduživanje i rezervisanje naslova iz biblioteke. Ako pak objekat ne postoji (osoba nije upisana) on prelazi u stanje upisan i onda je moguć prelaz u već pomenuto stanje cirkulacije.

U okviru cirkulacije što se tiče objekta klase CLAN ne postoje nikakva ograničenja u pogledu razduživanja i rezervisanja. Ograničenje postoji jedino za zaduživanje. Naime, na izlazu iz cirkulacije u zavisnosti da li je naslov koji objekat klase CLAN duži vraćen ili ne, moguća su dva prelaza. Ako naslov nije vraćen objekat klase CLAN prelazi u stanje opomenut i zadržava se u tom stanju sve dok ne vrati naslov. To je razlog rekurzivnog prelaza u stanje opomenut. Kada clan vrati naslov opet mu je moguć prelaz u cirkulaciju.

Iz cirkulacije (glavnog stanja poslovanja za objekte i klase CLAN i klase NASLOV) postoji jedan tok koji se odnosi na proveravanje statusa člana. Može se desiti da je član u toku cirkulacije izgubio status, pa su u zavisnosti od situacije mogući prelazi ili u opet u cirkulaciju ili u stanje ispisan.

Na sledećoj slici prikazan je dijagram stanja objekta klase CLAN.



Slika 6.39. Dijagram stanja objekta klase CLAN

Sada će biti dat dijagram stanja objekta klase zaduženja koja je u prethodnom izlaganju bila uslovno rečeno svrstana u red "asocijativnih" klasa.

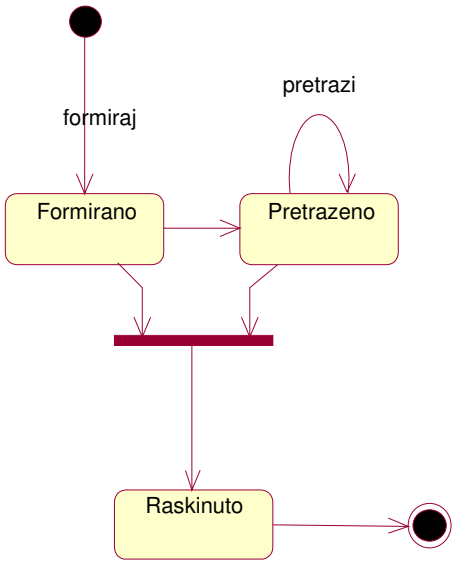
Dijagram stanja objekta klase ZADUZENJE

Prostor stanja objekta klase zaduženja nije velik. To je zato što se objekat klase zaduženja može naći u svega tri stanja : formirano, pretraženo i raskinuto. Objekat se na početku formira i ulazi u stanje formirano. Iz tog stanja moguć mu je prelaz odmah u stanje raskinuto u slučaju kada se do razduživanja ne prelistava baš to zaduženje, ili kada se napravi greška u zaduženju.

Sa druge strane moguć je prelaz u stanje pretraženo, jer se kada se jednom zaduženje formira ono u toku svog životnog veka može indirektno pretražiti bilo pri pregledu stanja naslova ili pri pregledu zaduženja nekog od članova. I ovde je ostvarena rekurzivna veza ka stanju pretraženo jer se to stanje u toku životnog veka objekta klase zaduženja može više puta ponoviti.

Iz stanja pretraženo može se preći u stanje raskinuto, što se u praksi i dešava, jer pre svakog čina razduženja mora prethoditi pretraživanje tog zaduženja.

Na sledećoj slici prikazan je dijagram stanja objekta klase ZADUZENJA.



Slika 6.40. Dijagram stanja objekta klase ZADUZENJA

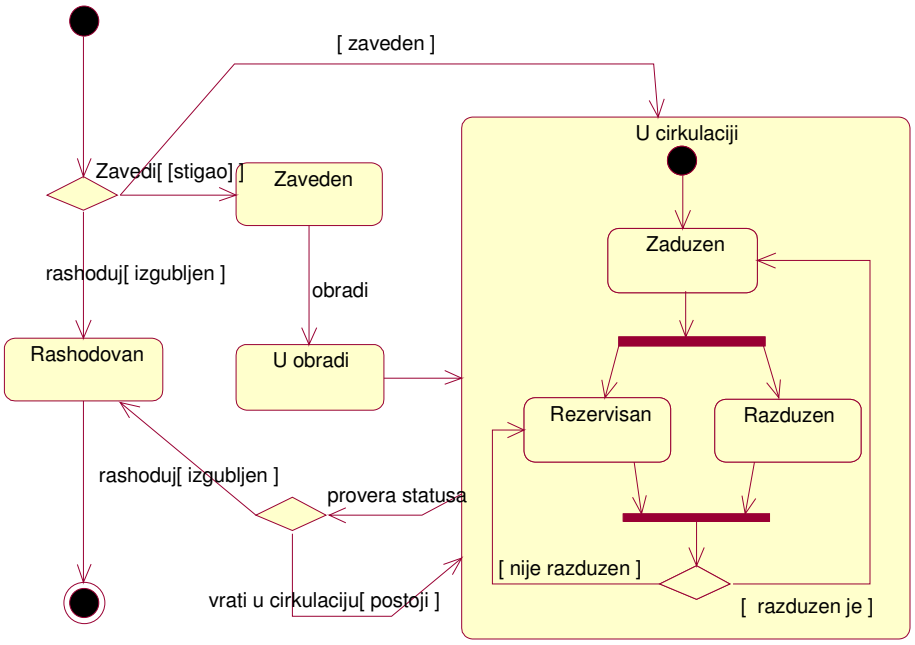
Dijagram stanja objekta klase NASLOV

Razumljivo je da je prostor stanja objekta klase NASLOV veći nego što je to bio slučaj na prethodnom dijagramu. Radi se o klasi čiji su objekti u središtu rada podsistema, pa su samim tim mogući scenariji brojniji. Objekat klase NASLOV može već postojati ili ne postojati. Ako postoji može mu se desiti ulazak u cirkulaciju (gde se dešavaju već poznate operacije) ili u stanje rashodovan (što predstavlja kraj životnog ciklusa objekta). Iz stanja zaveden prelazi se u stanje u obradi gde se vrši bibliografsko kataloška obrada naslova (dodeljuje mu se udk broj, pravi se kataloški listić...). Ovo pokazuje da nov naslov nikako ne može ući u cirkulaciju dok se ne izvrši bibliografsko-kataloška obrada.

U cirkulaciji naslov mora prvo biti zadužen da bi odatle mogao preći u stanje rezervisan ili razdužen. Naravno naslov koji je zadužen može se rezervisati. Na izlasku iz stanja rezervisan i razdužen prelaz se grana u dva pravca. Pod uslovom da je izvršena rezervacija, nju je moguće opet ponavljati, ali ne i prelaz u stanje zadužen. Prelaz u stanje zadužen moguće je samo na izlazu iz stanja razdužen. Na izlasku iz cirkulacije postoji provera statusa naslova. Ako naslov postoji on se vraća u cirkulaciju, u suprotnom vrši se prelaz u stanje rashodovano, pozivom destruktora klase naslov.

Dijagrami stanja objekata ostalih klasa slični su ili istovetni opisanim dijagramima stanja objekata klasa sa kojima se nalaze uslovno svrstani u srodnim grupama klasa.

Na sledećoj slici prikazan je dijagram stanja objekta klase NASLOV.



Slika 6.41. Dijagram stanja objekta klase NASLOV

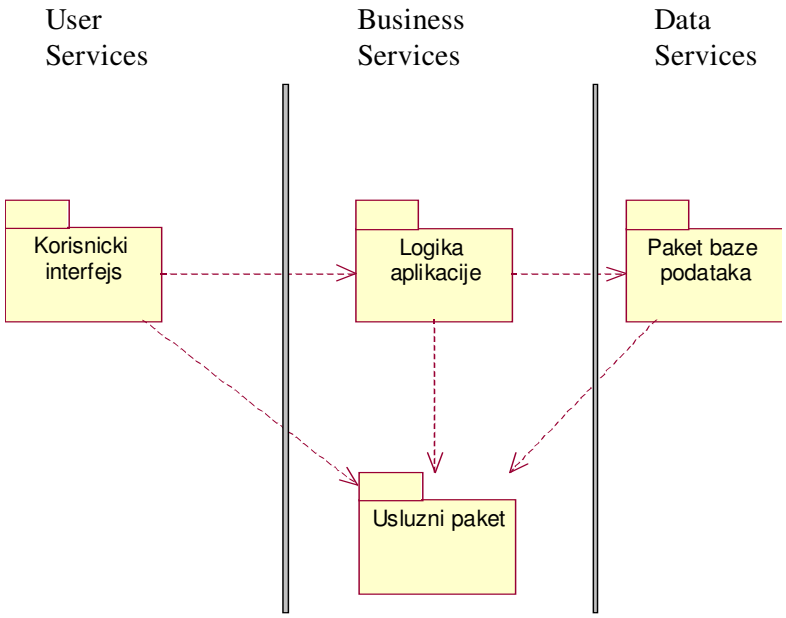
Definisanje paketa za poslove cirkulacije u biblioteci

Mnoštvo klasa i drugih stvari UML notacije redovna je pojava projekata velikih sistema, te je poželjno grupisati ih u neke logičke i funkcionalne celine. UML stvari koji predstavljaju te celine nazivaju se paketi. Paketi okupljaju semantički bliske i elemente koji se zajednički menjaju. Ponekad paket može predstavljati i pogled na sistemsku arhitekturu (primera radi u alatu RationalRose postoje četiri predefinisana paketa koji predstavljaju poglede na sistem: Use Case View, Logical View, Component View i Deployment View. Naravno pored njih mogu postojati i neki korisnički paketi. Između paketa kao i između klasa mogu se povući sve vrste veza (agregacija, asocijazija i zavisnost).

Međutim, i pored te sličnosti postoji jedna velika razlika između klasa i paketa. Klasa ima svoj primerak, instancu, pojavu dok paket to nema. On je jedan apstraktan koncept koji pre svega služi za grupisanje elemenata modela kojima je zajednička neka od mogućih veza prema drugoj grupi elemenata. S obzirom da paket predstavlja i oblast definisanosti elementa u paketu moguće je definisati i vidljivosti elemenat (klasa) unutar paketa (standardna su tri nivoa :public,

protected i private).

Na sledećoj slici biće dati paketi klasa sistema s obzirom na troslojnu arhitekturu aplikacije. O troslojnoj arhitekturi aplikativnog softvera koji se projektuje više će reči biti u poglavlju izbora tehnologije. Sada će se videti samo reprezentacija troslojne arhitekture putem paketa elemenata(klasa).



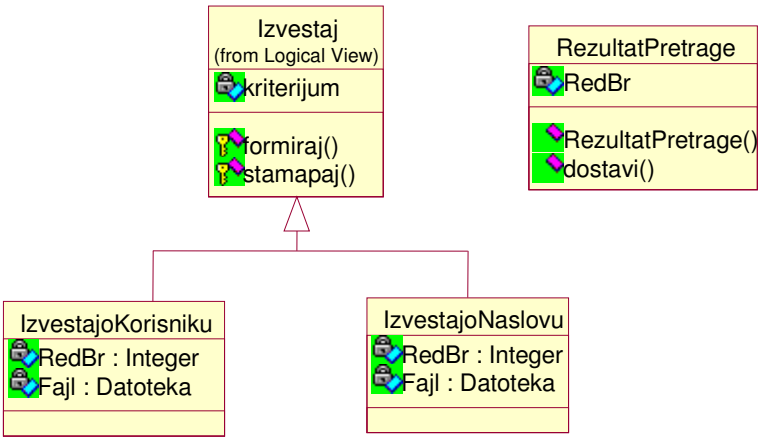
Slika 6.42. Paketi troslojne arhitekture

Dakle razlikojemo dva nivoa arhitekture i četiri paketa. O nivoima kasnije. Paketi okupljaju elemente notacije srodne po funkciji i ponašanju.

Paket *korisničkog interfejsa* sadrži sve klase i čitave dijagrame (o kojima je bilo reči u poglavlju razvoja dijagrama interfejsnih klasa) od značaja za opis i definiciju grafičke reprezentacije aplikacije -forme, dijalozi, kontrole itd...

Paket *baze podataka* sadrži klase odgovorne za postojano (persistent) čuvanje, memorisanje i održavanje podaka u skladištu.

Dva preostala paketa kao što je to slikovito prikazano imaju zadatak da povezuju dva već pomenuta paketa u funkcionalnu celinu. To su paketi *logike aplikacije i uslužni paket*. O sastavu paketa logike aplikacije bilo je već reči u poglavlju o izradi potpunih dijagrama klasa, dok u uslužnom paketu nalaze klase koje su zadužene za apstrakciju rezultata pretrage, izveštaja itd... Na sledećoj slici prikazan je sadrzaj uslužnog paketa.



Slika 6.43. Sadržaj uslužnog paketa

Implementacija za poslove cirkulacije u biblioteci

Implementacija poslova cirkulacije definisana je kroz:

- Izradu aplikacije poslova cirkulacije u bibloteci
    - Izrada baze podataka
    - Izrada korisnickog interfejsa
    - Mapiranje programiranje prevodjenje
- Definisanje tehnologije aplikativne i mrezne arhitekture za poslove cirkulacije
    - Definisanje tehnologije
    - Definisanje aplikativne arhitekure (dijagram komponenti)
    - Definisanje mrezne arhitekture (razvojni dijagram)

Izrada aplikacije za poslove cirkulacije u biblioteke

Da bi se smanjio rizik i povećala verovatnoća razvoja odgovarajuće aplikacije, razvoj bi treba da bude u što većoj meri zasnovan na analizi i dizajnu pre nego što programiranje otpočne. To ne znači da nema mesta za dizajn u toku programiranja; savremeni alati za razvoj omogućavaju brzo ispitivanje drugačijih pristupa.

Poželjeno je da dijagrami urađeni u fazi dizajna budu poluautomatski izmenjeni kako bi odrazili promene u prethodnoj fazi programiranja. Ovo se radi sa CASE alatom koji čita izvorni kod (kao što je Java) i automatski generiše, na primer, dijagrame klasa i kolaboracije.

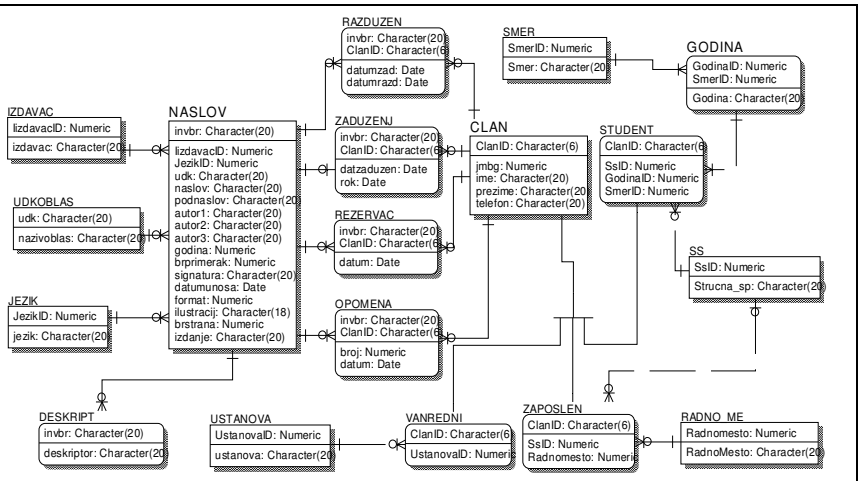
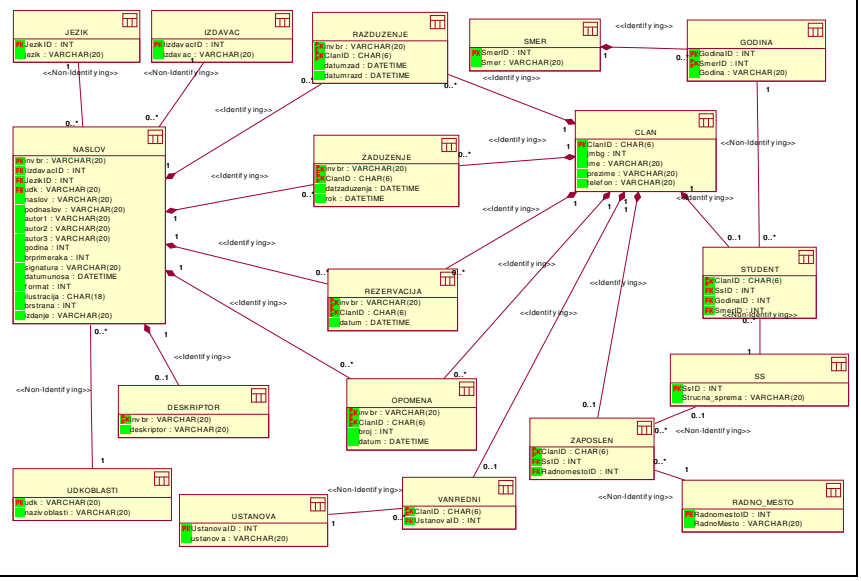
Dijagram klasa prikazuje ime klase, nadklase, prtotipove metoda, i proste atribute klasa. Ovo su neophodne stavke za stvaranje definicije klase u objektno-orjentisanom programskom jeziku.

Izradu aplikacije poslova cirkulacije u bibloteci satoji se od:

- Izrade baze podataka
- Izrade korisnickog interfejsa
- Mapiranje programiranje prevodjenje

Na sledećoj slici prikazan je fizički model podataka definisan u Erwinu i i odgovarajući fizički

model podataka u RationalRous-u dobijen postupkom reverznog inzinjeringa iz skript fajla generisanog u Erwin-u.

Fizicki model podataka u ERwin-u	
Model podataka u Rational Rous-u	

Slika 6.44. Model podataka

Izrade korisničkog interfejsa za poslove cirkulacije u biblioteci

Ovde će se opisati izgledi konkretnih formi i funkcionalnosti pojedinih kontrola koje se na njima nalaze. Već je rečeno koje su to funkcije koje sistem treba da ostvari, a sada će se videti i koje su to maske i kontrole koje ih ostvaruju.

Imajući u vidu da je korisnički pogled definisan u dijagramima slučajeva upotrebe, to se radi se o istoj strukturi funkcionalnosti. Ipak mogu se zapaziti izvesna razlika u odnosu na strukturu organizacije kontrola na formama aplikacije, koje predstavljaju inicijalizatore za ostvarivanje tih funkcija. Jedini razlog takvog rasporeda kontroli (komandnih dugmića) je omogućavanje rada koji

bi bio što bliži korisnicima.

Svi koncepti do kojih se došlo kroz objektno orjentisan razvoj u potpunosti su ugrađeni u rešenja korisničkog interfejsa i logike aplikacije, bez obzira na raspored komandi na maskama. Pre nego što se daju stavke glavnog menija aplikacije, idući redom od početka startovanja programa, prvo se nailazi na dijalog za autenfikaciju korisnika aplikacije. Radi se o tome da je u aplikaciju ugrađen svojevrstan sistem zaštite koji onemogućava rad sa aplikacijom korisniku koji za to nema ovlašćenja. Svi zaposleni u biblioteci koj rade na sitemu imaju svoje korisničko ime i odgovarajuću lozinku. Postoje dve kategorije korisničkih naloga za rad na aplikaciji. Uslovno su nazvani administratorski i obični.

Administratorski u odnosu na običan nalog ima mogućnost backup-ovanja podataka i administriranje svih korisničkih naloga i šifarnika baze podataka. Ograničavanje ovih funkcija samo na određenu kategoriju naloga bilo je neophodno kako bi se izbegle sve neželjene posledice samovoljnog unošenja podataka iz domena šifarnika. Pomenuto je i administriranje korisničkih naloga. Ono se obavalja u formama posebno namenjenim toj funkciji a za sada treba reći da je moguće dodavanje novih, brisanje i promena postojećih naloga (promena šifre).

Mora se uneti korisničko ime i lozinka imajući u vidu velika i mala slova. Naime, nije isto da li je za lozinku ukucano "E" ili "e". Sledeća forma u koju se ulazi (ako je lozinka i korisnčko ime ispravno)je forma glavnog menija aplikacije koja sadrži sledeće opcije:

• Cirkulacija
  • Zaduzivanje
  • Razduzivanje
  • Opomena
  • Rezervacija
• Pretrazivanje
  • Autor
  • Naslov
  • Izdavac
  • Jezik
  • Datum
  • Signatura
  • Deskriptor
  • UDK
  • Inventarski broj
• Korisnici
  • Upis
  • Ispis
  • Pregeled
	• Knjige
	  • Evidencija
	  • Zavodjenje
	  • Rashodovanje
	  • Stanje
	  • Kataloski listic
	  • Nalepnice
	• Izvestaji
	  • po Korisnicima
	  • po Knjigama
	  • Prinovljenih knjiga
	  • Godisnji
	  • Periodicni
	• Materijalno
	  • MP20
	  • VB-15
		• Opcije
		  • Novo nalog
		  • Izmene
		  • Promeni lozinku
		  • Izloguj se
		  • Sifarnici
		  • Smer
		  • Godina
		  • Strucna sprema
		  • Izdavac
		  • Radno mesto
		  • Ustanova
		  • Jezika
		  • UDK oblasti
		  • Dopune podataka
		  • O autoru
		• Backup
		  • Export
		  • Import
		• Kraj

Slika 6.45. Izgled menija

Iz ove forme se ulazi u sve ostale forme aplikacije, putem komandi iz menija u vrhu prozora. Sada se vidi mala reorganizacija funkcija podsistema u odnosu na stablo aktivnosti koje je dato u

poglavlju definisanja korisničkih zahteva, Tako postavljen meni je u funkciji efikasnijeg rada korisnika sa aplikacijom. Naime, korisno je imati sve operacije vezane za knjigu ili korisnika (iako je evidencija korisnika u dijagramima slučaja upotrebe bila svrstana u cirkulaciju) odvojene na jednom mestu, pogotovu što je takva organizacija menija bila prisutna i u prethodnom rešenju. U daljem izlaganju biće date opcije stavki glavnog menija. Imena tih opcija dovoljno plastično opisuju njihovu funkcionalnost, za neke kod kojih ona nije očigledna daće se i tekstualni opis. Pritisak kursorom miša na ove opcije otvara odgovarajuće forme na kojima se ostvaruje funkcionalnost podsistema.

Stavka backup-a ima opcije export-a i import-a, koje otvaraju odgovarajuće dijaloge za import i export podataka u podsistem.

Posebno je interesantan sadržaj stavke imena Opcije. Tu se administriraju korisnici aplikacije, otvaraju novi nalazi i promenjuju postojeći. Zatim se mogu ažurirati sadržaj šifarnika koji se nalaze u podsistemu (službi, ustanova, klasa, činova, izdavača, jedinica). Takođe tu se mogu dopunjavati podaci o naslovima i članovima biblioteke.

Izveštavati se može po korisnicima i po knjigama. Novina je izveštaj o knjigama koje su zavedene u biblioteci i to godišnji i periodični (za zadati vremenski interval).

U skladu sa postavljenim korisničkim zahtevima dodata je i stavka materijalno kojom se može doći do dva dokumenta koji se vode za naslove iz bibliotečkog fonda (materijalni list -MP 20 i karton bibliotečkog materijala-VB15).

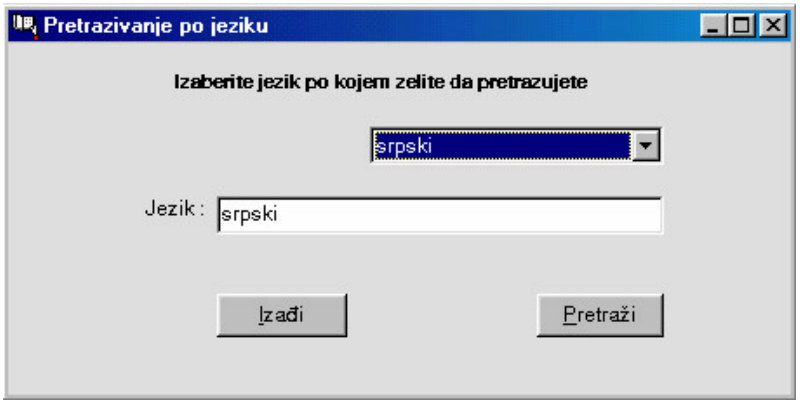
Stavke glavnog menija, kao što su cirkulacija, korisnici, knjige i pretraživanje su opcije iz sadržaja stavki u punoj funkciji i nesmetano se izvode. Recimo moguće je pretražiti naslov po deskriptoru, ali je i dodato pretraživanje po udk broju, kojim se mogu pronaći svi naslovi iz određene oblasti ljudskog stvaralaštva i određene tematike. Slede izgledi konkretnih formi koje se otvaraju izborom odgovarajućih opcija iz stavki menija, onim redom kako su date i u aplikaciji.

Zbog obimnosti aplikacije priazaće se samo elementi zaduživanja i forma knjige.

Definisanje zaduživanja

Pritiskom komande zaduživanja u stavci cirkulacija glavnog menija otvara se sledeći dijalog izbora kriterijuma pretraživanja naslova biblioteke (prethodi samom pretraživanju i zaduživanju). Pretraživanje se može izvoditi izborom opcije: Autor, Naslov, Izdavac, Jezik, Datum, Signatura, Deskriptor, UDK i Inventarski broj.

Važno je reći da će se ova forma pojavljivati i pre nekih drugih formi vezanih za naslove, jer je potrebno pronaći naslov za koji se radi izveštaj, pravi pregled, kataloški listić, nalepnica ili rezervacija. Ukupno ima devet kriterijuma i svi su operativni, a izbor se vrši jednostavnim čekiranjem nekog od kriterijuma. Istovremeno je moguće izabrati samo jedan kriterijum pretrage. U konkretnom primeru uzeto je da je izabran kriterijum pretrage po jeziku. Sledi izgled forme korisničkog interfejsa koji se otvara pritiskom na dugme dalje, u zavisnosti od izabranog kriterijuma.

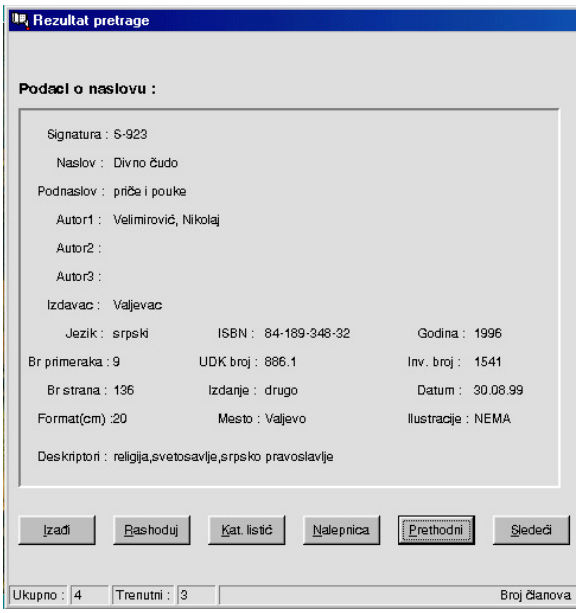


Slika 6.46. Forma izbora jezika po kojem se pretražuje naslov

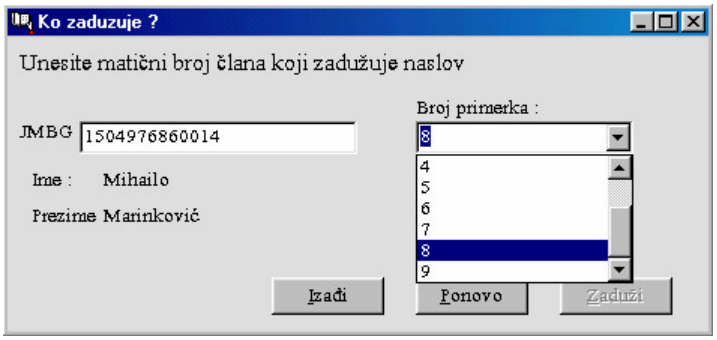
S obzirom da smo izabrali kriterijum pretraživanja po jeziku na sledećoj formi se pojavljuje element klase combobox koji vuče vrednosti naziva jezika iz tabele šifarnika jezika.

Ažuriranjem šifarnika jezika menjaće se i broj i sadržaj stavki combobox-a. Izabrani jezik se prenosi na donju tekst kontrolu. Na formi postoje i dve kontrole. To je dugme izađi kojim se ponovo odustaje od pretraživanja. Dugme pretraži odpočinje pretragu tabele naslova za svim onim koji su napisani na srpskom jeziku. Rezultat pretrage će biti dat na sledećoj formi i to će biti svi naslovi koji su ranije označavani kao dela na hrvatskom i srpskohrvatskom jeziku.

Forma rezultat pretrage koja se pojavljuje je najčešća pojava u radu sa aplikacijom BibIS 1.0. Njeno pojavljivanje može biti i rezultat nekih drugih akcija kao što je komanda rezervacije, pregleda stanja naslova, štampanja nalepnice i kataloškog listića. Stavljanjem više ovakvih kontroli, koje se sve definišu nad primerkom klase naslova, na jednu formu štedi prostor, skraćuje vreme u pristupu svim podacima vezanim za jedan naslov. Korisno je imati mogućnost obavljanjana jednom mestu više operacija nad naslovom, Na levoj strani maske se nalaze podaci vezani za naslov dok su sa desne strane podaci vezani za člana biblioteke koji duži primerak naslova (ako je naslov zadužen). Slučaj koji je i ovde prisutan (da je rezultat pretrage više naslova), da ima više naslova na srpskom jeziku, rezultuje pojavljivanjem broja takvih naslova u status baru. Uz pomoć odgovarajućih kontrola moguće je kretati se po skupu pretraženih naslova. U zavisnosti od trenutnog naslova se menja i sadržaj desnog dela ekrana. U status baru, u njegovom desnom delu, se nalazi i podatak o broju članova koji duže primerke takvog naslova, ako i njih ima više od jednog moguće je i kretati se po skupu članova koji duže trenutni naslov (kontrole kojima se to ostvaruje su dugmad pre.član i sle.član).



Slika 6.47. Forma rezultata pretrage naslova



Slika 6.48.Forma zaduživanja trenutnog naslova

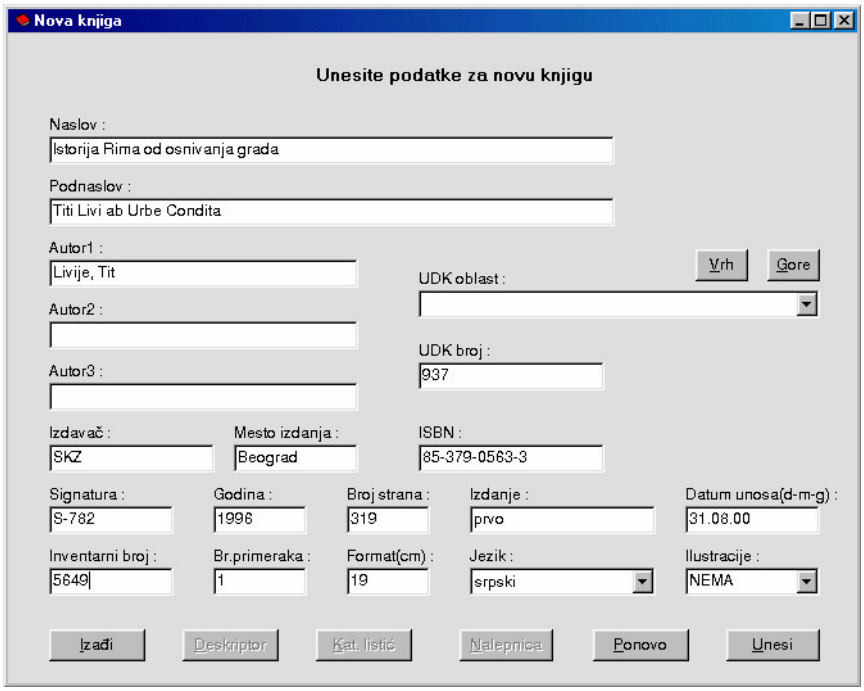
U zavisnost da li je izabrano zaduživanje, rashodovanje naslova, rezervisanje, štampanje nalepnice ili kataloškog listića neki od dugmića koji i nose imena ovih operacija nad instancom klase naslova će biti dostupni a neki ne. Ako je broj slobodnih primeraka veći od nula dugme zaduži će reagovati na događaj pritiska, u protivnom je moguće samo rezevisanje naslova. U konkretnom slučaju svi primerci naslova su slobodni pa je moguće zadužiti trenutni naslov. Pritisak na taster zaduži otvara sledeći dijalog zaduženja.

Ovde se ostvaruje koncept postavljen u toku faze dizajna, koji glasi : zadužuje se primerak, a rezerviše se naslov.

Član koji zadužuje naslov se identifikje preko matičnog broja i samo ako se pronađe član za zadati JMBG aktiviraće se dugme zaduži. No pre toga, ako se radi o naslovu koji ima više od jednog primerka u odgovarajućem combobox-u valja izabrati neki od slobodnih primeraka. Od kontrola vezanih za ovu funkciju moguće je još i odustati od zaduživanja naslova kao i izvršiti ponovni unos u slučaju greške.

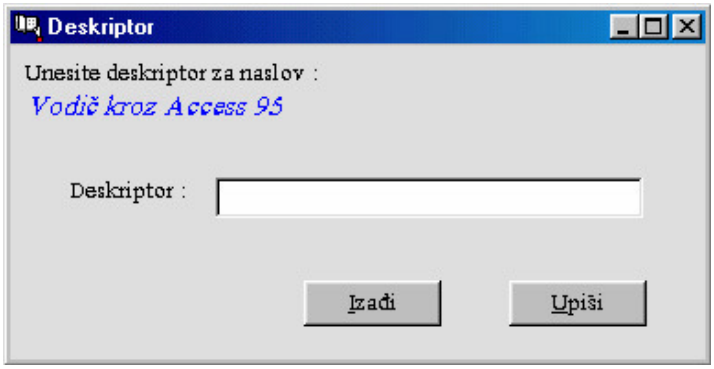
Forme knjige

U stavci knjiga moguće je pogledati stanje naslova, štampati nalepnicu naslova i kataloški listić, te rashodovati postojeći naslov. U slučaju izbora neke od tih opcija redosled otvaranja formi je istovetan onom kod zaduživanja ili rezervisanja naslova. Dakle, prvo se bira kriterijum pretraživanja, pa se unose vrednosti upita za izabrani kriterijum, a nad dobijenim rezultatima pretrage vrše se odgovarajuće operacije. Jedinu razliku u odnosu na ovakvo ponašanje predstavlja unošenje podataka novopristiglog naslova u biblioteku, koje otpočinje izborom opcije evidencija/zavođenje iz stavke knjiga glavnog menija aplikacije.



Slika 6.49. Forma zavođenja novog naslova

Dizajn i ograničenja vezani za unos podataka novog naslova sprečavaju nepravlnosti u vođenju podataka naslova. do kojih je ranije dolazilo. Postavljanjem zabrane unosa null podataka za većinu ključnih polja entiteta naslova, izbegava se slučaj nemogućnosti pretraživanja ili zaduživanja i rezervisanja naslova. U ranijem podsistemu od nešto preko 10.000 naslova u bibliotečkog fonda, preko 1500 njih nije imalo inventarni broj, što je nedopustivo jer se radi o primarnom ključu po kojem se formiraju zaduženja, rezervacije i pretražuje naslov. Jedina dva polja koja su neobavezna su polja podnaslova i koautora naslova. Pri unosu je nemoguće uneti naslov inventarnog broja ili signature koja već postoji. Neki podaci kao što je UDK broj se vuku iz odgovarajućeg šifarnika, kretanjem po hijerarhijskom stablu udk oblasti na način kako je to opisano u formi zadavnja upita po udk broju. Od ostalih komandi obezbeđene su još uobičajne komande za izlazak iz forme i ponovni unos u slučaju greške. Zatim je tu komanda deskriptor koja otvara formu za unos deskriptora tekućeg naslova čije se ime ispisuje u gornjem levom uglu.



Slika 6.508. Forma unosa deskriptora

Mapiranje proizvoda dizajna u programski kod

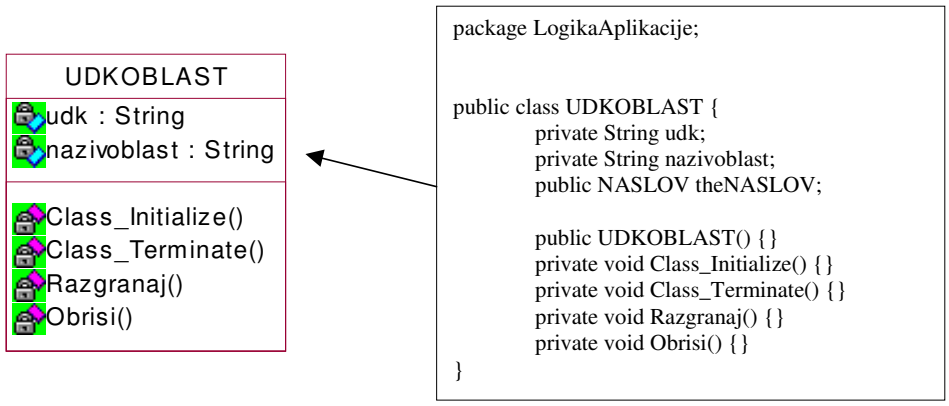
Sada će biti dati primeri klasa iz klasnog dijagrama logike aplikacije sa odgovarajućim kodom koji alat RationalRose generiše u jeziku Java.

Jezik Java, je potpuno objektno orjentisan, očigledan i pogodan za prikaze ove vrste gde je potrebno povući paralelu između modela u UML notaciji i koda koji se za njega generiše. Dakle, trenutni izbor jezika za generisanje koda uslovljen je potrebom za boljim prikazom mapiranja modela u izvorni kod. Atributi i metode koje se pojavljuju jednako i u grafičkoj prezentaciji klase i u dijagramu klasa i u deklaraciji klase neće biti posebno opisivani, osim naravno onih koji predstavljaju razliku u odnosu na dijagram klasa. Jedna metoda se u deklaraciji svake klase pojavljuje a nema je u klasi na dijagramu. Ta metoda koja je javna i nosi isto ime kao i klasa je funkcija članica konstruktor klase.

Mapiranje klase UDKOBLAST u izvorni kod

Deklaracije klase UDKOBLAST ima i pridodati referencijalni atribut tipa klase NASLOV. To znači, i već je rečeno da udk oblast mora imati iz razloga pretraživanja po udk broju podatke o svim naslovima istog udk broja.

Na sledećoj slici prikazano je mapiranje klase UDKOBLAST u izvorni kod.

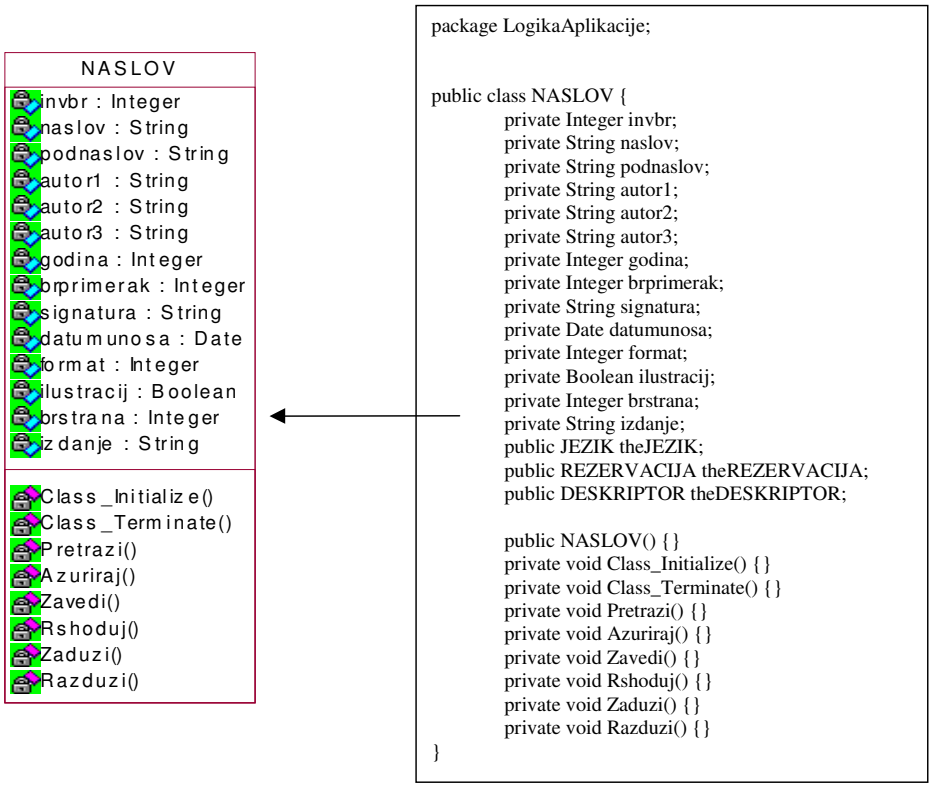


Slika 6.51. Mapiranje klase UDKOBLAST u izvorni kod

Mapiranje klase NASLOV u izvorni kod

Dodati atributi su referencijalni tipa klasa DESKRIPTOR, JEZIK i REZERVACIJA. Ovde se na konkretnom primeru može videti realizacija principa postavljenog još u fazi analize, da se rezervišu naslovi a zadužuju konkretni primerci naslova.

Na sledećoj slici prikazano je mapiranje klase NASLOV u izvorni kod.

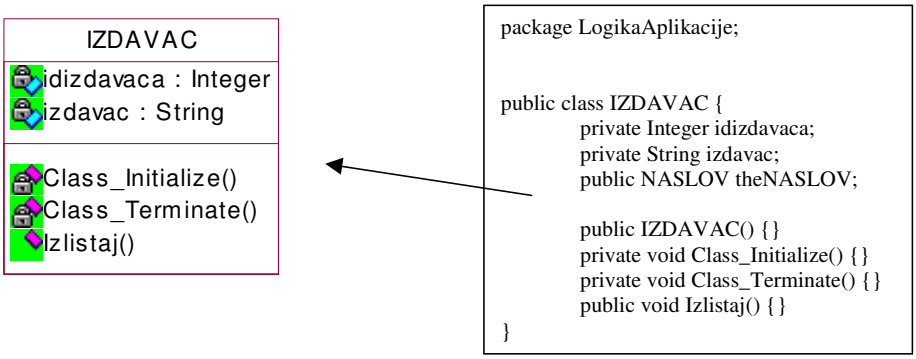


Slika 6.52.Mapiranje klase NASLOV u izvorni kod

Mapiranje klase IZDAVAC u izvorni kod

U deklaraciju klase izdavač je dodat je referencijalni atribut tipa klase NASLOV, iz razloga pretraživanja naslova po izdavaču. Svi referencijalni atributi pa i ovaj su posledica asocijativne veze između klasa, odnosno navigabilnosti tih veza.

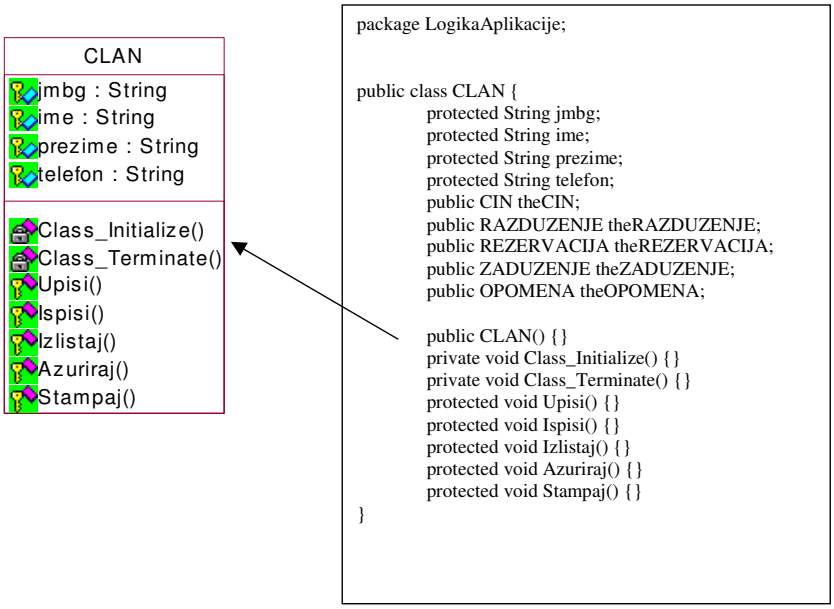
Na sledećoj slici prikazano je mapiranje klase IZDAVAC u izvorni kod.



Slika 6.53.Mapiranje klase IZDAVAC u izvorni kod

Mapiranje klase CLAN u izvorni kod

Klasa CLAN je izuzetno važna za funkcionisanje aplikacije. Njenoj deklaraciji je generator koda gde su definisani referencijalni atributi koji ukazuju na klase CIN, RAZDUZENJE , REZERVACIJA, ZADUZENJE i OPOMENA. Većina atributa i metoda ove klase su nivoa zaštite protected jer služe kao i sama klasa za nasleđivanje, specijalizaciju člana biblioteke na podklase STUDENT, ZAPOSLEN. Na sledećoj slici prikazano je mapiranje klase CLAN u izvorni kod.



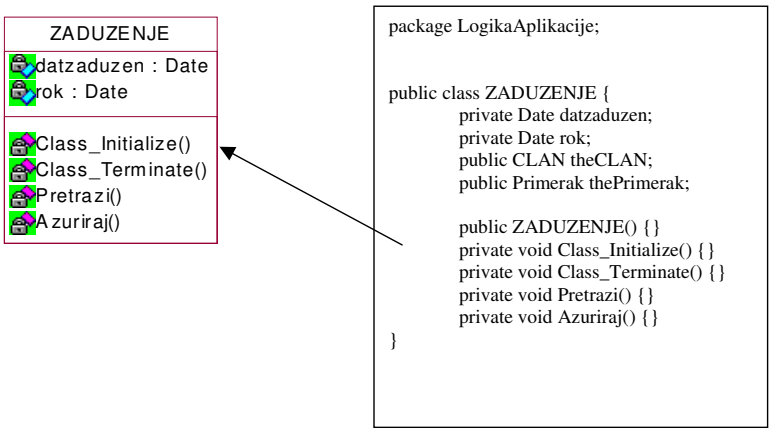
Slika 6.54.Mapiranje klase CLAN u izvorni kod

Mapiranje klase ZADUZENJE u izvorni kod

U deklaraciji klase ZADUZENJE se prisutnošću referencijalnih atributa tipa klase CLAN i Primerak. Objekat klase ZADUZENJE znači, zahvaljujući asocijativnim vezama i navigabilnošću u oba pravca ima informacije o objektima klasa CLAN i Primerak koje združuje u relaciji

zaduženja.

Na sledećoj slici prikazano je mapiranje klase ZADUZENJE u izvorni kod.

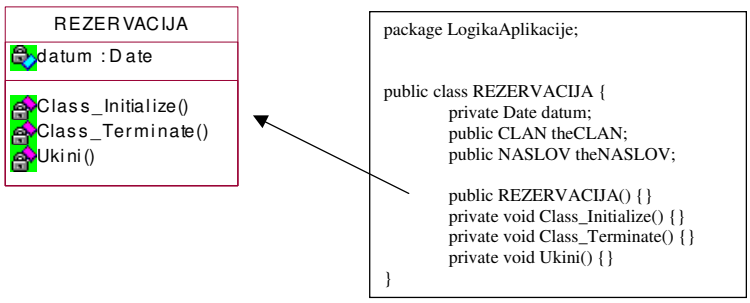


Slika 6.55. Mapiranje klase ZADUZENJE u izvorni kod

Mapiranje klase REZERVACIJA u izvorni kod

I klasa REZERVACIJA zahvaljujući svojim vezama navigabilnosti u oba pravca i smera dobija u svojoj deklaraciji kao i klasa ZADUZENJA dva nova referencijalna atributa, s tom razlikom što jedan od njih nije tipa klase Primerak već klasa NASLOV. O tome je ranije više puta bilo reči (rezerviše se naslov a zadužuje se primerak naslova).

Na sledećoj slici prikazno je mapiranje klase REZERVACIJA u izvorni kod.



Slika 6.56. Mapiranje klase REZERVACIJA u izvorni kod

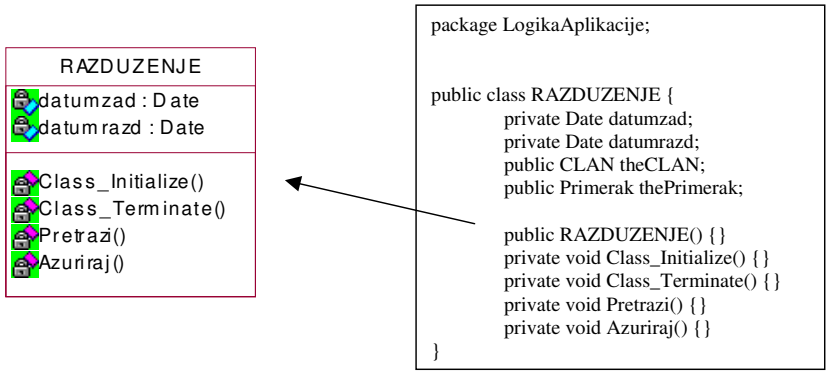
U pogledu posedovanja, na osnovu osobina veza asocijacije pridodatih referencijalnih atributa , klase OPOMENA i RAZDUZENJE (i generisani kod njihove deklaracije) se ne razlikuju nimalo u odnosu na klasu zaduženja.

Mapiranje klase RAZDUZENJE u izvorni kod

U odnosu na klasu ZADUZENJA klasa RAZDUZENJE, pored pomenutih referencijalnih atributa, ima još dva podataka člana tipa datuma koji čuvaju podatak o danu pozajmice i danu vraćanja

primerka naslova.

Na sledećoj slici prikazano je mapiranje klase RAZDUZENJE u izvorni kod.

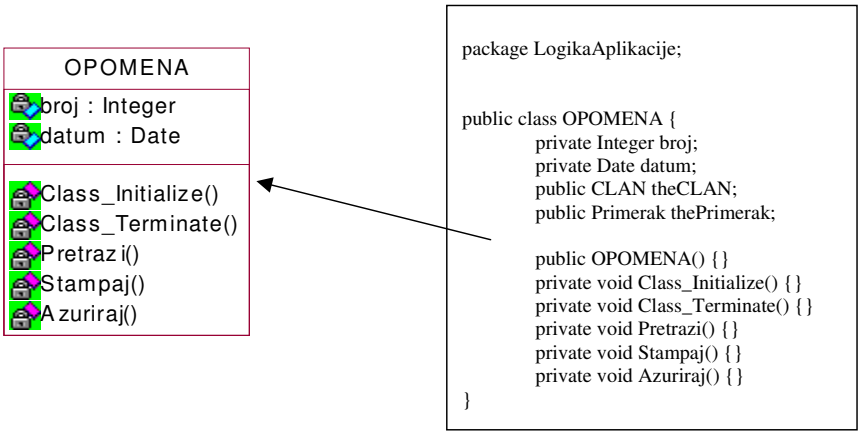


Slika 6.57. Mapiranje klase razduženja u izvorni kod

Mapiranje klase OPOMENA u izvorni kod

Objekti klase OPOMENA se kreiraju dinamički u toku izvršenja programa na događaj pritiska na određenu stavku forme glavnog menija. U takav objekat se kao što se vidi na primeru deklaracije klase smeštaju podaci o članu, primerku (kao referencijalni atributi) ali i podaci datumu i rednom broju opomene. Klasa ima i odgovarajuće funkcije članice(metode).

Na sledećoj slici prikazano je mapiranje klase OPOMENA u izvorni kod.



Slika 6.58. Mapiranje klase OPOMENA u izvorni kod

Definisanje tehnologije aplikativne i mrezne arhitekture za poslove cirkulacije

Definisanje tehnologije aplikativne i mrezne arhitekture za poslove cirkulacije

- Definisanje tehnologije

- Definisanje aplikativne arhitekure (dijagram komponenti)
- Definisanje mrezne arhitekture (razvojni dijagram)

Definisanje tehnologije

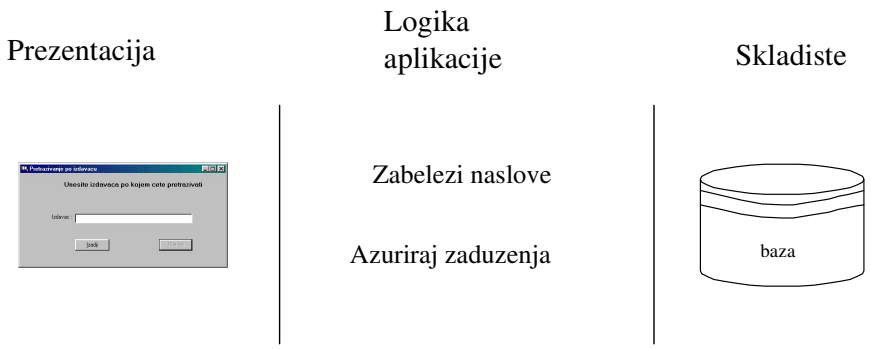
U pogledu izbora tehnologije bilo je najmanje dvoumljenja, jednostavno je vreme samo nametnulo svoje rešenje. Radi se o izboru troslojne arhitekture o kojoj je ranije bilo dosta reči.

Troslojna arhitektura je zajednička za informacione sisteme koji uključuju korisnički interfejs i postojano skladište podataka. Uobičajan opis vertikalnih slojeva je:

- Prezentacija- prozori, izveštaji, i.t.d.
- Logika aplikacije- zadaci i pravila upravljanja procesima
- Skladište- postojani mehanizam skladištenja.

Posebna prednost troslojne arhitekture je razdvajanje logike aplikacije u logički poseban srednji sloj softvera. Sloj prezentacije je relativno oslobođen od poslova aplikacije; prozori prosleđuju zahteve zadataka srednjem sloju. Srednji sloj komunicira sa poslednjim slojem skladišta.

Koncept troslojne arhitekture aplikacije razdvaja različite komponente sistema u tri sloja usluga: sloj prezentacije, sloj procesiranja i druge srednje slojeve, i sloj podataka.



Slika 6.59. Troslojna arhitektura

Ovakava arhitektura se razlikuje od dvoslojnog dizajna, u kojem je primera radi, logika aplikacije smeštena u definiciji prozora, koji čita i upisuje direktno u bazu podataka; ne postoji nikakav srednji sloj koji izdvaja logiku aplikacije. Nedostatak dvoslojne arhitekture je nemogućnost da se logika aplikacije predstavi u odvojenim komponentama, što povlači za sobom nemogućnost ponovnog korišćenja koda. Takođe nije moguće distribuirati aplikaciju između odvojenih računara.

Kod troslojne arhitekture je to moguće i odvajanjem sloja prezentacije od sloja skladištenja se postiglo to da nije bitno u kom se alatu realizuju interfejsi i kojem sistemu za upravljanje bazom podataka se pristupa. Tako je moguće raditi korisničke interfejse u Delphi-ju koji će koristiti

podatke iz jedne tako ozbiljne baze kao što je Oracle-ova.

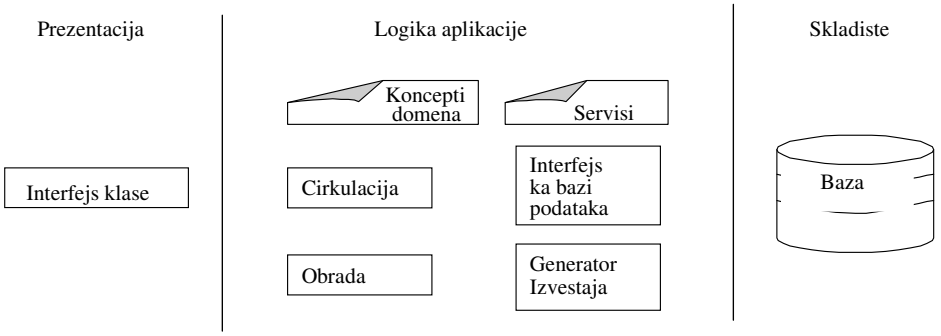
Višeslojna objektno-orjentisana arhitektura

Preporučena višeslojna arhitektura za objektno- orjentisane informacione sisteme podrazumeva razdvajanje odgovornosti koje sprovodi klasična troslojna arhitektura. Ove odgovornosti se dodeljuju softverskim objektima.

Dekomponovanje sloja logike aplikacije

U objektno-orjentisanom dizajnu sloj logike aplikacije je razložen u detaljnije slojeve. Ova arhitektura organizovana u obliku softverskih klasa je data na sledećoj slici. Sam sloj logike aplikacije razložen je na sledeće podslojeve:

- Objekti domena
- Servisi



Slika 6.60. Dalje dekomponovanje troslojne arhitekture

Kada se gleda ovako dekomponovan sloj logike aplikacije može se reći da se više ne radi o troslojnoj već o višeslojnoj arhitekturi. Moguće je čak vršiti dalju dekompoziciju, na primer, na niže servise (čitanje i upisivanje u datoteku) i više servise (izrada izveštaja).

- Troslojna logička arhitektura može biti fizički razvijena u različtim oblicima:
- Slojevi prezentacije i logike aplikacije su na klijent računaru, skladište je na serveru.
- Prezentacija je na klijent računaru, logika aplikacije je na serveru aplikacije, a skladište na odvojenom serveru podataka.

Opredeljenje u konkretnom problemu je za prvu od dve gore navedene opcije, jer ona obezbeđuje zadovoljavajuće rezultate uz manje ulaganje. Sa povećanjem upotrebe programskih jezika i tehnologija koje sa lakoćom podržavaju distribuiranu obradu, kao što je Java, i razvoj podsistema će isto tako biti sve više distribuiran.

Razlozi za uvođenje višeslojne arhitekture su:

- Izdvajanje logike aplikacije u odvojene komponente koje se kasnije ponovo mogu koristiti u

drugim sistemima.

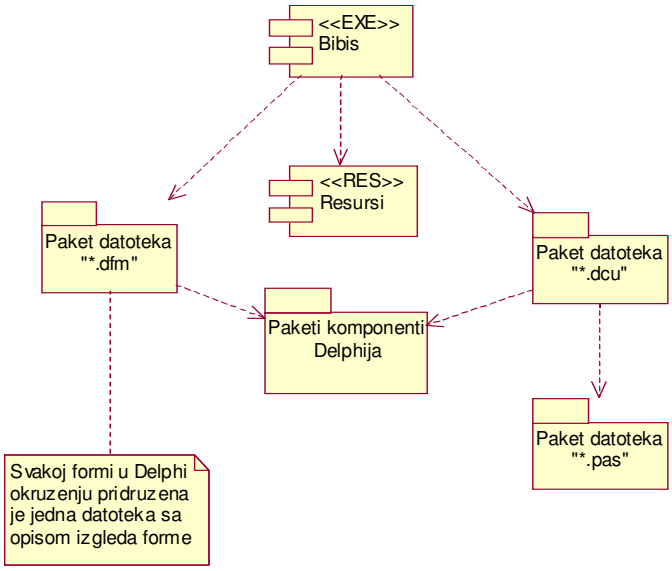
- Distribucija slojeva na različite fizičke računarske čvorove. Ovo može da poboljša perfomanse i poveća kordinaciju i deljenje informacija u klijent-server sistemu.
- Raspoređivanje programera na razvoj određenih slojeva, kao što je timski rad na sloju prezentacije. To podržava specijalizaciju u smislu razvojnih veština, i mogućnost paralelnog timskog rada.

Nakon svega navedenog ne bi trebalo da ima sumnje u opredeljenju za troslojnu arhitekturu, pogotovu kada se zna da je ona u današnje vreme postala gotovo standard. Nezavisnost korisničkog interfejsa od sistema za upravljanje bazama podataka, koji implementira postojano skladište podataka, je jednostavno nezamenljiva.

Definisanje aplikativne arhitekure (dijagram komponenti)

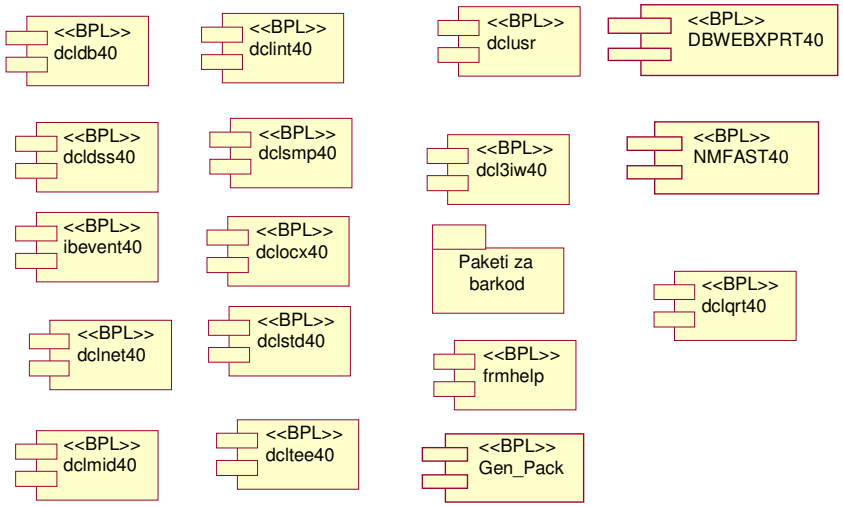
Pod ovom aktivnošću se podrazumeva definisanje softverskih komponenti koje obezbeđuju funkcionisanje aplikacije, kao i definisanje veza zavisnosti između njih. To je opis softverske arhitekture sistema. Takva komponenta je fizički izmenljiv deo sistema koji obezbeđuje realizaciju skupa interfejsa. Komponenta je fizički deo sistema jer ona zaista postoji u vidu datoteke instalirana na sistemu na kojem se izvodi aplikacija. Komponenta je izmenljiva u tom smislu što je moguće vršiti intervenciju na kodu komponente i tako na jednom mestu menjati, recimo, funkciju koja se više puta poziva u izvornom kodu. To doprinosi izuzetno poželjnoj osobini softvera koji se projektuje- ponovno iskoristljivost, reuseability of code.

Radi se o ideji da se jednom napisani kod, recimo algoritam neke obrade, nikad više ne mora pisati već se po potrebi zahvaljujući konceptu deljenja koda na komponente, koji odlikuje sve savremene alate za objektno orjentisano programiranje, jednostavno uvršćuje u različite projekte. To umanjuje vreme izrade aplikacije, povećava efikasnost u radu i smanjuje verovatnoću da će se napraviti greška u kodu. Stereotipovi komponenti sistema koji se razvija zavise od odlika okruženja (kompajlera) koji se koristi.
Tako su za jezik C++ neki od predefinisanih stereotipova za komponente softvera recimo EXE i DLL. No i ako se aplikacija ne razvija u tom programskom jeziku, kao što je to ovde slučaj, ne mari mnogo jer se stereotip softverske komponente može i ručno upisati. Inače u paketu RationalRose (koji je ovde korišćen) aplikativna arhitektura se opisuje dijagramom komponenti. Na sledećoj slici je dat osnovni komponentni dijagram za aplikaciju podsistema koji se razvija.



Slika 6.61. Osnovni komponentni dijagram

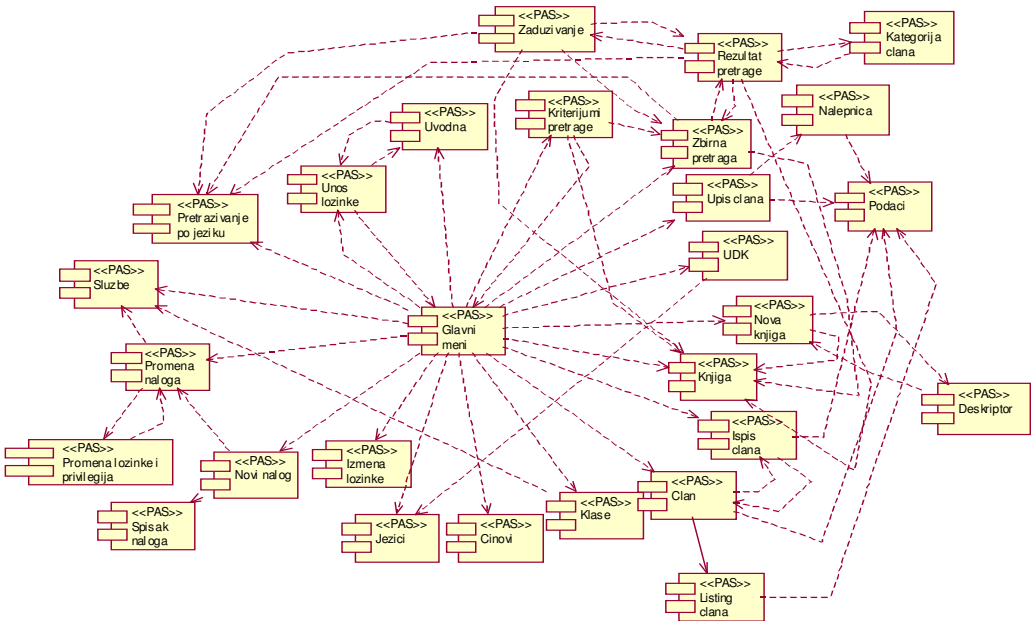
Kao što se može naslutiti iz priloženog komponentnog dijagrama, izabrano razvojno okruženje aplikacije je Delphi. To je kompajler firme Borland koji implementira objektno-orjentisan Pascal. Izvorišni kod takvog alata čuva se u datotekama tipa "*.PAS", koje odgovaraju datotekama "*.CPP" u jeziku C++. Svaka klasa u objektno orjentisanom Pascal-u ima svoju datoteku tipa "*.PAS". One su smeštene u odgovarajućem paketu na dijagramu na slici gore. Za svaku od tih datoteka (koje praktično predstavljaju klase formi Delphi aplikacije)izvorišog postoje još po dve datoteke koje je prate. To su datoteke tipa "*.DFM" i datoteke tipa "*.DCU". Datoteke "*.DFM" (Delphi Forms) sadrže parametre koji određuju spoljni, ekranski izgled svake forme- visina, širina, boja, pozicija... Datoteke "*.DCU"(Delphi Compiled Unit) predstavljaju rezultat prevođenja datoteka izvorišnog koda- ekstenzije "*.PAS". One su paralela datotekama tipa "*.OBJ" u jeziku C++. Jedna softverska komponenta, jedan fajl, koji ih povezuje je izvršni fajl radnog naziva Bibis.exe, opisan stereotipom <<EXE>>. Sve ove komponente na dijagramu su međusobno povezane vezama zavisnosti. Semantika ovih veza je sledeća: Svaka promena na fajlu (softverskoj komponenti) ili njeno nepostojanje uticaće na izgled komponente koje zavisi od nje. Na ovom primeru to bi izgledalo ovako. Svaka promena na kodu izvorišnog fajla tipa "*.PAS" uzrokovaće primene u odgovarajućoj datoteci tipa "*.DCU". Isto tako ako nema izvorišne datoteke nama ni njenog prevedenog oblika u binarnom zapisu. Pored pomenutih komponenti i paketa komponenti na dijagramu se nalazi i paket komponenti Delphi-ja, koje su ugrađene u alat ali bez njih nema valjanog prevođenja izvorišnog koda što se i da videti na dijagramu. Zavisnost važi samo ako je komponenta iz paketa korišćena u izradi aplikaciji.



Slika 6.62. Komponente Delphi okruženja

Kao što se da videti radi se o velikom broju komponenti i naravno da nisu sve korišćene za problem koji se ovde razmatra. Radi se o datotekama tipa "*.BPL"- Borland Package Library. Od nestandardnih komponenti Delphi okruženja ističe se paket komponenti za podršku rada sa bar kodom. One su neophodne s obzirom na specifikaciju korisničkih zahteva da budući podsistem integriše u sebi i predstavljanje matičnog broja CLANa i inventarnog broja knjige linjskim kodom. Pored svih pomenutih komponenti vidi se da komponenta Bibis zavisi i od komponete na slici 54. nazvane resursi, koja je zapravo jedna datoteka tipa "*.RES" i usebi čuva podatke o resursima koje koristi aplikacija. Podrazumevano u toj datoteci se nalazi podatak o ikonici projekta, a ručno se mogu dodati i drugi resursi , kao što su recimo korišćeni fontovi. Tako će aplikacija koristiti iste fontove na kojem god računaru da se instalira i bez obzira da li na njemu postoje takvi fontovi.

Sad će na sledećoj slici biti dat malo komplikovaniji komponentni dijagram sa prikazom zavisnosti datoteka izvorišnog koda. Na dijagramu se ističe komponenta datoteke " Glavni meni.pas" koja zavisi od velikog broja drugih komponenti. Značenje ovih veza je da svaka izmena u nekoj od komponenti menja funkcionisanje komponente "Glavni meni.pas".



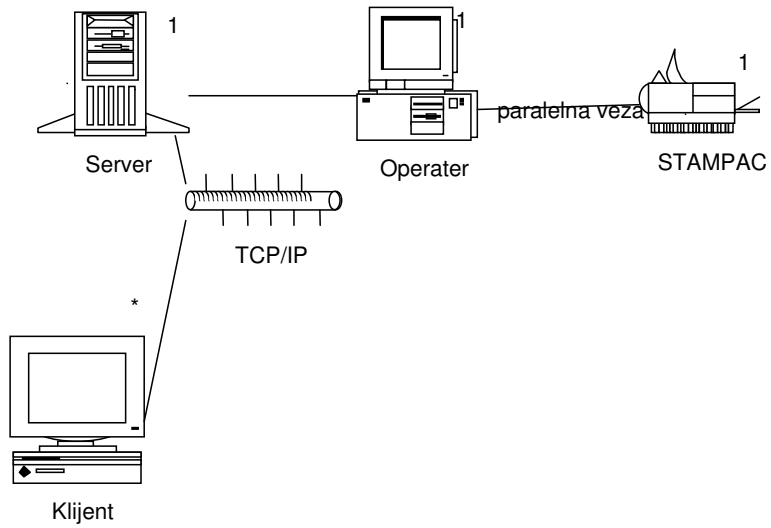
Slika 6.63. Komponentni dijagram sa prikazom zavisnosti datoteka izvorišnog koda

Definisanje mrežne arhitekture (dijagram razvoja)

Ovo je zadnji korak u objektno orjentisanom dizajnu. Podrazumeva definisanje strukture hardverskih komponenti na kojoj treba da se izvršavaju pomenute softverske komponente. Pored hardverskih komponenti mogu se predstaviti i softverske komponente koje se na njima izvršavaju. Pomenuta struktura se najbolje opisuje dijagramom razmeštaja (deployment diagram) u alatu ParadigmPlus, koji za razliku od alata RatonalRose ima vizuelniju reprezentaciju hardverskih uređaja sistema.

Za razliku od postojećeg okruženja novo bi u sebe uključilo znatna HW proširenja kao i promene u arhitekturi podsistema. Zadržava se centralni server na kojem operišu zaposleni u biblioteci, gde se nalaze svi podaci (baza podataka) vezani za bibliotečko poslovanje. Ostaje i štampač povezan na server standardnom paralelnom linijom, sa potrebom nabavke štampača savremenije tehnologije (ink jet ili laser) kako bi se ubrzao rad na uvođenju novih naslova i clanova u poslovanje, ali i izveštavanje. Najveća novinu predstavlja uvođenje određenog broja klijent računara i uspostavljanje odgovarajuće mrežne arhitekture. Time bi se povećala funkcionalnost podsistema, kojem bi sa udaljenih lokacija pristupali clanovi biblioteke. Sa ovakvih klijenata bi se izvršavale funkcije pretraživanja bibliotečkog fonda i rezervisanja naslova. Veza klijenata i servera bi se realizovala standardnim TCP/IP mrežnim protokolom. U cilju optimizacije perfomansi mreže većina rutina bi se nalazila na serveru i tu se izvršavala. Svakako da realizacija ovakvog okruženja iziskuje izdvajanje izvesnih materijalnih sredstava.

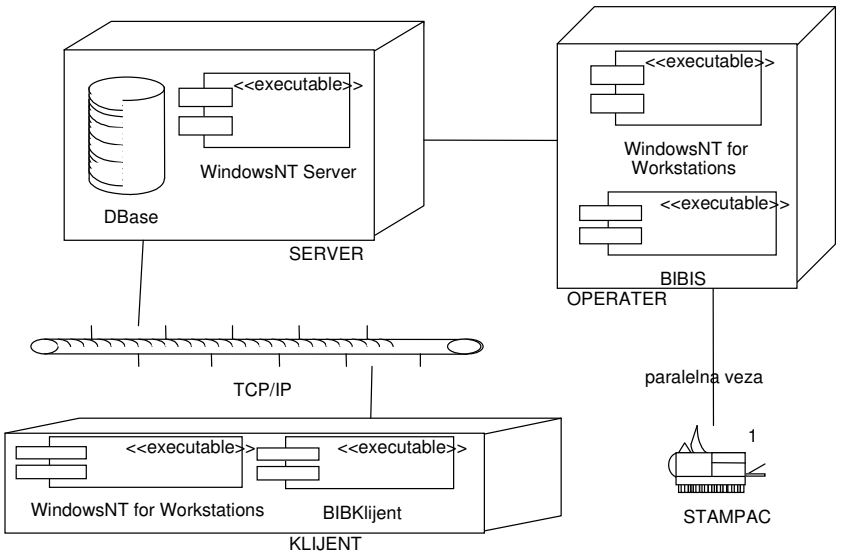
Dijagram razvoja je zapravo graf čiji su čvorovi hardverski uređaji sitema. Veza između pojedinih čvorova grafa predstavlja direktnu konekciju dva hardverska sklopa u sistemu. Na sledećoj slici je dat izgled moguće hardverske strukture podsistema bibliotečkog poslovanja.



Slika 6.64. Dijagram razmeštaja

Dakle, moguće rešenje bi bilo postojanje jednog računara servera na kojem bi se čuvali podaci i više računara klijenata na kojima bi se obavljali korisnički servisi nad podacima. Veza ka serveru bi se ostvarivala TCP/IP mrežnim protokolom. Kako bi se server oslobodio bilo kakvog rada u odnosu na dijagram razmeštaja iz poglavlja definisanja zahteva dodat je i jedan čvor u vidu računara operatera na kojem bi radili zaposleni u biblioteci. Naime klijenti su namenjeni CLANovima biblioteke odakle bi se mogli pretraživati naslovi i vršiti rezervacije. sve ostale funkcije, kao što je više puta naglašeno, mogli bi da vrše zaposleni u biblioteci preko računara-operatera. Na računar-operater je povezan paralelnom vezom štampač preko kojeg bi zaposleni štampali odgovarajuće nalepnice CLANa, naslova, kataloške listiće, dokumenta materijalnog knjigovodstava i razne izveštaje. Zapaža se da je pored veza čvorova grafa data i kardinalnost veza. Svakako da je server jedan i samo jedan, računara klijenata je više(ali bi mogao biti i jedan) dok je operater jedan. Naravno, s obzirom da u biblioteci ima više zaposlenih (troje) ostavlja se mogućnost da ih bude i više, ali se postavlja pitanje ekonomske opravdanosti takvog ulaganja kada se uzme u obzir dnevni protok CLANova i naslova kroz biblioteku.

Treba ipak istaći da je dijagram sa slike gore, nepotpun u smislu što predstavlja ogoljen hardver bez podataka o softveru koji se na njemu izvršava. Taj nedostatak će se pokušati otkloniti slikom koja sledi.



Slika 6.65. Dijagram razmeštaja sa odgovarajući softverskim komponentama

U pogledu hardveske strukture dijagrama na nije pretrpeo nikakve izmene. Novinu predstavljaju softverske kompnente koje su na čvorovima grafa hardverske strukture an kojima se izvršavaju. Razlikuju se softverske komponente operativnog sistema i aplikacije. U pogledu operativnog sistema izbor je pao na WindowsNT firme Microsoft za rad u mrežnom okruženju. Taj operativni sitem je sa jedne strane dovoljno komercijalan da bi bio pristupačan, a sa druge strane sasvim solidno ispunjava bezbedonosne zahteve kakve može imati rad u mrežnom okruženju. Naravno na serveru će biti instalirana server verzija ovog operativnog sistema, dok će na klijentima i operateru biti instaliran WindowsNT for Workstations. Baza podataka se nalazi na serveru. Aplikacija bibliotečkog podsistema se u punom obimu izvršava na računaru-operateru gde administriranje vrše zaposleni u biblioteci . Na računarima klijentima izvršavao bi se modul aplikacije koji je ovde uslovno nazvan BibKlijent, a koji bi u skladu sa svim napred izrečenim mogao da vrši samo pretraživanje naslova i rezervisanje. Mrežna arhitektura podsistema bibliotečkog poslovanja je tako projektovana da uzima u obzir ekstremnu udaljenost pojedinih objekata u okviru.

Zaključak

Da bi rad u novom okruženju bio moguć neophodno je da zaposleni u biblioteci pomoću odgovarajuće opcije u stavkama glavnog menija dopune podatke za naslove koji do sada nisu vođeni, kao i odgovarajuće šifarnike. Pravci u kojima treba da se kreću eventualna buduća poboljšanja su proširivanje fonda sa informatičkom evidencijom časopisa (isto kao što je učinjeno za naslove), zatim ako to bude svrsishodno i razmotriti mogućnosti primene Web tehnologija u cilju ostvarivanja bolje saradnje sa bibliotekama akademskih ustanova u inostranstvu.

Literatura

1 Veljović A. Objektno modeliranje informacionih sistema, Fakultet za poslovne studije, MEGATREND Univerzitet, 2003. godina

2. Veljović A. Praktikum iz analize informacionih sistema, Fakultet za poslovne studije, MEGATREND Univerzitet, 2005. godina

3. Veljović A. Objektni pristup razvoju informacionih sistema i baze podataka, VTA Beograd, 2002. godina.