3. Strukturna sistemska analiza

3.1. Uvod

Glavni cilj razvoja, odnosno uvođenja informacionog sistema (IS) je da se informaciono podrže suštinski *procesi* koji se odvijaju u jednoj organizaciji potrebni za ostvarenje njenih ciljeva. Iz tog razloga je na samom početku razvoja IS neophodno utvrditi koji su to suštinski procesi u organizaciji koje treba podržati, a to drugim rečima znači, postaviti sledeće pitanje: Šta budući informacioni sistem u pogledu funkcija treba da radi da bi ispunio zahteve korisnika, odnosno pomogao u ostvarenju ciljeva organizacije? Analiza sistema i zahteva korisnika, kao prva faza u životnom ciklusu razvoja IS, ima za cilj da odgovori na ovo pitanje. Posmatrajući sistem sa *aspekta njegovih funkcija*, kao rezultat ove faze i ujedno kao odgovor na postavljeno pitanje dobija se *model procesa* posmatranog sistema.

Strukturna sistemska analiza predstavlja jednu od metoda za analizu sistema i zahteva korisnika, odnosno za modelovanje funkcija sistema. Ona je najšire korišćena metoda za analizu sistema u okviru konvencionalnog strukturnog pristupa razvoju IS. Razvijena krajem sedamdesetih godina, ona je uz modifikacije dugo godina predstavljala deo standardne metodologije za razvoj IS u vladi Velike Britanije, a takođe je u osnovi standarda državnih organa SAD za modelovanje funkcija. Glavna karakteristika ove metode ogleda se u mogućnosti hijerarhijskog opisa procesa sistema, tj. postupnom dekomponovanju kompleksnih procesa sistema na podprocese. SSA za opis procesa definiše skup jednostavnih grafičkih koncepata, imajući u vidu činjenicu da u analizi sistema i zahteva korisnika učestvuju i krajnji korisnici sistema.

Ako se jedna grupa čitalaca (konkretno misleći na buduće i sadašnje inženjere organizacije – menadžere) već nakon kratkog uvodnog dela pita kakvu korist ona ima od ovog poglavlja kada se sve odnosi samo na informatičare, sledeće rečeno pogađa baš njih. Analiza sistema, odnosno modelovanje funkcija ne mora de facto da znači samo funkcionalnu specifikaciju sistema, kao osnovu procesa razvoja IS. Modeli (poslovnih) procesa se mogu iskoristiti za simuliranje i analizu funkcionisanja realnog poslovnog sistema i otkrivanje „uskih grla“ poslovanja. Time modeli funkcija sistema igraju značajnu ulogu u procesu (re)inženjeringa poslovanja, odnosno poslovnih procesa u organizaciji. Takođe, dokumentovani modeli poslovnih procesa organizacije pružaju osnovu za standardizaciju poslovanja u cilju uvođenja sistema kvaliteta. Isto tako modeli poslovnih procesa pronalaze primenu i kod obuke zaposlenih, pri čemu doprinose bazi znanja jedne organizacije. Konačno, gledajući sa suprotne strane, linija menadžmenta odlučuje o tome da li će se i koje funkcije jednog IS uvesti u organizaciju, pa se poznavanje metoda za analizu sistema, odnosno modelovanja procesa vidi kao prednost u komunikaciji sa stručnjacima iz IT-a. Što se potrebe krajnjih korisnika IS i ciljevi menadžmenta organizacije u fazi analize sistema realnije i tačnije iskažu, to će efektivnija biti upotreba budućeg IS u smislu zadovoljenja postavljenih organizacionih ciljeva.

3.1.1. Sadržaj

Ovo poglavlje se bavi strukturnom sistemskom analizom, kao metodom za analizu, odnosno modelovanje funkcija sistema. Zbog toga se na samom početku jedno

poglavlje posvećuje osnovnim konceptima u analizi, odnosno modelovanju sistema. Zatim sledi detaljno bavljenje metodom SSA. S jedne strane se uvode osnovni koncepti metode SSA koji čine alat za modelovanje funkcija, dok se sa druge strane objašnjava način i postupak njihovog korišćenja. Nakon glavnog dela koji razmatra SSA, daje se sažet pregled ostalih pristupa za modelovanje funkcija sistema, podeljen na konvencionalne i savremene pristupe. Primer – studija slučaja u kojoj se modeluju funkcije jedne e-prodavnice kompletira poglavlje o metodi SSA.

3.1.2. Ciljevi

Konkretan cilj poglavlja je upoznavanje Strukturne sistemske analize, kao konvencionalne metode za modelovanje funkcija sistema. Ne samo kroz navođenje osnovnih koncepata i pravila izgradnje modela definisana ovom metodom, nego i kroz diskutovanje primene i opšteg postupka modelovanja, čitalac može ovladati tehnikom strukturne analize sistema, koja će mu omogućiti da na postupan način analizira sisteme različitih nivoa složenosti i proizvede odgovarajuće modele.

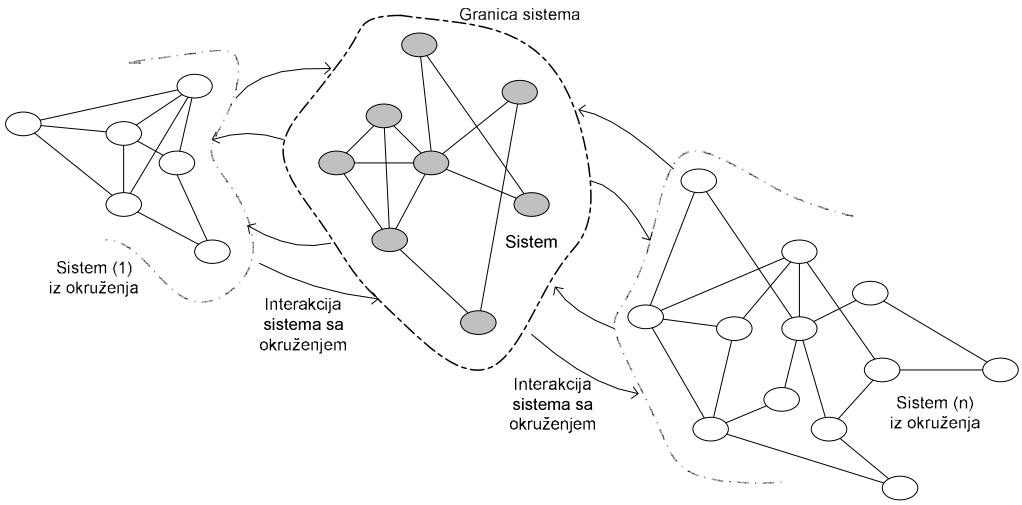
Opšti cilj poglavlja sastoji u razumevanju značaja faze analize sistema u procesu razvoja informacionih sistema. S tim u vezi želi se istaći njen značaj ne samo u pogledu specifikacije softvera, već i u pogledu modelovanja poslovanja što je naročito izražena karakteristika savremenih pristupa za modelovanje funkcija sistema.

3.2. Analiza sistema

Cilj uvodnog poglavlja o analizi sistema nije da izloži detaljnu teorijsku pozadinu o sistemima i sistemskom razmišljanju. Obimna literatura o teoriji sistema je tome posvećena. Nasuprot tome, ovo poglavlje se, na manje formalan način, osvrće na osnovne pojmove *sistema*, *analize sistema* i *modelovanja sistema*. Nakon uvedenih osnovnih pojmova, objašnjava se pojam modelovanja funkcija sistema. Osnovni cilj je da se značenje pomenutih, vrlo opštih pojmova ograniči i sažeto izloži imajući u vidu glavnu temu poglavlja Strukturnu sistemsku analizu. Time se želi obezbediti lakše, sistematskije razumevanje Strukturne sistemske analize, kao jednog od pristupa za analizu, odnosno modelovanje funkcija sistema.

3.2.1. Sistem

Besmisleno je govoriti o bilo kakvoj analizi sistema, ako se prethodno ne ustanovi šta je uopšte sistem. Postoji jako puno definicija sistema koje se međusobno razlikuju u zavisnosti od toga o kojoj se vrsti sistema govori. Međutim, jedna uopštena definicija može da glasi ovako: *Sistem predstavlja skup elemenata i njihovih međusobnih veza*. Ipak, skup elemenata sistema ne može biti neograničen skup, jer bi se, u suprotnom, svaka diskusija o bilo kom sistemu završavala sa zaključkom da je svaki sistem univerzum. Da bi bilo jasno šta čini jedan sistem, tj. koji su to njemu pripadajući elementi, sistem se mora posmatrati u odnosu na njegovo *okruženje*; mora se definisati *granica sistema*. Zato se može reći je sistem u odnosu na okruženje jedan ograničen skup elemenata i njihovih međusobnih veza.



Slika 1 Sistem i njegovo okruženje

Slika 1 prikazuje jedan sistem i njegov položaj u odnosu na okruženje. Granica sistema radvaja sistem od spoljnih sistema. Sistem je u neprekidnoj interakciji sa svojom okolinom. Interakcija sistema sa okruženjem se sastoji iz skupa ulaznih i izlaznih dejstava. Dejstvo okoline na sistem naziva se *ulaz*, a dejstvo sistema na okolinu *izlaz* sistema.

Jedan sistem se može posmatrati sa različitih *aspekata*. Svaki ugao posmatranja definiše jedan podskup sistemskih elemenata. Na primer, jedan organizacioni sistem se može posmatrati sa *funkcionalnog* aspekta, pri čemu su osnovni elementi tog sistema *funkcije* koje se u njemu odvijaju. S druge strane, isti sistem se može posmatrati sa aspekta njegove strukture, odakle se kao elementi uočavaju organizacione jedinice, odnosno radna mesta, odeljenja, divizije i dr. Gledano sa funkcionalnog aspekta, jedno proizvodno preduzeće kao svoje elemente sadrži npr. funkcije proizvodnje, nabavke, prodaje i upravljanja. Što se tiče interakcije proizvodnog sistema sa okruženjem, jedan od spoljnih sistema proizvodne organizacije može biti dobavljač osnovnih sirovina. On je sa sistemom od interesa u interakciji kroz kontinuiranu razmenu materijala i informacija. Pokušajte da na osnovu slike 1 i uvedenog primera sami skicirate sliku sličnog sistema i njegovog okruženja.

U nastavku teksta se, kada se govori o sistemima, njihovoj analizi, odnosno modelovanju, podrazumevaju dve vrste sistema: informacioni (softverski) sistemi i organizacioni (poslovni) sistemi.

3.2.2. Analiza sistema

Analiza predstavlja postupak izučavanja neke pojave u cilju njenog boljeg razumevanja, odnosno nekog problema u cilju njegovog rešavanja. Postupak analize se, generalno, karakteriše time što se posmatrana pojava, odnosno problem *razlaže* na sastavne delove koji se pojedinačno razmatraju.

Problem može da se definiše kroz postavljanje sledećih pitanja: Šta čini jedan sistem? Kako izgleda unutrašnjost nekog (kompleksnog) sistema? U tom slučaju se odgovor na ova pitanja dobija postupkom *analize sistema*.

U postupku analize sistema se sistem najpre posmatra kao „*crna kutija*[1]“. Posmatrajući sistem kao jednu celinu, utvrđuje se koji ga to spoljni sistemi okružuju, odnosno sa kojim drugim sistemima je u interakciji. Zatim se definišu skupovi ulaznih (dejstvo okruženja na sistem) i izlaznih dejstava (dejstvo sistema na okruženje). U daljim koracima se, *postupno razlažući sistem*, uočavaju sistemski elementi i njihove međusobne veze. Postupak analize sistema se završava na onom nivou detalja na kom su svi dobijeni sistemski elementi i njihove veze sa strane posmatrača dovoljno jasni da se dalje ne moraju razlagati.

Ovako postavljena analiza sistema, koja započinje posmatranjem sistema kao „crne kutije“ i njegovim daljem razlaganjem na pojedinačne elemente ima karakter metode „Od grubog ka detaljnom“ (*Top-Down*) i bazira se na pristupu sistemskog razmišljanja, holističkom pristupu proučavanja kompleksnih sistema[2]. *Top-down* metoda podrazumeva postupak analize sistema od apstraktnog ka konkretnom, odnosno od generalnog ka specijalnom.

Pod pojmom analize sistema se u oblasti informatike podrazumeva prva faza u razvoju informacionog sistema, odnosno softvera. U okviru ove faze potrebno je specificirati *šta* jedan (softverski) sistem treba da pruži u pogledu funkcionalnosti, ali ne i *kako* će sistem pružiti određenu funkcionalnost, odnosno kako će biti realizovan. Jasno razdvajanje bavljenja pitanjima *šta i kako* olakšava sa jedne strane sam proces analize sistema, a sa druge strane ostavlja mogućnost za više alternativnih realizacija sistema[3]. Pored toga, potrebno je da analiza sistema bude sastavljena iz ugla posmatranja njegovog korisnika, jer je on budući korisnik sistema, a ne sistem-analitičar koji samo vrši posao analize i eventualno učestvuje u projektovanju. Time što jasno definiše šta jedan IS treba da pruži i to uzimajući u obzir zahteve korisnika, analiza sistema ne može da osigura uspešan projekat razvoja IS, ali zato sigurno može da smanji rizik od mogućeg neuspeha.

Kao rezultat analize bilo kog sistema dobija se jasna slika o sistemu koji se posmatra. Ta slika o sistemu može ostati zapamćena u glavi sistem-analitičara (obično ne zadugo), ali sigurno to nije krajnji cilj analize sistema. Jasna slika o sistemu, odnosno znanje o sistemu se eksplicitno može predstaviti modelom sistema. O modelu i metodi modelovanja sistema će u narednom poglavlju više biti reči.

---

[1] „Crna kutija“ je sistem, ili deo sistema, sa poznatim ulazom, izlazom ali nepoznatom internom strukturom. Posmatranje sistema kao crne kutije omogućuje analitičaru da se koncentriše samo na ulaze i izlaze sistema, privremeno zanemarujući detalje vezane za internu strukturu sistema, odnosno način funkcionisanja.

[2] Sistemsko razmišljanje obuhvata [1]:

- Koncepte za opis kompleksnih celina i njihovih zavisnosti
- Pristupe bazirane na modelima, pomoću kojih je moguće predstaviti realna kompleksna pojavljivanja
- Pristupe koji podržavaju sveobuhvatno razmišljanje.

[3] Za koncepte analize sistema (šta?) i njegovog kasnijeg projektovanja (kako?) postoji interesantna fraza u engleskom jeziku: “*Do the right thing (analysis), and do the thing right (design)*”.

3.2.3. Modelovanje sistema

Modelovanje sistema predstavlja postupak kreiranja *modela sistema*. Postupak kreiranja modela sistema se definiše i opisuje *metodom modelovanja*. Naredna dva podpoglavlja detaljnije opisuju ove pojmove najpre uopšteno, a zatim u kontekstu razvoja informacionih sistema.

3.2.3.1.Model sistema

Model predstavlja subjektivnu, apstraktnu (uprošćenu) sliku sistema, opisujući elemente tog sistema i njegove veze. Model sistema je uvek subjektivna slika sistema sagledana iz ugla gledanja posmatrača. Pored toga, model je uvek uprošćena slika sistema, posmatrana sa jednog apsekta. Najprostiji primer modela je geografska karta sveta. Subjektivnost se u slučaju geografske karte, kao modela sveta, najekstremnije može uočiti ako se današnja karta sveta uporedi sa kartom sveta jednog geografa sa početka srednjeg veka koji je pokušavao da pronađe „četiri ćoška“ sveta. Geografska karta posmatra svet sa fizičkog aspekta, a ona se definitivno razlikuje od karte sveta koju gledamo za vreme vremenske prognoze, što predstavlja meteorološki aspekt. I pojam uprošćavanja, odnosno zanemarivanja detalja je očigledan. Da li se na karti sveta u razmeri 1: 5 000 000 vidi zgrada Narodnog pozorišta u Beogradu?

Čemu služe modeli? *Modeli služe za bolje razumevanje, analizu, izgradnju novog sistema ili poboljšanje postojećeg*. U toku analize sistema se uz pomoć modela zanemarivanjem nebitnih ističu samo one karakteristike sistema koje su od interesa za posmatrača i time olakšava razumevanje. S druge strane, pri izgradnji jednog sistema moguće je pomoću modela budućeg sistema ispitati njegove karakteristike, pa tako na vreme uočiti eventualne nedostatke. Da li se jedan most gradi bez prethodnog detaljnog nacrta, tj. modela mosta od strane arhitekte? U procesu razvoja softvera je takođe važno, posebno u fazi analize sistema i zahteva korisnika, sastaviti model budućeg sistema. Na taj način se karakteristike budućeg sistema mogu proveriti u odnosu na zahteve njegovih korisnika i prema potrebi, na vreme korigovati i usaglasiti. Mnogo je jeftinije i bezbolnije odbaciti jedan model sistema i napraviti novi, prilagođen korisniku, nego proizvesti gotov softver i onda ustanoviti da nikome ne koristi i zaboraviti ga.

Celokupan razvoj informacionih sistema se zasniva na izgradnji modela. Svaka faza razvoja IS posmatra, odnosno opisuje sistem sa jednog specifičnog aspekta, od kojeg zavisi i specifična vrsta modela koja se kreira. *U fazi analize sistema i zahteva korisnika se sistem posmatra sa aspekta njegovih funkcija, pa se kao rezultat te faze dobija model funkcija sistema.* Model funkcija sistema daje opis svih funkcija koje budući informacioni sistem treba da pruži svojim korisnicima.

Pored funkcionalnog aspekta, sistem se u ostalim fazama još posmatra sa aspekata strukture, dinamike, kao i fizičkog aspekta, što rezultuje odgovarajućim vrstama modela.

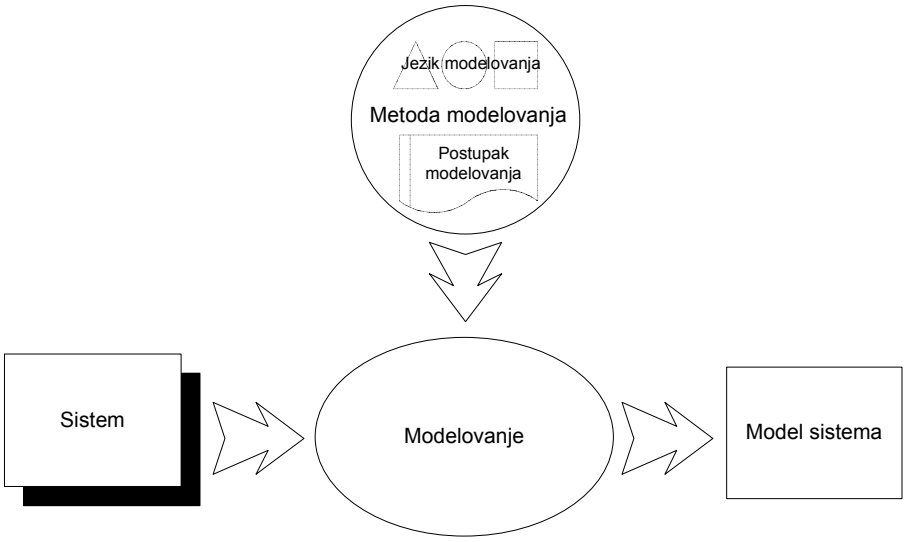
3.2.3.2.Metoda modelovanja

Metoda modelovanja[4] definiše jedan mogući način za modelovanje sistema. Šta se podrazumeva pod jednom metodom modelovanja? Da bi opisali sistem, odnosno

[4] Kao sinonim za metodu koristi se i termin „tehnika“.

kreirali model, potrebno je da raspolažemo odgovarajućim alatom, odnosno potrebno je da imamo adekvatan skup koncepata kojim se možemo poslužiti da bi svoje znanje o sistemu pretočili u model. *Jezik modelovanja* definiše skup koncepata potrebnih za izgradnju modela, odnosno predstavlja alat za opis sistema modelom[5]. Jezikom modelovanja se kao i kod govornog jezika definiše *sintaksa* i *semantika* koncepata koji se koriste i dodatno, *notacija*. Sintaksa definiše koncepte i pravila jezika i opisana je gramatikom, dok semantika definiše značenje koncepata. Notacija definiše za svaki koncept njegovu grafičku predstavu, odnosno simbol. Pored toga, postavlja se pitanje kako na najbolji način koristiti raspoložive koncepte u cilju izgradnje modela. S tim u vezi, potrebno je uz jezik kao alat za opis modela dati i uputstvo za njegovu upotrebu. Potrebno uputstvo za upotrebu se definiše *postupkom modelovanja*, tako što se navodi *redosled koraka* za izgradnju modela, kao i *rezultat* koji se dobija primenom svakog od koraka (u vidu modela ili njegovog dela).

Prema tome, jedna potpuna *metoda modelovanja* sastoji se iz: jezika modelovanja i postupka modelovanja, kao uputstva za korišćenje jezika u cilju uspešnog kreiranja modela. Jezik modelovanja se vrlo formalno može definisati, za razliku od postupka koji je obično neformalan. Slika 2 prikazuje odnos uvedenih pojmova. Modelovanje sistema se može zamisliti kao proces koji od posmatranog sistema (ulaz) generiše model sistema (izlaz), na način specificiran primenjenom metodom modelovanja (transformacija).



Slika 2 Odnos pojmova u modelovanju

Kao što sve faze razvoja informacionog sistema prate razne vrste modela, isto tako se za izgradnju tih modela koriste odgovarajuće metode. Skup specifičnih metoda definisan za svaku od faza u razvoju IS predstavlja *metodologiju razvoja informacionih sistema*.

[5]I sam jezik modelovanja se može predstaviti kao model, jer se po definiciji sastoji iz skupa koncepata i njihovih međusobnih veza. Takav model, pomoću koga opisujemo druge modele predstavlja model o modelu i naziva se metamodel.

Nakon uvedenih pojmova se možemo vratiti na glavnu temu ovog poglavlja, *Strukturnu sistemsku analizu, kao jednu od metoda za modelovanje funkcija sistema koja se koristi u fazi analize sistema i specifikacije zahteva korisnika*.

3.3. Strukturna sistemska analiza

Strukturna sistemska analiza[6] *(SSA) je jedna poptuna, samosvojna metoda za analizu i funkcionalnu specifikaciju informacionog sistema, odnosno softvera. Ona se na različite načine može povezati sa metodama drugih faza u neku specifičnu metodologiju celokupnog razvoja IS* ([5], str. 351).

Primenom metode SSA se dobija model procesa posmatranog sistema, koji treba da da odgovor na sledeća pitanja:

- Koje funkcije poseduje sistem, odnosno koje funkcije mora imati budući IS? Kakve veze postoje između funkcija?
- Kakve transformacije treba sistem da izvrši? Koje ulaze (podatke) procesom obrade transformisati u koje izlaze?
- Koje zadatke treba da izvrši sistem? Odakle sistem povlači informacije potrebne za izvršavanje zadataka? Kuda odlaze rezultati obrade?

Kao metoda modelovanja, strukturna sistemska analiza jasno definiše skup koncepata, odnosno jezik modelovanja i postupak modelovanja u cilju korektne izgradnje modela. Shodno tome će se i u narednim odeljcima opisati sastavni delovi metode SSA. Nakon uvodnog pregleda osnovnih koncepata, sledi pojedinačno detaljno izlaganje osnovnih koncepata imajući u vidu njihovu strukturu, značenje, način predstavljanja i pravila koja se moraju poštovati pri njihovom korišćenju (Poglavlja **Error! Use the Home tab to apply Beschriftung to the text that you want to appear here.**3.3.1, 3.3.2, 3.3.3). Poglavlje 3.3.4 objašnjava postupak dekomponovanja funkcija sistema, odnosno tehniku za postepeno savladavanje kompleksnosti sistema, kao glavne karakteristike metode SSA. S tim u vezi se uvodi još nekoliko koncepata, a zatim navode dodatna pravila i kriterijumi dekompozicije. Kao što će se videti, modelovanje funkcija sistema metodom SSA se bazira na uočavanju funkcija sistema na taj način što se posmatraju podaci koje funkcije sistema obrađuju. Zbog toga, pored uvedenih koncepata koji, pored funkcija, predstavljaju podatke u sistemu, poglavlje 3.3.5 uvodi Rečnik podataka, kao dodatni alat u metodi SSA za opis strukture i sadržaja podataka sistema.

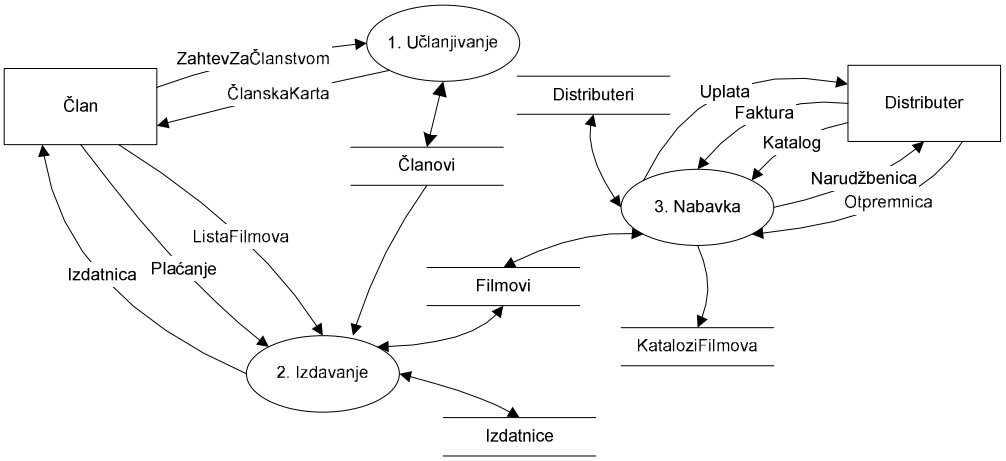
3.3.1. Osnovni koncepti Strukturne sistemske analize

SSA posmatra informacioni sistem kao *funkciju (proces obrade)* koja, na bazi ulaznih podataka i stanja sistema predstavljenog preko *skladišta podataka*, generiše izlazne podatke. Ulazni podaci se dovode u proces obrade, a izlazni iz njega odvode preko tokova podataka. Isto tako, proces obrade koristi i menja podatke o stanju sistema

[6] Zašto baš „strukturna“ metoda? Autori strukturne sistemske analize dodali su epitet strukturne, da bi naglasili njene tada napredne karakteristike u odnosu na prethodne pristupe analizi sistema, koji su se odlikovali u najvećoj meri vrlo obimnim tekstovima [2][3][4]. Upotreba grafičkih modela za specifikaciju sistema naspram dotadašnjeg „suvog“ teksta i mehanizam razlaganja sistema na podsisteme, omogućili su uređenu, kompozitnu tzv. „strukturiranu“ analizu sistema.

razmenom tokova podataka sa skladištem. *Tok podataka* se tretira kao vod kroz koji stalno teku podaci ili kao pokretna traka koja stalno nosi podatke na najrazličitijim nosiocima - papirni dokumenti, niz poruka koje čovek unosi preko tastature terminala, „paket“ informacija dobijen preko neke telekomunikacione linije ili slično ([5], str. 351). Entiteti iz okruženja sa kojima IS komunicira nazivaju se *interfejsi*. Ako ceo IS posmatramo kao jedan proces, onda interfejsi predstavljaju izvore svih ulaznih i ponore svih izlaznih tokova podataka.

Dakle, funkcije, odnosno procesi obrade podataka, skladišta podataka, tokovi podataka i interfejsi predstavljaju osnovne koncepte metode SSA. Za njihov prikaz koriste se *Dijagrami tokova podataka* (DTP). Slika 3 daje primer tipičnog DTP.



Slika 3 Primer DTP – Opis osnovnih funkcija jednog Video-kluba

Dijagram na slici 3 prikazuje model suštinskih procesa jednog Video-kluba. Cilj uvođenja ovog primera bez detaljnog rasvetljavanja značenja njegovih elemenata je da se postave sledeća pitanja: Da li se na prvi pogled dijagram čini razumljivim? Da li je notacija očigledna i jednostavna? Da li se intuitivno prepoznaje šta su funkcije, šta tokovi podataka, a šta skladišta i interfejsi? Odgovor na ova pitanja je potvrdan, bez ikakve sumnje. Jednostavnost u prikazu i intuitivnost u razumevanju predstavljaju veoma važnu karakteristiku DTP. Razlog za to se može izvući iz odgovora na pitanje kome su dijagrami uopšte i namenjeni. Sigurno ne samo sistem-analitičaru, odnosno projektantu IS, nego i krajnjem korisniku, neinformatičaru, koji treba da ocenu o tome, da li su njegovi zahtevi na adekvatan način analizirani i izmodelovani.

Na DTP se procesi predstavljaju elipsom, interfejsi pravougaonikom, tokovi podataka usmerenim linijama, a skladište podataka sa dve paralelne linije. Deo dijagrama sa slike 3 se sada može protumačiti na sledeći način: Sistem Video-kluba se sastoji od osnovnih procesa: učlanjivanja, izdavanja filmova i nabavke. Spoljni objekti sistema su članovi i distributeri filmova. Budući član dostavlja popunjen zahtev za članstvom video-klubu, gde nakon učlanjivanja dobija člansku kartu i biva zapamćen u listi članova video-kluba. Lista članova omogućuje video-klubu da vrši proces izdavanja, tako što će zahtevane filmove izdavati samo onim članovima koji su registrovani u listi članova. Video klub nabavlja filmove od ovlašćenih distributera koji šalju katalog najnovijih filmova itd.

Do sada je postalo jasno kako tumačiti DTP, ali mnogo važnije i zabavnije je kako sastaviti jedan DTP i to tako da bude sintaksno i logički korektan i pregledan. Sledeća dva poglavlja su posvećena Dijagramima tokova podataka, odnosno konceptima i pravilima za njihovo uspešno kreiranje.

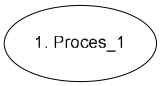
3.3.2. Dijagrami tokova podataka

Dijagram tokova podataka (DTP) predstavlja osnovni alat za opis funkcija sistema. Ali, zašto onda dijagram *tokova* podataka? Zašto se ne zove npr. „Dijagram procesa“, kada već opisuje procese? Osnovni razlog leži u načinu posmatranja funkcija sistema. Funkcije sistema se u metodi SSA, odnosno pri izgradnji DTP posmatraju kao *procesi obrade podataka*, tj. posmatraju se sa tačke gledišta *podataka*. Uočavajući podatke koji ulaze u sistem, obrađuju se i iz sistema izlaze, drugim rečima, koji teku sistemom, uočavamo i koji su procesi potrebni za obradu (transformaciju) tih podataka. Praćenjem tokova podataka je moguće ustanoviti kompletnu sliku o tome šta se to dešava unutar sistema, odnosno kako sistem funkcioniše. DeMarco opisuje taj postupak kao „intervjuisanje podataka“ [3].

DTP kao alat za opis funkcija se nadalje objašnjava kroz navođenje njegovih osnovnih komponenti: procesa, tokova podataka, skladišta podataka i interfejsa. Sintaksa i semantika ovih koncepata je u velikoj meri naslonjena na originalne autore metode SSA [3][4]. Kako postoje različiti standardi za ilustraciju simbola u okviru DTP, ovde navedena notacija je u skladu sa, na našim prostorima etabliranom notacijom utvrđenom u Laboratoriji za informacione sisteme na Fakultetu organizacionih nauka [6].

3.3.2.1.Proces

Proces predstavlja deo sistema (ili u posebnom slučaju ceo sistem, o tome više u poglavlju 3.3.4.1) koji ima ulogu da transformiše ulazne podatke u izlazne. Grafički se sistem predstavlja elipsom (Slika 4).



Slika 4 Proces

Kako proces predstavlja aktivnost, radnju, važno je imenovati ga na adekvatan način. Proces se obično imenuje parom „predikat – objekat (predmet)“. U imenima najopštijih procesa se objekat radnje može izostaviti. Primeri: Nabavka, Učlanjivanje, Knjigovođstvo, Obrada porudžbine, Provera adrese itd. Kao što se vidi predmet obrade procesa su uvek paketi podataka u nekoj formi, papirna ili elektronska dokumenta, pojedinačni podaci unešeni preko tastature računara i dr. Davanje smislenog imena procesu obrade umnogome olakšava razumljivost dijagrama i opravdava njegovo ulogu. Postoji nepisano pravilo koje kaže da, ako analitičar nije u stanju da dodeli ime procesu, to samo znači da ne razume funkciju koju proces obavlja ([6], str. 8).

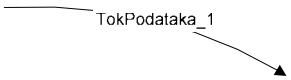
Jako je važno imati u vidu da se procesi u organizaciji koju analiziramo u jednom trenutku vremena izvršavaju *paralelno*, a ne sekvencijalno, kako bi na prvi pogled to možda izledalo. Uzmimo primer proizvodne radne organizacije, koju čine sledeći osnovni procesi: nabavka, proizvodnja i prodaja. Naravno da možemo pitati: „Čekaj,

ali redosled odvijanja ovih procesa je prilično poznat, najpre nabavim materijale, lansiram proizvodnju i konačno prodam proizvode, zar ne?“. Odgovor je zasigurno potvrdan. Međutim, ako posmatramo organizacioni sistem u celini, *u jednom trenutku vremena svi procesi se izvršavaju paralelno*, proces proizvodnje je lansiran, u istom trenutku proces nabavke obezbeđuje materijale za sledeći ciklus proizvodnje, a prodaja je prisutna na tržištu sa gotovim proizvodima iz prethodnog ciklusa proizvodnje. Još jedan primer: Da li ste slučajno primetili da ceo fakultet prestane sa radom kada dođete da prijavite ispit? Paralelizam u obavljanju funkcija je karakteristika odvijanja procesa u organizaciji ([5], str. 361). Podpoglavlje 3.3.2.3 o Skladištima podataka dodatno pojašnjava ovu činjenicu.

Svaki proces poseduje pored imena i svoju brojnu oznaku. Brojna oznaka procesa služi samo za referenciranje procesa, a nikako ne predstavlja redosled izvršavanja procesa, zbog paralelizma u izvršavanju koji je upravo diskutovan. Način numerisanja procesa će se detaljnije objasniti u poglavlju 3.3.4.3 o Pravilima i kriterijumu dekompozicije.

3.3.2.2.Tok podataka

Tok podataka se tretira kao vod kroz koji stalno teku podaci ili kao pokretna traka koja stalno prenosi pakete podataka iz jednog dela sistema u drugi, i na taj način ostvaruje vezu između komponenti sistema. Odavde automatski sledi da svaki tok podataka mora imati svoje izvorište i ponorište. Tok predstavlja podatak u stanju kretanja. Skladište predstavlja podatke u stanju mirovanja (vidi odeljak 3.3.2.3). Grafički simbol za prikaz toka podataka je usmereni luk (Slika 5).

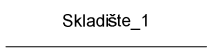


Slika 5 Tok podataka

Tok podataka se imenuje prema značenju podataka koji se tim tokom prenose. Da bi se ostvarila čitljivost i razumljivost DTP-a, imena treba da budu prirodna, nasuprot specifičnim, kodiranim, skraćenim imena. Primeri: Faktura, Narudžbenica, KatalogFilmova, ČlanskaKarta itd. Imena tokova podataka koja u sebi sadrže reči „PodaciO“, „InformacijeO“, ili još gore, „PotrebniPodaci“, „NeophodneInformacije“ treba u velikom luku izbegavati. Takva imena sugerišu na potpuno nejasnu predstavu analitičara o tome, koji se paketi podataka prenose tokom. Nemogućnost adekvatnog imenovanja može da znači da takav tok nema egzistencijalnog smisla, a to dalje znači ili njegovo potpuno uklanjanje ili razlaganje na više konkretnijih tokova podataka. Tok podataka takođe govori o usmerenju kretanja podataka. Strelica označava da li podaci u jedan proces, skladište ili interfejs poniru ili iz njega izviru i obrnuto.

3.3.2.3.Skladište podataka

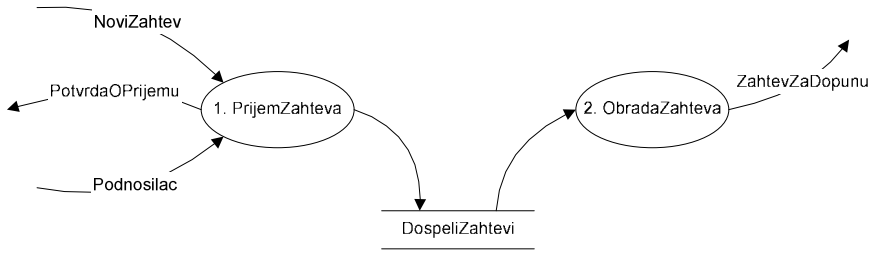
Skladište podataka predstavlja podatke u stanju mirovanja. Grafički se prikazuje kao na slici 6.



Slika 6 Skladište podataka

Ime skladišta bi trebalo da predstavlja množinu imenice toka podataka koji u njega ulazi ili izlazi, naglašavajući time da se radi o skupu objekata, paketa podataka. Na primer, skladište u koje ponire tok *NarudžbenicaDobavljača* treba da nosi naziv *NarudžbeniceDob*. Još jedan slikovit primer predstavlja skladište *Testovi*, koje se sa ulaznim tokom *IspitniTest* može zamisliti kao gomila papira koja na radnom stolu čeka da bude pregledana.

Skladište omogućava sistemu da čuva svoje stanje u vremenu. To konkretno procesima omogućava međusobnu vremensku nezavisnost, odnosno da se u slučaju različitih paketa podaka procesi mogu *paralelno* ili u slučaju istih paketa, sa *zakašnjenjem* izvršavati. Ilustrujmo to sledećim primerom. Na slici 7 je dat odsečak jednog DTP.



Slika 7 Neophodnost postojanja skladišta

Novi zahtevi pristižu u sistem i bivaju primljeni od strane procesa *PrijemZahteva*. Nakon prijema (recimo da se u procesu Prijema zahteva može vršiti provera validnosti zahteva i potrebnih dokumenata) se zahtev preko toka podatka šalje u skladište *DospeliZahtevi*. Dospeli zahtevi se čitaju iz skladišta i proces *ObradaZahteva* dalje vrši potrebnu obradu. Ukoliko ne bi postojalo skladište između procesa, svaki pristigli *NoviZahtev* bi nakon *PrijemaZahteva* morao istog momenta da pređe u proces *ObradeZahteva*. Šta ako u trenutku prijema novog zahteva prethodni zahtev još uvek nije obrađen? Skladište omogućava da se paketi podataka između procesa obrade čuvaju, što omogućava da se procesi u jednom trenutku mogu paralelno izvršavati i obrađivati različite pakete podataka, što oslikava paralelizam odvijanja različitih procesa u jednoj organizaciji. U našem slučaju *PrijemZahteva* prima nove zahteve, dok u isto vreme proces *ObradaZahteva* obrađuje prethodno primljene, „nagomilane“ zahteve. U slučaju istih paketa podataka, skladište omogućava da se obrada paketa podataka između dva procesa vrši sa kašnjenjem. U primeru će primljeni zahtev neko vreme „čekati“ uskladišten da bi prešao u proces Obrade zahteva.

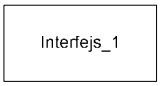
Drugačije rečeno, skladišta podataka na DTP služe da povežu dva procesa. Kad god izlazni tok jednog procesa predstavlja ulazni tok drugog procesa, potrebno je uvesti skladište podataka koje će premostiti vezu ta dva procesa.

Slučaj kada tok podataka iz jednog procesa ulazi u skladište se može tumačiti kao postupak upisivanja novih podataka, ažuriranja ili brisanja postojećih.

Slučaj kada tok podataka iz skladišta izlazi i ulazi u proces opisuje se kao mogućnost procesa da čita podatke iz tog skladišta, odnosno označava mogućnost pristupa tom skladištu[7].

3.3.2.4.Interfejs

Interfejs predstavlja spoljni objekat sa kojim sistem komunicira. Spoljni objekat može npr. biti osoba ili grupa osoba, korisnika sistema. Dalje, spoljni objekat može biti odeljenje unutar organizacije ili van nje – ili čitava eksterna organizacija. Primeri za to su respektivno: Odeljenje Računovodstva, Odeljenje za opšte poslove u Opštini, Banka i sl.). Informacioni sistem iz okruženja ili njegov deo takođe može predstavljati interfejs, sa kojim naš sistem komunicira (Sistem za online naplatu, Servis za pretraživanje internet-stranica). Interfejs se na dijagramu vizuelno predstavlja kao pravougaonik (Slika 8).



Slika 8 Interfejs

Interfejse je lako identifikovati. Ako sistem posmatramo kao „crnu kutiju“, odnosno kao jedan proces koji na ulazu prima tokove podataka, vrši proces obrade i generiše izlazne tokove, lako možemo uočiti sa kim on komunicira, odnosno od kog spoljnog objekta prima podatke, a kome obrađene podatke dalje isporučuje. Detaljnije o interfejsima u poglavlju 3.3.4.1 o Dijagramu konteksta.

3.3.3. Pravila i preporuke za kreiranje DTP

U prethodnom poglavlju su predstavljene komponente Dijagrama toka podataka: proces, tok podataka, skladište podataka i interfejs. Tom prilikom su uz svaki koncept navedena i određena pravila, kao na primer pravila za imenovanje koncepata. Za valjanu izgradnju dijagrama koristeći se ovim jednostavnim konceptima, potrebno je uvesti još nekoliko pravila. Pravila postavljaju ograničenja na određene postupke u kreiranju DTP, kako bi se izbeglo da rezultujući dijagrami budu pogrešni, nepotpuni ili logički nekorektni. Preporuke u smislu ograničenja treba „labavije“ shvatiti. Može se reći da pravila predstavljaju neophodan uslov, a preporuke dovoljan uslov za uspešnu izgradnju DTP.

Pravila i preporuke za kreiranje Dijagrama tokova podataka su sledeća:

(1) Svaki *proces* mora da ima barem jedan ulazni i barem jedan izlazni tok podataka. Proces bez ulaznog toka generisao bi izlaz ni iz čega, a proces bez izlaznog toka je nesvrsishodan ([6], str. 34). Postojanje takvih procesa automatski sugeriše na logičku grešku. Proces generisanja slučajnih brojeva bi bio jedini izuzetak kao proces koji poseduje samo izlazne tokove podataka. Nešto blaže ograničenje u ovom smislu, dalje, može da glasi:

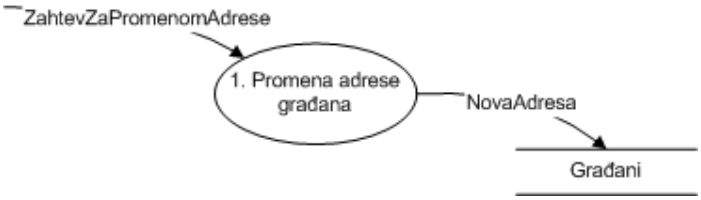
   i. Proces može da generiše izlaz samo iz svih za obradu potrebnih ulaznih tokova podataka[8], i obratno:

[7] Ako skladište unapred zamislimo kao bazu podataka, a proces kao softver koji radi nad bazom podataka, onda bi ulazni tokovi predstavljali ustvari procedure unosa podataka, ažuriranja ili brisanja postojećih podataka iz baze, a izlazne tokove kao standardne upite nad tabelama baze podataka.

ii. Proces može da generiše izlaz samo na osnovu za obradu raspoloživih tokova podataka.

(2) Preporučuje se da se jedan *proces* povezuje sa drugim procesom samo posredno preko skladišta podataka. Direktnu vezu dva procesa samo preko toka podataka trebalo bi izbegavati, jer se na taj način veoma ograničava njihovo nezavisno izvršavanje. Trenutak generisanja izlaznog toka prvog procesa automatski uslovljava početak izvršavanja drugog procesa. U velikoj većini slučajeva potreban je izvestan vremenski razmak, u kome će se izlazni paket podataka najpre sačuvati u skladištu podataka, odakle u proizvoljnom trenutku može biti „iščitan" od strane drugog procesa. (Pogledati poglavlje 3.3.2.3 o skladištu podataka). Pored toga, nepostojanjem skladišta podataka između dva procesa sistem se lišava mogućnosti da zapamti sopstveno međustanje (nakon završetka jedne i početka druge obrade), a to je jedna od osnovnih osobina skladišta podataka.

(3) Samo *tokovi podataka* koji idu ka, odnosno od skladišta podataka ne moraju biti imenovani. Ako tok između procesa i skladišta nije imenovan, podrazumeva se da tok nosi celokupan sadržaj i strukturu podataka tog skladišta. Ukoliko to nije slučaj, tj. ako tok sadrži samo deo strukture podataka, treba ga adekvatno imenovati. Primer na slici 9 iliustruje slučaj kada tok podataka ne prenosi sve podatke o Građaninu (Ime, Prezime, Pol, Datum rođenja, Adresa...) već samo jedan deo podataka, novu adresu.



Slika 9 Skladište sa imenovanim tokom podataka

(4) *Tokovi podataka* koji poniru u jedno skladište ili iz njega izviru, mogu da prenose samo one pakete podataka koji se u skladištu mogu čuvati.

(5) *Tok podataka* mora da ima izvor i ponor. Bilo koja druga komponenta DTP može da bude izvor ili ponor. Međutim, za jedan tok, bilo izvor, bilo ponor (bilo oba) mora da bude proces ([6], str.34). Iz ovog pravila sledi da dva interfejsa, dva skladišta, ili interfejs i skladište ne mogu direktno biti povezani tokom podataka (Slučaj procesa je prodiskutovan pod (2)). Razmotrimo detaljnije ove slučajeve:

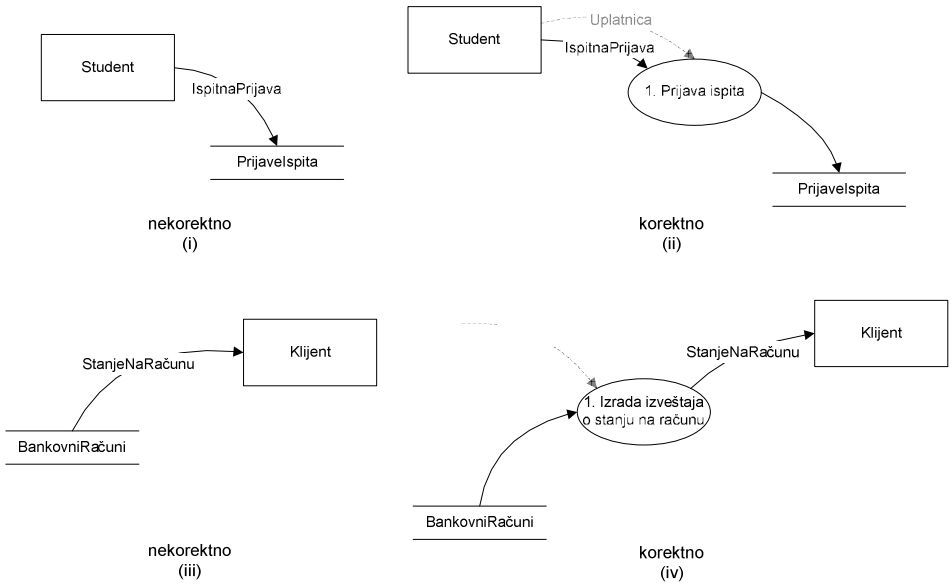
i. *Skladišta* ne mogu međusobno direktno biti povezana tokom podataka, jer skladišta, kao pasivne komponente, ne vrše nikakvu obradu.

ii. *Interfejsi* ne mogu međusobno direktno biti povezani tokom podataka, jer bi se na taj način opisivala komunikacija dva objekta van sistema koja, iako je moguća, nije od interesa za sistem koji posmatramo.

iii. *Interfejs* i *skladište* ne mogu direktno biti povezani tokom podataka jer bi to značilo da spoljni objekat direktno, bez kontrole internog procesa poseduje pristup skladištu podataka. Ovakve situacije se razrešavaju

[8] Proces ne može biti nosilac podataka, osim u slučaju konstantnih vrednosti. Na primer proces sračunavanja sumarne cene jedne porudžbine može kao konstantu procesa pamtiti visinu poreza na dodatu vrednost izraženu u procentima. Ipak, treba imati u vidu da svako pamćenje podataka kao konstanti čini IS „krućim" u odnosnu na buduće promene.

uvođenjem odgovarajućeg procesa između interfejsa i skladišta. U primeru prikazanom na slici 10(i), interfejs *Student* direktno pristupa skladištu podataka *PrijaveIspita*. Logičnije i sintaksno korektnije bi bilo uvođenje procesa obrade *Prijava ispita*, koji pretpostavlja kontrolu predate ispitne prijave, kao na primer i proveru uplate ispita (Slika 10(ii)). I za slučaj obrnutog smera toka podataka, od skladišta podataka ka spoljnom objektu, data je ilustracija nekorektnog i korektnog modelovanja na slikama 10(iii) i 10(iv), respektivno. Bez postojanja sistemskog procesa, koji bi najpre proverio identitet i prava klijenta, a zatim kreirao izveštaj o stanju na računu, klijentu bi bio omogućen direktan pristup celokupnom skladištu bankovnih računa.



Slika 10 Ilustracija pravila za izgradnju korektnih DTP

(6) Svako *skladište* mora da ima barem jedan ulazni i barem jedan izlazni tok podataka. U praksi je ponekad teško odmah uočiti da li jedno skladište stvarno ima i ulaz i izlaz. Često se za skladište najpre ustanovi samo jedan od tokova podataka, ali se kasnijom analizom drugog dela sistema utvrdi i ostatak tokova.

(7) *Interfejsi* moraju biti povezani sa sistemom, odnosno procesima sistema barem sa jednim ulaznim ili izlaznim tokom podataka.

(8) Preporuka vezana za preglednost dijagrama kaže, da se u cilju izbegavanja nepotrebnog presecanja linija bilo *skladište* bilo *interfejs* na jednoj slici može višestruko ponoviti. U tom slučaju potrebno je samo pored imena koncepta dodati znak „*“.

Prilikom izgradnje DTP treba imati i to u vidu, da prva skica dijagrama nikad ne može biti perfektna, imuna na sve sintaksne i logičke greške. Čovek ima sposobnost da razmišlja na iterativan način. Kroz ponavljanje bilo kog postupka on se uči i usavršava, kao prirodan proces učenja. Imajući to u vidu generalna preporuka koja se tiče izgradnje DTP glasi, da prvi put izgrađen dijagram uvek treba kroz nekoliko prolaza revidirati i u skladu sa pravilima i preporukama poboljšavati, čak i po cenu

odbacivanja stare i crtanja potpuno nove verzije. DeMarco u tom smislu kaže: „*Ukoliko se nađete u situaciji da ste proizveli konačnu verziju dijagrama na papiru na kom ste i započeli kreiranje, to samo znači da niste dozvolili vašem mozgu da razmišlja na prirodan, iterativan način*“ ([3], str.69).

Pored navedenih postoji još nekoliko pravila i preporuka koja se tiču sprovođenja pravilne hijerarhijske dekompozicije DTP, što je tema narednog poglavlja.

3.3.4. Dekompozicija Dijagrama tokova podataka

Uloga DTP se sastoji u tome da prikaže sve sistemske procese koji na bazi ulaznih tokova podataka vrše obradu i generišu izlazne tokove. Izvori i ponori svih tokova podataka jednog sistema su na DTP predstavljeni ili spoljnim objektima ili skladištima podataka.

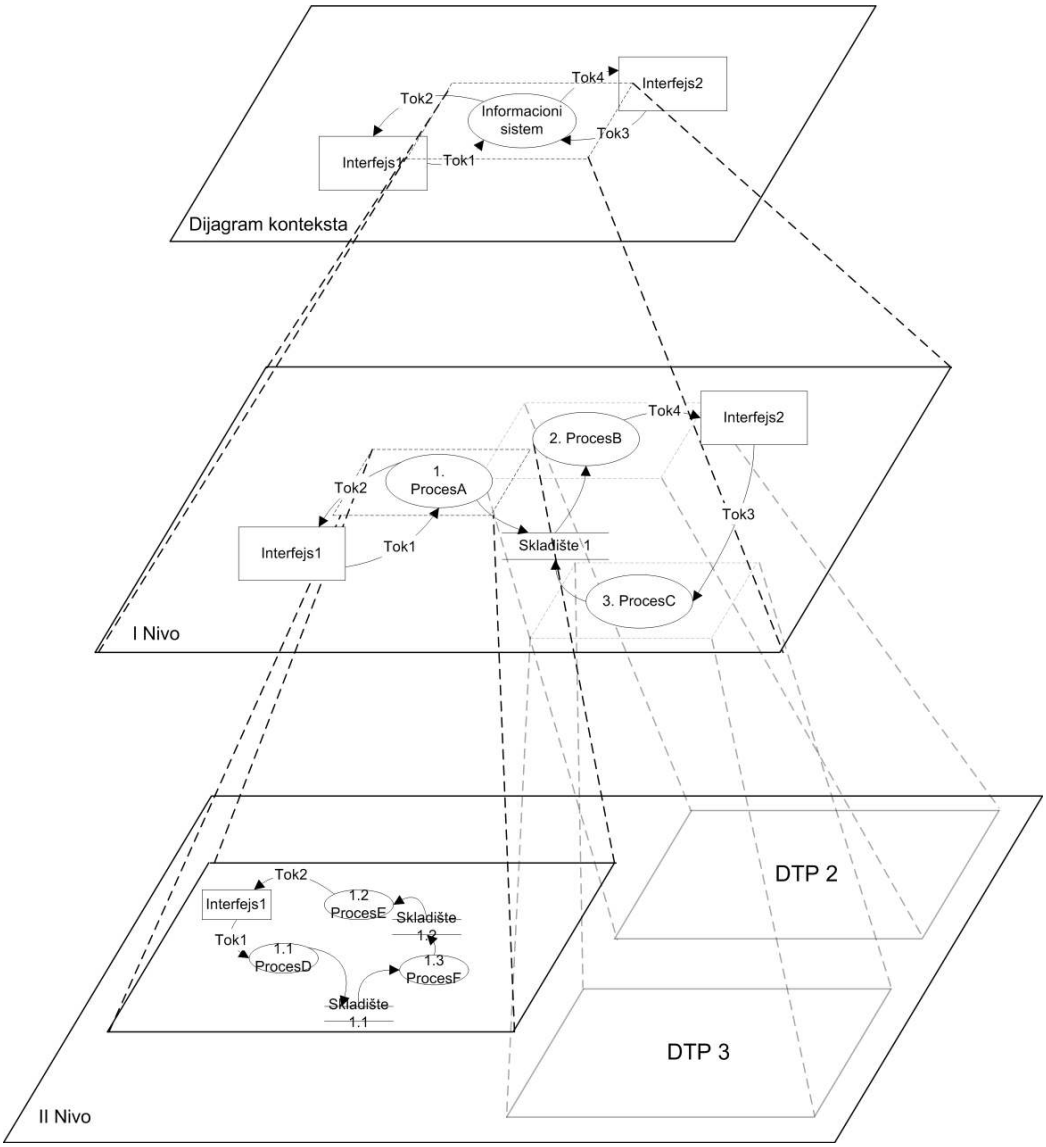
Veoma je važno da dijagrami tokova podataka budu pregledni, jasni i lako razumljivi, ne samo sistem-analitičaru, već i krajnjem korisniku. Način da dijagram ostane jednostavan je da sadrži relativno mali broj procesa uključujući sve potrebne interfejse, tokove i skladišta podataka. Ali, kako održati jednostavnost, odnosno mali broj procesa ako se želi detaljan prikaz složenog sistema koji se sastoji od nekoliko desetina, stotina ili hiljada procesa? Kada bi sistem takvih razmera predstavili samo jednim DTP, dobili bi neshvatljivu gomilu procesa isprepletanih mrežom tokova podataka nalik špagetima, a sam dijagram bi bio veličine bilborda.

Odgovor na ovo pitanje je jednostavan. Ceo sistem možemo najpre da podelimo na skup podsistema. Zatim svaki od podsistema dalje možemo podeliti na podpodsisteme. Ovaj princip postepenog razlaganja sistema na podsisteme možemo pratiti sve dok se razlaganjem još uvek dobijaju složeni procesi, odnosno dok ne stignemo do nivoa prostog, primitivnog procesa koji se dalje ne može deliti. Ako svaki nivo razlaganja sistema opišemo dijagramima, i to tako što ćemo na svakom nivou za svaki novodobijeni podsistem (skup podprocesa) kreirati odgovarajući dijagram, dobićemo hijerarhijski organizovan skup neuporedivo preglednijih, jasnijih DTP od prvobitnog „bilborda“[9].

Uvedeni postupak bazira se na osnovnom principu razrešavanja kompleksnih problema postepenim uvođenjem detalja, odnosno korišćenjem metode apstrakcije[10]. Dakle, tehnika izgradnje dijagrama tokova podataka se svodi na to da se na višim nivoima definišu globalniji procesi, a da se zatim svaki takav globalni proces, na sledećem nižem nivou, predstavi novim dijagramom toka podataka ([5], str.352). Princip dekompozicije funkcija koji se bazira na posmatranju sistema na „različitim nivoima apstrakcije“, ilustrovan je na slici 11.

[9] Slikovita analogija ovom postupku geografska karta, odnosno organizacija atlasa sveta. Atlas sveta započinje najpre slikom celog sveta podeljenog na kontinente. Zatim sledi opis svakog kontinenta pojedinačno dekomponujući ga na pojedinačne države. Zatim sledi opis unutrašnje organizacije države itd.

[10] Apstrakcija predstavlja postupak zanemarivanja detalja u cilju isticanja bitnih karakteristika posmatrane pojave.



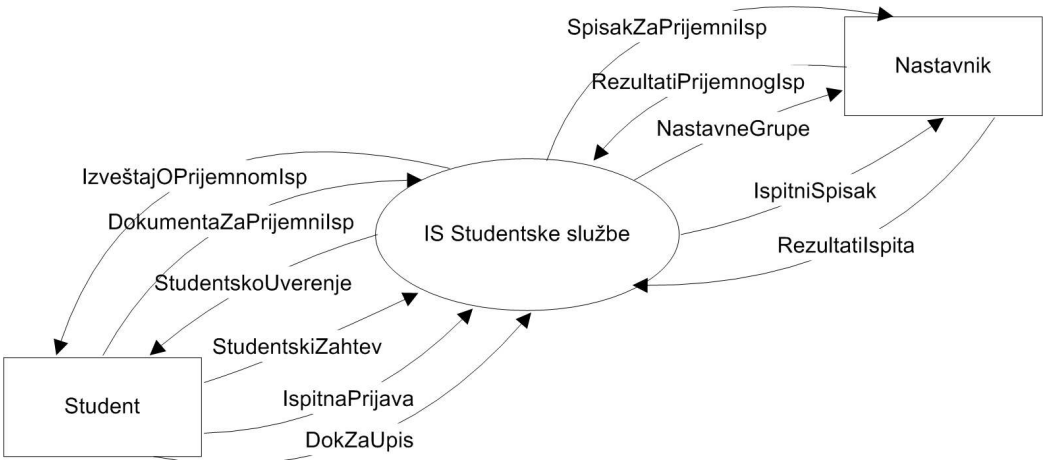
Slika 11 Princip hijerarhijske dekompozicije funkcija sistema

Dijagram tokova podataka na vrhu hijerarhije naziva se *dijagram konteksta*, a procesi na dijagramima najnižeg nivoa hijerarhije, koji se dalje ne mogu dekomponovati, nazivaju se *primitivni* procesi. Dijagram konteksta i primitivni procesi se detaljnije objašnjavaju u naredna dva podpoglavlja. Nakon toga se navode *pravila i kriterijumi dekompozicije*. Na kraju, postavlja se potreba da se svi procesi sistema dobijeni dekompozicijom ili jedan deo procesa predstavi na jedan sveobuhvatan, sumaran način. U tom smislu se uvodi Dijagram hijerarhijske dekompozicije funkcija, čime se okončava razmatranje dekompozicije DTP.

3.3.4.1.Dijagram konteksta

Najopštiji dijagram tokova podataka je dijagram konteksta. Dijagram konteksta posmatra sistem kao „crnu kutiju“, kao jedan proces, koji na bazi ulaza generiše izlazne podatke. Uloga dijagrama konteksta je da definiše kontekst analize, odnosno da odredi jasnu granicu sistema, da odgovori, šta su to sastavni delovi sistema, a šta su spoljni objekti sa kojima on interaguje. Zatim je potrebno identifikovati komunikaciju

između posmatranog sistema i njegovog okruženja, odnosno uočenih spoljnih objekata. Drugim rečima, potrebno je znati koje informacije iz okruženja utiču u sistem i koje informacije sistem proizvodi i šalje kao izlaz u okruženje.
Kao specijalan slučaj DTP, dijagram konteksta mora sadržati samo jedan proces koji predstavlja sistem, skup interfejsa i tokove podataka, koji od interfejsa ulaze u sistem i obratno, koji kao rezultat procesa obrade izlaze iz sistema ka interfejsima. Primer jednog Dijagrama konteksta dat je na slici 12 (Uporediti sa slikom 1 iz poglavlja 3.2.1).



Slika 12 Dijagram konteksta

S obzirom da *proces* na Dijagramu konteksta predstavlja ceo sistem, ime procesa obično predstavlja ime celokupnog informacionog sistema, kao što je na slici 12 slučaj. *Interfejsi* su povezani sa sistemom preko ulaznih i izlaznih tokova podataka. Važno je ponoviti da interfejsi ni u kom slučaju ne smeju međusobno direktno da komuniciraju. Ako ipak postoji povezanost dva interfejsa koja bi bila od interesa za sistem, onda je verovatno da će oni međusobno posredno komunicirati preko sistemskog procesa. Kada su identifikovani svi interfejsi, potrebno je uočiti sve relevantne *tokove podataka*.

Sistem interaguje sa okruženjem tako što reaguje na stimulanse iz okruženja. Na osnovu analize mogućih događaja iz okruženja na koje sistem reaguje i događaja koje sistem kao reakciju generiše, mogu se identifikovati neophodni tokovi podataka.. Dogadjaji koji deluju na sistem mogu biti spoljni dogadjaji koje izazivaju objekti iz okruženja, vremenski dogadjaji koji nastaju jer je nastupio odredjeni vremenski trenutak, ili interni događaji koji se „okidaju“ kada sistem dođe u neko specifično stanje ([6], str.31). Obično su spoljni događaji ti koji najviše govore o mogućim tokovima podataka. Na primer, za jedan IS Opštine spoljni događaj može biti „Stranka podnosi zahtev za izdavanje potvrde od državljanstvu“, na koji IS sistem treba da reaguje procesom obrade zahteva. Tako bi jedan ulazni tok podataka onda bio: „Zahtev za izdavanje potvrde o državljanstvu“, a izlazni tok podataka: „Potvrda o državljanstvu“. Događaj koji IS nakon obrade zahteva generiše bi mogao biti: „Potvrda o državljanstvu izrađena“ uz odgovarajući tok podataka. Primer vremenskog događaja može biti neki datum kada građanima treba uputiti određeno obaveštenje. U ovom slučaju događaj odgovara jednom izlaznom toku podataka ka građanima kao interfejsu. Na ovaj način se mogu efikasno identifikovati svi tokovi podataka i to za

svaki interfejs pojedinačno, čime se sumarno dobija kompletan skup potrebnih tokova podataka za ceo sistem.

Izgradnja dijagrama konteksta je najvažniji zadatak u postupku kreiranja DTP. Izgleda jednostavno nacrtati jedan proces, nekoliko interfejsa i povezati ih sa procesom tokovima podataka. Međutim, vrlo je važno uočiti sve moguće interfejse na samom početku, kao i sve događaje, odnosno tokove podataka koji ulaze u sistem i iz njega nakon obrade izlaze. Na primer, zanemarivanje jednog ili grupe tokova podataka bi u daljem postupku dekompozicije moglo da dovede do izostavljanja procesa koji bi u normalnom slučaju trebalo da vrše obradu podataka, kao i da prouzrokuje propuste u identifikaciji potrebnih skladišta podataka.

3.3.4.2.Primitivni procesi

Primitivni procesi su oni procesi koji se dalje ne mogu dekomponovati. Predstavljaju se Dijagramima tokova podataka na dnu hijerarhije, kao rezultat konačne dekompozicije funkcija sistema. Primitivni procesi se nazivaju još i atomskim procesima, što naglašava njihovu nemogućnost daljeg dekomponovanja. Za razliku od složenih procesa na višim nivoima hijerarhije, primitivni procesi se izvršavaju *sekvencijalno*, redosled aktivnosti u okviru njih je definisan.

Identifikovanjem svih primitivnih procesa jednog sistema završava se proces dekompozicije, pa samim tim i proces analize funkcija sistema, odnosno odgovora na pitanje šta sistem radi, odnosno koje funkcije sistem izvršava. Sledeći korak u metodi Strukturne sistemske analize jeste dati *specifikaciju logike primitivnih procesa*. Specifikacija logike primitivnih procesa, tzv. Mini-speficikacija ima zadatak da za svaki primitivni proces detaljnije opiše sekvencijalni postupak njegovog izvršavanja. Drugim rečima, mini-specifikacija svake funkcije treba da pruži odgovor na pitanje kako na osnovu ulaznih podataka transformacijom dobiti izlazne podatke. S obzirom da na taj način dobijamo odgovor na pitanje „Kako?“, sistem, shvaćen kao funkcija koja na bazi ulaznih podataka generiše izlazne, može da se posmatra kao „bela kutija“, jer je poznata ne samo njegova interna struktura, već i način funkcionisanja.

Postoji široka paleta tehnika za specifikaciju sekvencijalnih procesa, odnosno za opis logike primitivnih procesa. Najzastupljenije tehnike su dijagrami toka programa, strukturni jezici, pseudokod, tabele odlučivanja, stabla odlučivanja i dr. Specifikacija logike primitivnih procesa, odnosno tehnike za opis logike nisu od interesa, pa se nadalje ne razmatraju.

Kada govorimo o primitivnim procesima, postavlja se pitanje kako razlikovati složen proces od primitivnog, odnosno koji su to kriterijumi koji definišu kada prestaje dalja dekompozicija procesa. Sledeće poglavlje daje odgovor na to pitanje i navodi pravila za korektan postupak dekompozicije Dijagrama tokova podataka.

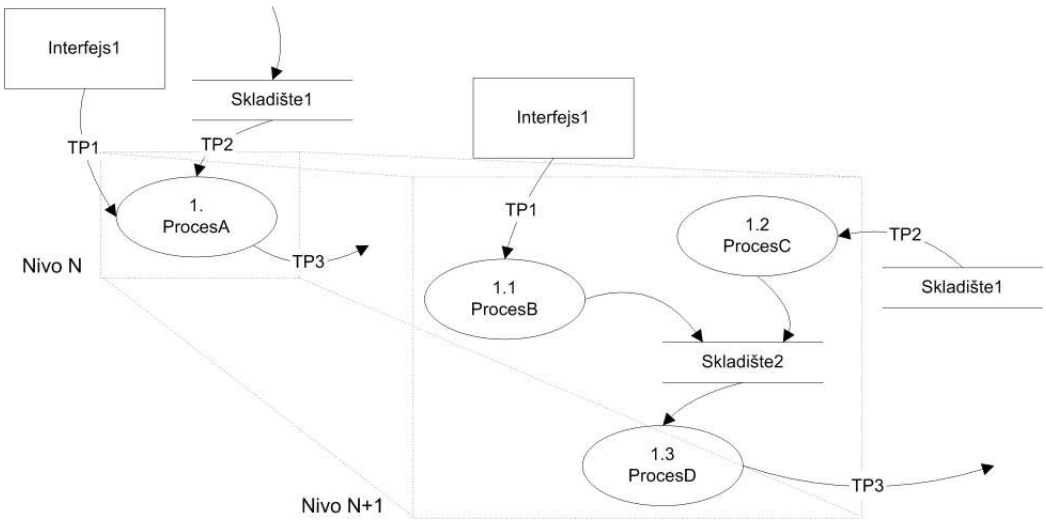
3.3.4.3.Pravila i kriterijumi dekompozicije

Kao što postoje pravila za izgradnju pojedinačnih Dijagrama tokova podataka, potrebno je pridržavati se i određenih pravila pri dekompoziciji procesa, odnosno pri hijerarhijskom opisu funkcija sistema. Pored toga potrebno je znati i pod kojim uslovima treba prekinuti dalju dekompoziciju procesa, a to znači prepoznati primitivne procese koji se dalje ne dekomponuju.

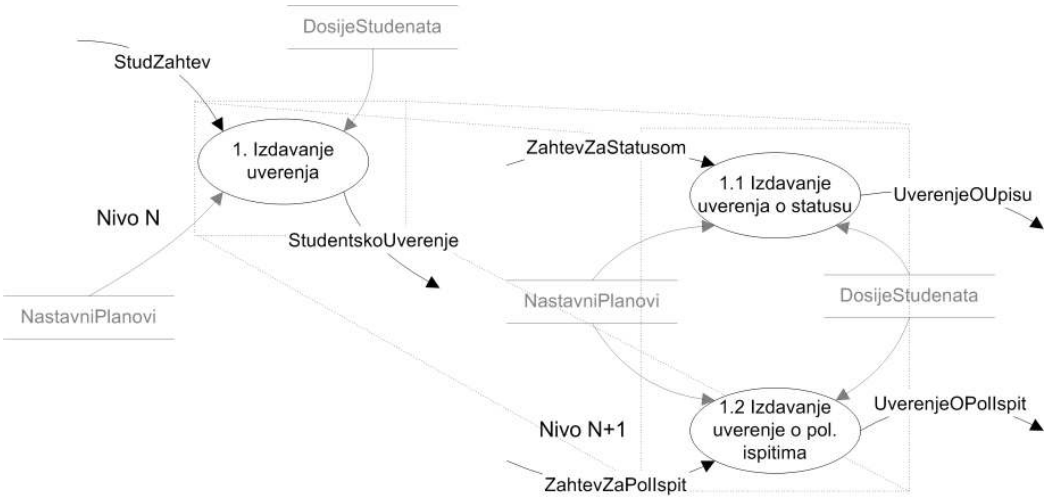
Kao što je to na slici 12 prikazano, na vrhu hijerarhije se nalazi Dijagram konteksta. Njegovom dekompozicijom se dobija tzv. *Prvi nivo dekompozicije*. Daljom dekompozicijom dijagrama prvog nivoa, dobijaju se dijagrami drugog nivoa čiji ukupan broj odgovara ukupnom broju procesa sa prvog nivoa. Postupak se sukcesivno nastavlja na sledećem nivou, sve dok se ne dođe do nivoa primitivnih procesa.

Sledeća pravila regulišu postupak dekompozicije:

- *Pravilo balansa tokova*. Najznačajnije pravilo koje se mora poštovati pri dekompoziciji procesa je pravilo balansa tokova: Ulazni i izlazni tokovi na celokupnom DTP-u koji je dobijen dekompozicijom nekog procesa P moraju odgovarati ulaznim i izlaznim tokovima toga procesa P na dijagramu višeg nivoa ([5], str. 354). Drugim rečima, svi tokovi koji ulaze i izlaze iz procesa moraju se pojaviti i na DTP sledećeg nivoa, koji taj proces dekomponuje. Slika 13 ilustruje pravilo balansa tokova. Tokovi podataka TP1, TP2 i TP3 procesa ProcesA identifikovani na nivou N, javljaju se i na DTP nivoa N+1, koji dekomponuje ProcesA. Međutim, moguće je odstupiti od ovog pravila u slučaju *dekompozicije samih tokova podataka*. Ukoliko je na višem nivou hijerarhije uveden tok podataka koji se suštinski sastoji iz više pojedinačnih tokova, na sledećem nivou se on može dekomponovati i time prividno narušiti balans tokova. Na slici 14 je dat primer dekompozicije tokova podataka *StudentskiZahtev* i *StudentskoUverenje* koji se dekomponuju na sledećem nivou u tokove *ZahtevZaStatusom*, *ZahtevZaPolIspit*, *UverenjeOUpisu* i *UverenjeOPolIspit*, respektivno. Dekompozicija tokova podataka se eksplicitno dokumentuje u Rečniku podataka (vidi poglavlje 3.3.4), čime balans tokova u hijerarhiji dijagrama ostaje nenarušen.



Slika 13 Opšti primer Pravila balansa tokova



Slika 14 Dekompozicija tokova podataka

- *Pravilo numerisanja procesa i dijagrama*: Procesi se u cilju lakše čitljivosti i preglednosti numerišu na sledeći način. Procesi na Prvom nivou dekompozicije se numerišu redom sa 1, 2, ...N. Svaki dijagram nižeg nivoa nosi oznaku procesa kojeg dekomponuje, a u njemu sadržani procesi na taj broj dodaju svoj jedinstveni broj. Dakle, dekompozicija procesa 1 predstavljena je dijagramom sa oznakom 1, i sastoji se od M podprocesa sa oznakama 1.1, 1.2, 1.3...1.M. Pravilo se nastavlja za svaki sledeći nivo hijerarhije daljim dodavanjem jedinstvenog broja procesa sa tog nivoa, 1.1.1, 1.1.2, ...
- *Skladišta podataka*. Skladišta podataka, sa sistemske tačke gledišta, pretstavljaju stanja sistema, odnosno fundamentalne, *unutrašnje* karakteristike kako celog IS, tako i svakog pojedinačnog procesa. Zbog toga se ona mogu pojaviti i na nižim nivoima dekompozicije, iako se nisu pojavljivala na prethodnim ([6], str.14). Pravilo za uvođenje novih skladišta glasi: *Skladišta se uvode po prvi put na onom DTP-u na kome predstavljaju vezu između dva ili više procesa. Nakon što su se pojavila na jednom nivou uz jedan proces, skladišta se moraju pojavljivati i na svim dijagramima nižeg nivoa koji dekomponuju taj proces* ([4], str.204).
- *Dodatak pravilima dekompozicije*. Nepisano pravilo pri dekompoziciji dijagrama glasi da se svaki proces može dekomponovati najviše u sedam podprocesa, tačnije da svaki dijagram ne treba da sadrži više od sedam plus ili minus dva procesa [11]. Veći broj procesa od ovoga mogao bi značiti da je „preskočen“ jedan nivo apstrakcije.

[11] Psiholog-matematičar George Miller je u svom radu, koji se bavi problemikom granica ljudske sposobnosti pri obradi informacija, došao do zaključka da ljudi mogu efikasno da obrađuju, odnosno prate u jednom trenutku sedam plus minus dva predmeta, odnosno koncepta. Preko tog broja se broj grešaka u obradi informacija disproporcijalno povećava, odnosno efikasnost opada. Tačan naziv publikovanog rada je: „The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information“ [7]. Ovaj zaključak bio je fundamentalan za razvoj svih strategija za segmentiranje, dekomponovanje problema na podprobleme, u cilju njegovog lakšeg savladavanja uz minimum grešaka.

Očigledno je da proces dekompozicije u jednom trenutku mora prestati. Prirodno, dekompoziciju prekidamo kada dođemo do primitivnih procesa. Ali, postavlja se pitanje, kada se za jedan proces može reći da je primitivan. Originalni autori metode SSA, Yourdon, DeMarco navode veoma opšte i neprecizne kriterijume pod kojima definišu trenutak prestanka dekompozicije, kao na primer: „*Dekompoziciju treba prekinuti onda kada se logika procesa može opisati mini-specifikacijom veličine ne veće od jedne stranice teksta*“ ([3], str. 83, 84). Još jedan primer navodi Yourdon: „*Često se odluka o daljem dekomponovanju utvrđuje kada zamolite korisnika da Vam objasni šta proces treba da uradi. Ako se korisnik najpre namršti, zatim duboko udahne i kaže – Pa, znate, to je jedna dugačka priča, ali pokušaću da Vam objasnim... – onda je vrlo verovatno da treba da nastavite sa dekompozicijom procesa*“ ([4], 433 – 441).

Nešto precizniji kriterijum glasi da dekompoziju procesa treba prekinuti u trenutku kada svaki proces poseduje samo jedan ulazni i jedan izlazni tok podataka. Izuzetak u ovom kriterijumu je postojanje malog broja primitivnih procesa, koji za generisanje jednog izlaza zahtevaju više od jednog ulaznog toka.

Navedeni kriterijumi predstavljaju korisne savete u definisanju kriterijuma za prekid dekompozicije procesa i pomažu u davanju odgovora na pitanje kada je jedan proces primitivan. Međutim, na ovo pitanje se mora dati suštinski odgovor.

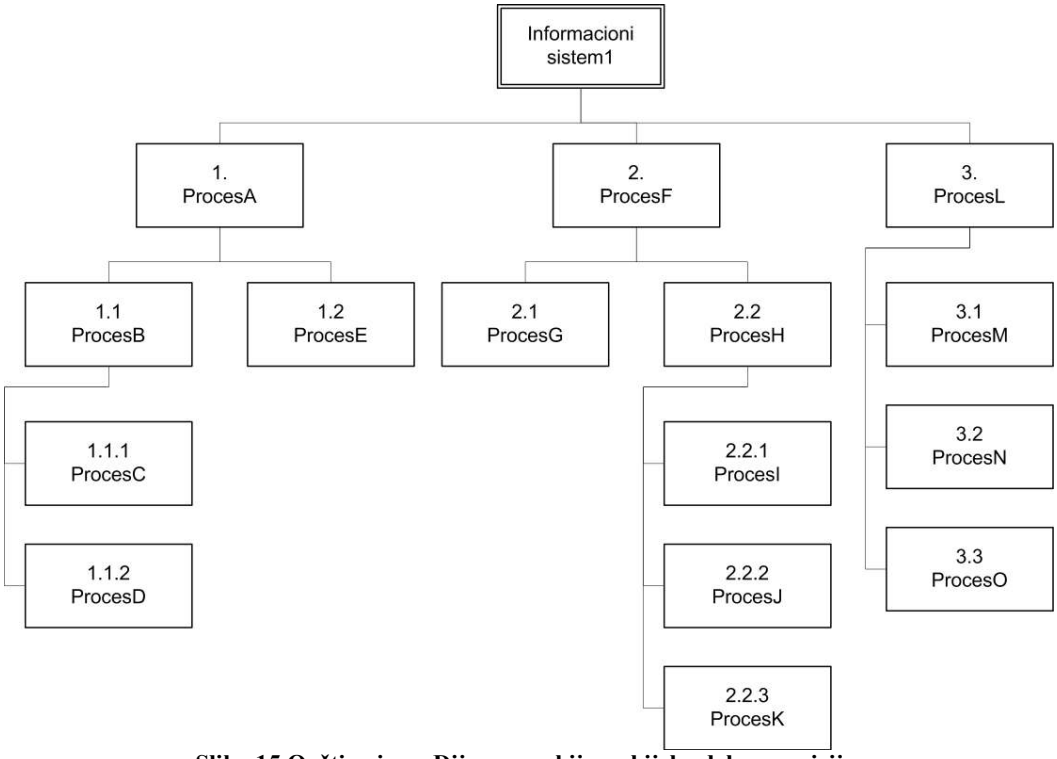
Podsetimo se da, sa jedne strane, izvršavanje procesa u organizaciji odlikuje paralelizam obavljanja funkcija (vidi poglavlje 3.3.2.1 o procesu obrade). Paralelni procesi se nazivaju *suštinski procesi* jedne organizacije. Uz to, paralelizam suštinskih procesa omogućen je postojanjem *suštinskih skladišta*, o čemu je bilo reči u poglavlju 3.3.2.3 o skladištima podataka. S druge strane, u poglavlju 3.3.4.3 o primitivnim procesima, rečeno je da se primitivni procesi izvršavaju sekvencijalno i da se opis njihove logike, umesto dekompozicijom, može opisati tehnikama za opis sekvencijalne logike (Pseudokod, Strukturni jezici i dr.). Upravo iz razlike u karakteru procesa koji se odvijaju u organizaciji, odnosno iz razlike između paralelnih i sekvencijalnih procesa Lazarević dopunjuje originalne autore i izvodi osnovni kriterijum dekompozicije u SSA:

Dekompoziciju treba vršiti dok se neka funkcija prirodno može dekomponovati na suštinske paralelne funkcije koje međusobno komuniciraju isključivo preko suštinskih skladišta. Drugim rečima, dekompoziciju treba okončati kada se dođe do procesa koji su prirodno sekvencijalni. I sami alati SSA podržavaju ovaj kriterijum - Dijagrami toka podataka za opis paralelnih procesa, a Pseudokod (Dijagram toka programa) za opis sekvencijalnih procesa ([5], str.361).

3.3.4.4.Dijagram hijerarhijske dekompozicije

Kao rezultat funkcionalne dekompozicije dobija se hijerarhijski uređen skup Dijagrama tokova podataka, kojim su postupno opisane sve funkcije jednog sistema. Očekivano je da sa povećanjem kompleksnosti sistema raste i broj kreiranih DTP, odnosno broj funkcija sistema. Jedan način za, tako reći, navigaciju kroz gomilu dobijenih dijagrama i procesa je praćenje hijerarhijske numeracije dijagrama i procesa o kojoj je bilo reči u prethodnom poglavlju. Ipak, javlja se potreba za jednom preglednom, a ipak dovoljno detaljnom, globalnom predstavom sistemskih funkcija.

U tom smislu se, za prikaz svih funkcija sistema ili njegovog dela koristi *hijerarhijski dijagram dekompozicije*. Slika 15 daje opšti prikaz jednog dijagrama dekompozicije.



Slika 15 Opšti primer Dijagrama hijerarhijske dekompozicije

Postupak izgradnje ovog dijagrama je više nego jednostavan. Dovoljno je pratiti hijerarhijsku numeraciju dijagrama i procesa i shodno tome, sastaviti hijerarhijski uređen skup svih procesa sistema ili pojedinih delova. Obično se u dokumentaciji specifikacije nekog sistema Dijagram dekompozicije koristi kao sadržaj za lakšu navigaciju kroz Dijagrame tokova podataka koji čine jednu specifikaciju. Uloga Dijagrama dekompozicije je analogna ulozi sadržaja jedne knjige, koji daje celokupan pregled sadržanih poglavlja.

3.3.5. Rečnik podataka

Dijagrami tokova podataka prikazuju sve procese jednog sistema izvedene postupnom dekompozicijom sistema, kao i pakete podataka, koji tokovima podataka cirkulišu u sistemu, odnosno koji bivaju sačuvani u skladištima podataka. Pored strukture procesa sistema dobijene detaljnom dekompozicijom, DTP u pogledu podataka daju samo uvid u to, koji podaci se u sistemu pojavljaju, bez detaljnijeg bavljenja njihovom strukturom, tj. *strukturom tokova podataka i skladišta podataka*. Baš kao i procesi, i tokovi podataka i skladišta mogu imati složenu strukturu, pa je potrebno dekomponovati ih na prostije elemente. Dekompozicija tokova podataka i skladišta se opisuje u *Rečniku podataka*. Rečnik podataka (RP) predstavlja alat za strukturirani opis podataka u sistemu, odnosno opis njihovog sadržaja i strukture. Opis je strukturiran zato što RP definiše poseban skup koncepata i pravila za opis podataka, odnosnu sintaksu za opis strukture podataka[12].

[12] Rečnik podataka metode SSA opisuje podatke baš kao što i bilo koji rečnik, npr. rečnik stranih termina i izraza opisuje sadržaj i značenje stranih pojmova. Za razliku od takvog rečnika u kome je

Sintaksa za opis strukture će biti objašnjena u predstojećem delu ovog poglavlja. Najpre se objašnjava koje su to osnovne, primitivne komponente strukture podataka i na koji način se one definišu. Zatim se uvode mogući oblici strukture složenih tokova podataka i skladišta.

3.3.5.1.Primitivne komponente strukture

Osnovne, primitivne komponente strukture podataka predstavljaju elementarni podaci, tzv. *polja*. Polja su osnovna komponenta strukture podataka jer predstavljaju podatke koji se dalje ne mogu dekomponovati. Primeri polja mogu biti: *Ime, Prezime, BrLičneKarte, Ocena, Cena, BrIndeksa* itd. Svako polje ima svoju vrednost. Skup iz kojeg jedno polje može da uzima vrednosti naziva se *domen*. Na primer, polja *Ime* i *Prezime* mogu da sadrže samo niz karaktera i to samo alfabeta, *Ocen*a i *Cena* uzimaju vrednosti iz skupa pozitivnih realnih brojeva itd. U primeru polja *Ocena*, u cilju preciznije definicije, potrebno je suziti skup vrednosti na podskup brojeva u intervalu od 5 do 10. *Ograničenje nad domenom* sužava skup mogućih vrednosti koje polje može da uzme iz domena. Po svojoj prirodi domeni mogu biti *predefinisani* ili *semantički* domeni. Predefinisani domeni su standardni, sveprisutni domeni kao što su skup celih brojeva, skup realnih brojeva, skup karaktera ili skup logičkih vrednosti (tačno/netačno). Semantički domeni daju novo ime predefenisanim domenima i obično kroz ograničenje definišu samo podskup predefinisanog skupa elemenata. Na primer, za polje *Ocena* može se definisati semantički domen *FakultetskeOcene* koji sužava skup prirodnih brojeva na interval od 5 do 10.

Uobičajeno je da se sva polja u Rečniku podataka, njihovi domeni i ograničenja prikažu preko odgovarajuće tabele. Tabela 1 i Tabela 2 daju primer za opis polja, domena, odnosno domenskih ograničenja.

POLJA		
NAZIV POLJA	DOMEN	OGRANIČENJE
ImeIPrezime	CHAR(30)	
Ocena	FakultetskeOcene	
VrstaStudija	VrsteStudija	
Cena	REAL(10,2)	>0
...	...	...

Tabela 1 Tabela polja

DOMENI		
NAZIV DOMENA	PREDEFINISANI DOMEN	OGRANIČENJE
FakultetskeOcene	INTEGER(2)	[5,10]
VrsteStudija	CHAR(20)	(„Osnovne“, „Magistarske“, „Doktorske“)
...	...	...

Tabela 2 Tabela semantičkih domena

opis neformalan, opisan prirodnim jezikom, Rečnik podataka SSA definiše odgovarajuću sintaksu kojom se dobija strukturan, formalan opis. I uloga RP je slična klasičnom rečniku, jer služi da nedvosmisleno definiše značenje podataka u sistemu opisujući njihov sadržaj i strukturu.

Kao što se u primeru vidi, za imena predefinisanih domena koriste se standardne oznake programskih jezika. Postoje različite notacije za opis polja, domena i ograničenja. Ono što je bitno je da jednom preuzetu notaciju treba konzistentno primenjivati. Jedna detaljna specifikacija notacije za opis polja, domena i ograničenja može se naći u [6].

3.3.5.2.Složene strukture podataka

Tokovi i skladišta podataka definisani na DTP obično imaju složenu, kompozitnu strukturu. Tipični paketi podataka sa složenom strukturom koji „teku“ kroz jedan IS su dokumenta kao što su: Katalog (filmova, proizvoda), Narudžbenica, Račun, Ispitna prijava, Uverenje, Cenovnik i sl.

Složene strukture podataka se sastoje iz više komponenata. Komponenta može biti elementarna, tj. polje i/ili neka druga definisana složena struktura. Na slici 16 je dat primer dokumenta Narudžbenice, koji se sastoji iz sledećih polja: *BrojNarudžbenice*, *ŠifraKupca, NazivKupca, DatumNaručivanja*. Pored polja, dokument Narudžbenice sadrži i jednu internu strukturu koju možemo nazvati *ListaArtikala*. *ListaArtikala* sa svoje strane sadrži sledeći skup polja: *RedniBroj, ŠifraArtikla, NazivArtikla, NarKoličina*.

NARUDŽBENICA

Broj:00123 **Datum**:12.12.2005

Podaci o kupcu

Šifra kupca: 009897342
Naziv kupca: SW-Inženjering D.o.o
Adresa kupca: Ulica inženjera b.b, Beograd, SCG

Lista artikala

Redni broj	Šifra artikla	Naziv artikla	Količina
01	A-00222-1	Monitor Siemlips L2335 23” TFT	6
02	C-30212-9	Projektor Canjitzu XEED SX 50	1
...	...	...	...

Slika 16 Dokument – Narudžbenica

Moguće su sledeće konstrukcije kojima se od komponenata (primitivnih ili složenih) gradi struktura ([6], str.18). Za svaku kontrukciju se navodi i način njenog predstavljanja u Rečniku podataka:

0. *Agregacija komponenti*. Agregacija predstavlja složenu strukturu u vidu liste N komponenti, bilo elementarnih, bilo složenih. Za njeno predstavljanje se koristi sledeća notacija: <K1, K2,...Kn.>, gde su K1, K2, Kn sastavne komponente strukture. Na primer, tok podataka *IspitnaPrijava* se zapisuje na sledeći način:

   IspitnaPrijava: <BrIndeksa, ImeStudenta, NazivPredmeta, DatumPolaganja, Ocena, ImeNastavnika>

1. *Specijalizacija (unija) komponenti*. Specijalizacija predstavlja strukturu u kojoj se bira jedna (*ekskluzivna* specijalizacija) ili više (*neekskluzivna* specijalizacija) od navedenih komponenti (Opcija). Komponente ekskluzivne specijalizacije se

navode unutar uglastih zagrada: [K1, K2,...Kn]. Primer specijalizacije sa ekskluzivnim izborom je skladište podataka *PoslovniPartneri*:

PoslovniPartneri: <SifraPP, NazivPP, AdresaPP, [ImeKontaktOsobe, Pol]>

Struktura PoslovniPartneri predstavlja agregaciju polja SifraPP, NazivPP, AdresaPP, i ekskluzivne specijalizacije polja ImeKontaktOsobe (u slučaju kada je poslovni partner pravno lice) i Pol (u slučaju kada je poslovni partner fizičko lice).

Komponente neekskluzivne specijalizacije se navode untar kosih zagrada: /K1, K2, ...Kn/. Primer specijalizacije u kojoj je moguće izabrati više od jedne komponente je složeni tok podataka Uverenje:

Uverenje: </ UverenjeOUpisu, UverenjeOPolIspit />

Fakultet može na zahtev studenta da izda bilo uverenje o upisu, bilo uverenje o položenim ispitima, bilo oba uverenja.

2. *Skup (Iteracija)*. Skup je struktura u kojoj se jedna komponenta ponavlja više puta, tj. iterira (kroz različite vrednosti komponente). Komponenta sa ponavljanjem se unutar neke strukture predstavlja vitičastim zagradama: {K1}. Primer za strukturu skupa je upravo dokument Narudžbenica sa Slike 16, koji u sebi pored elementarnih polja sadrži i složenu strukturu u vidu agregacije polja koja se višestruko ponavljaju:

   Narudžbenica: <BrojNarudžbenice, ŠifraKupca, NazivKupca, DatumNaručivanja, {<RedniBroj, ŠifraArtikla, NazivArtikla, NarKoličina>}>

   Ovakva struktura je vrlo česta i najlakše se prepoznaje kod dokumenata koji u sebi pored elementarnih polja sadrže i neki oblik nabrajanja u vidu tabele ili liste.

Na kraju poglavlja o Rečniku podataka recimo i to da u cilju izbegavanja redudanse, rečnik treba sadrži opis strukture svih tokova podataka i samo onih skladišta koja nisu skladišta već opisanih tokova. Pored toga, strukture tokova podataka opisane Rečnikom predstavljaju solidnu osnovu za detaljni opis podataka, odnosno izgradnju modela podataka, o čemu će biti reči u poglavlju o Modelovanju podataka, odnosno modelu objekti i veze.

3.3.6. Postupak modelovanja

U dosadašnjem izlaganju je predstavljen kompletan alat za opis funkcija sistema u metodi SSA. Takođe su navedena pravila, preporuke i konkretni postupci pri izgradnji modela procesa. Međutim, potrebno je prodiskutovati i opšti postupak modelovanja procesa metodom SSA, koji objedinjuje sve prethodno navedeno u jednu potpunu metodu za modelovanje funkcija sistema. S tim u vezi, postavlja se kao prvo opšte pitanje, koji se sistem uopšte analizira. Da li se modeluje postojeće stanje sistema, odnosno postojeća implementacija informacionog sistema neke organizacije ili se daje predlog buduće verzije implementacije IS? Da li treba izgraditi model funkcija sistema nezavisno od tehnologije, odnosno buduće implementacije ili sva tri

prethodna slučaja treba modelovati? U ovom poglavlju se prvo daje odgovor na ova pitanja, a zatim sumira do sada izloženo navođenjem jednog mogućeg postupka modelovanja.

3.3.6.1.Logički i fizički modeli procesa

Logički model procesa predstavlja model suštinskih procesa realnog sistema (organizacije) nezavisno od tehnoloških i organizacionih ograničenja. Podsetimo se da se suštinski procesi karakterišu paralelizmom u izvršavanju. Logički model treba da odgovor na pitanje *šta* budući sistem treba da radi da bi zadovoljio zahteve korisnika, odnosno da bi ispunio ciljeve organizacije. Pri izgradnji logičkog modela procesa sistem se posmatra *nezavisno od toga kako* je implementiran, tj. kao da je, ili će biti implementiran u idealnoj tehnologiji[13]. To između ostalog znači da pri analizi procesa ne treba uzimati u obzir da li taj proces izvršava radnik u organizaciji ili je proces automatizovan postojećim, odnosno budućim IS, bitno je samo identifikovati potrebu postojanja procesa.

S druge strane, *fizički model* predstavlja model procesa informacionog sistema, uzimajući u obzir sva ograničenja postavljena tehnološkim i organizacionim okruženjem pri implementaciji tog IS u organizaciji. Fizički model procesa koji analizira *postojeći* IS naziva se *„snimak" postojećeg stanja*. Fizički model procesa koji na bazi logičkog modela procesa uključuje i specifična tehnološka i organizaciona ograničenja budućeg sistema predstavlja *fizički model budućeg sistema*.

Cilj metode SSA je izgradnja logičkog modela procesa sistema specificirajući tačno šta sistem treba da uradi da bi zadovoljio organizacione ciljeve, ne razmatrajući kako će se sistem realizovati u specifičnom tehnološkom i organizacionom okruženju.

3.3.6.2.Modelovanje logičkog modela procesa

Klasičan postupak modelovanja procesa zagovaran u originalnoj specifikaciji metode SSA je takozvani postupak *Izdvajanja logičkog iz fizičkog modela procesa*. Ideja se bazira na tome da se prvo uradi „snimak" postojećeg stanja, odnosno izmodeluje postojeći fizički model sistema, da bi se zatim iz tog modela izvukli suštinski procesi i dobio logički model sistema, na taj način što se uklanjaju sve komponente modela koje predstavljaju rezultat tehnoloških i organizacionih organičenja. U kasnijim fazama se na bazi budućih ograničenja definiše fizički model budućeg sistema. Ovaj postupak se pokazao kao neefikasnim u analizi, jer transformacija postojećeg fizičkog u logički model dosta zahtevan zadatak i podložan subjektivnoj oceni analitičara. Više o tome u ([6], str. 29, [4], str. 373 – 379).

Jedan drugi postupak modelovanja, ovde od interesa, zagovara *direktnu* izgadnju logičkog modela procesa tzv. postupak d*irektnog modelovanja logičkog modela procesa* ([6] str. 30 - 34). Ovaj postupak podrazumeva dobro poznavanje vrste sistema koji se analizira (bankarski, proizvodni, fakultetski, administrativni sistemi, sistem trgovine i dr). Polazi se od nekog opšteg „teorijskog" poslovnog modela te vrste sistema (referentni model), a zatim se analiziraju karakteristike konkretnog sistema, da bi se opisali njegovi specifični zahtevi.

[13] Idealna je ona tehnologija koja je nezavisna od vremena i prostora, koja ne troši nikakve resurse i ne sadrži greške.

Imajući u vidu postupak direktnog modelovanja i sve do sada prethodno navedene konkretne postupke i pravila za izgradnju pojedinih komponenti modela, odnosno dijagrama, može se reći da se jedan *postupak modelovanja logičkog modela procesa metodom SSA* može podeliti u dve osnovne faze:

- *Opis okruženja sistema*. Opis okruženja sistema ima za cilj da definiše svrhu postojanja sistema i kontekst u kojem on funkcioniše, ne ulazeći u detalje kako sistem iznutra funkcioniše da bi ispunio ciljeve. To podrazumeva:
  - *Definisanje osnovnog cilja i svrhe postojanja sistema koji se modeluje.* Cilj je da se jednim kratkim, preciznim tekstom, ne većim od jednog pasusa, budućem korisniku (ili top-menadžmentu org.) predstavi svrha postojanja sistema. Obično se iz ovog teksta vidi i koji su spoljni objekti u interakciji sa sistemom.
  - *Izgradnja dijagrama konteksta (DK)*. DK je detaljno obrađen u ranijem tekstu (vidi poglavlje 3.3.4.3). Cilj: (1) identifikacija svih interfejsa, (2) za svaki od interfejsa, identifikacija svih događaja, odnosno informacija koje sistema prima i na koje reaguje odgovorom.
- *Opis ponašanja sistema*. Opis ponašanja sistema treba da pruži sliku o tome kako sistem iznutra funkcioniše da bi ispunio svoju svrhu. Razjašnjava se kako sistem obrađuje informacije koje prima i kako generiše izlazne informacije - sistemski gledano, kako sistem reaguje na događaje iz okruženja. To se ostvaruje kroz:
  - *Hijerarhijski opis DTP*. Započinje se sa dekomponovanjem DK na taj način što se utvrđuju opšti procesi koji obrađuju jednu grupu povezanih informacija, odnosno koji reaguju na grupu povezanih događaja iz okruženja. Neophodno je procesima obuhvatiti sve ulazne i izlazne informacije identifikovane u opisu okruženja sistema. Dalje se postupak nastavlja poštujući ranije opisana pravila. Međutim, treba na ovom mestu dodati da dekompozicija, ipak, ne mora biti striktno „top-down“ proces. Nekad je teško uočiti „big picture“ u startu. U tom slučaju treba početi od nekog „srednjeg“ nivoa na kome je slika jasnija i uočiti „sitne“ procese koji se zatim grupišu u opštije „krupnije“ procese, što takođe važi i za tokove podataka. I kada pratimo „top-down“ proces, često se dešava da analizirajući nivo N moramo da se popnemo na nivo N-1 i napravimo izmene, što nikako nije greška. Bitno je, pritom, voditi računa o balansu tokova. U svakom slučaju, kao krajnji rezultat se uvek dobije hijerahijski „top-down“ opis dijagrama.
  - *Opis podataka Rečnikom podataka(RP)*. Izgradnja Rečnika podataka može otpočeti paralelno sa izgradnjom DTP, kada se opisuju složeni tokovi i skladišta podataka. Međutim, tek nakon izgradnje DTP se i RP može kompletirati, tako što će se opisati svi elementarni tokovi i dekomponovati složeni i na kraju proveriti da li postoji redudansa u opisu i da li je RP u saglasnosti sa DTP.

- *Specifikacija logike primitivnih procesa.* Tek nakon što se opiše i logika svih primitivnih procesa može se reći da je potpuna specifikacija sistema metodom SSA gotova. Međutim, često se u praksi specifikacija logike ostavlja za sledeću fazu u razvoju IS ili se koriste drugi alati za opis, zbog čega mini-specifikacije nisu od značaja za ovo poglavlje.

Postoji još jedan postupak modelovanja, kao opšti pristup koji je se primenjuje za modelovanje složenijih poslovnih sistema i koji se bazira se na *Analizi životnih ciklusa osnovnih delatnosti i resursa u sistemu,* po čemu pristup nosi naziv. Dalja analiza ovog postupka prevazilazi okvire poglavlja, pa se čitalac upućuje na navedenu literaturu ([5], str. 358 - 361).

3.4. Ostali pristupi za modelovanje funkcija sistema

Pristupe za modelovanje funkcija sistema možemo podeliti na konvencionalne i savremene. Pored podele na konvencionalne i savremene, pristupi se međusobno razlikuju i po konkretnoj specifičnosti primene, jer modelovanje funkcija sistema se može koristiti, s jedne strane, za modelovanje poslovanja i s druge strane, za specifikaciju aplikacija. Ovim poglavljem će se samo ukratko navesti nekoliko najzastupljenijih konvencionalnih i savremenih metoda i pomenuti njihova specifična namena. Opširnije bavljenje ovim metodama za modelovanje funkcija sistema prevazilazi potrebe ovog udžbenika. U tom smislu se čitalac upućuje na navedenu listu šire literature.

3.4.1. Konvencionalni pristupi

SSA spada u red konvencionalnih metoda. Kao što je rečeno, SSA predstavlja metodu za analizu sistema i specifikaciju korisničkih zahteva, čime pruža osnovu strukturnog pristupa razvoju softvera. SSADM (*Structured Systems Analysis and Design Methodology* – SSADM) predstavlja specifičnu implementaciju originalne metode SSA i dugo godina je bila korišćena kao referentna metodologija za razvoj informacionih sistema u vladi Velike Britanije [8].

U red konvencionalnih, strukturnih metoda spada i IDEF0 (*Integration Definition for Function Modeling*), standard državnih organa SAD za modelovanje funkcija [9]. IDEF0 standard je baziran na metodologiji SADT (*Structured Analysis and Design Technique*), razvijenoj u kompaniji SofTech [10]. IDEF0 standard s jedne strane nalazi primenu u projektima analize, razvoja, reinženjeringa i integracije IS. S druge strane koristi se kao tehnika modelovanja poslovnih procesa u cilju analize, simulacije, reinženjeringa i dokumentacije poslovnih procesa organizacije. Slično kao i SSA, IDEF0 standardni jezik za modelovanje funkcija se karakteriše jednostavnim, grafičkim konceptima, hijerarhijskim postepenim dekomponovanjem funkcija, što doprinosi da modeli budu lako razumljivi i neinformatičarima, ali i dovoljno precizni za detaljnu specifikaciju IS.

Grupi konvencionalnih pristupa modelovanju funkcija sistema pripadaju i Petrijeve mreže, kao alat za modelovanje dinamike sistema[14]. Za razliku od DTP u metodi SSA,

[14] Petrijeve mreže potiču iz radova Karla Adama Petrija (*Carl Adam Petri*), koji je ovaj jezik formalizovao u svojoj doktorskoj tezi (*Komunikation mit Automaten*) ranih šezdesetih godina [11].

Petrijeve mreže modeluju *sekvencu izvršavanja funkcija* (opis logike izvršavanja funkcije u vremenu analogno Dijagramu toka programa). Klasične Petrijeve mreže su tokom godina pretrpele značajna proširenja kako bi odgovorile na zahteve modelovanja funkcija kompleksnih i velikih sistema. Mogućnost modelovanja podataka i vremena i koncept hijerahijske dekompozicije velikih modela su najznačajnija proširenja u tom smislu ([12], str. 41 - 48).

3.4.2. Savremeni pristupi

Savremene pristupe za modelovanje funkcija sistema možemo podeliti u dve grupe. Prvu grupu čine pristupi koji se prvenstveno koriste za specifikaciju softvera. Međutim, ovi pristupi se uz odgovarajuća proširenja mogu koristiti i za analizu poslovnih sistema, odnosno modelovanje funkcija sistema. Drugu grupu čine pristupi koji imaju specifičnu namenu u modelovanju poslovanja, odnosno modelovanju poslovnih procesa organizacije.

3.4.2.1.Pristupi za specifikaciju softvera

U prvu grupu pristupa spadaju *Modeli slučajeva korišćenja* i *Dijagrami aktivnosti*, koji predstavljaju komponente UML-a (*Unified Modeling Language*), standardnog jezika za analizu i dizajn u sklopu objektno-orijentisanog pristupa razvoju softvera [13].

- *Modeli slučajeva korišćenja (SK)* se originalno koriste kao alat za specifikaciju aplikacija[15], na taj način što se, kako samo ime kaže, za svakog korisnika uočavaju slučajevi korišćenja, odnosno funkcije koje aplikacija treba da pruži korisniku. Iako pronalaze primenu u modelovanju funkcija sistema, Modeli SK se pokazuju kao neadekvatna metoda za modelovanje složenijih sistema. Osnovni nedostatak ovog alata ogleda se u nepostojanju mehanizma dekompozicije funkcija, što predstavlja osnovni koncept za istovremeno jasno i detaljno opisivanje sistema ([5], str.367). Postoje i drugi pristupi koji proširuju koncepte Modela SK upravo u pogledu koncepta dekompozicije funkcija. Jedan takav pristup je UMM (UN/CEFACT (*United Nations Centre ofr Trade Facilitation and Electrionic Business*) Modelling Methodology) [14].
- *Dijagrami aktivnosti (DA)* kao standardna komponenta UML-a predstavljaju alat za opis logike slučajeva korišćenja. Kako samo ime kaže, dijagrami aktivnosti modeluju *redosled izvršavanja aktivnosti* unutar funkcija sistema i za tu svrhu poseduju moćan skup koncepata kao što su koncept odlučivanja, specifikacija paralelnog izvršavanja (grananje aktivnosti i sinhronizacija), plivačke staze (za prikaz izvršilaca aktivnosti) i dr. Koncepti definisani u DA vode poreklo iz prethodno pomenutih Petrijevih mreža. U slučaju modelovanja poslovnih procesa, za razliku od metode SSA i slučajeva korišćenja, DA omogućavaju detaljno modelovanje *sekvence* izvršavanja procesa, ali u isto vreme takođe „pate“ za eksplicitnim mehanizmom za dekomponovanje procesa. Međutim, UML je dovoljno opšt jezik koji konceptom stereotipa omogućuje da se njegovi standardni alati, kao što su Modeli SK i DA, prošire dodatnim konceptima specifične namene.

[15] Pojam aplikacije u ovom razmatranju treba razlikovati od pojma informacionog sistema. Informacioni sistem kao složeni sistem integriše više specifičnih aplikacija.

3.4.2.2.Pristupi za modelovanje poslovnih procesa

Druga grupa savremenih pristupa za modelovanje funkcija sistema orijentiše se isključivo na *modelovanje poslovnih procesa organizacije* u sklopu opštijeg menadžment koncepta *Menadžmenta poslovnih procesa* (BPM – *Business Process Management*). Sveobuhvatni koncept menadžmenta poslovnih procesa predstavlja pristup za strategijsko planiranje, upravljanje, izvršavanje i kontrolu poslovnih procesa uz podršku informacionih tehnologija (IT). Podrška IT se sastoji, s jedne strane, od softverskih alata za modelovanje poslovnih procesa, a sa druge strane, od informacionih sistema koji vrše automatizaciju poslovnih procesa. Srž ovog pristupa čini modelovanje poslovnih procesa, kao metoda koja omogućuje dokumentovanje, analizu, optimizaciju, izvršavanje i kontrolu procesa u organizaciji.

Jezici za modelovanje poslovnih procesa moraju biti jednostavni i orijentisani s jedne strane ka poslovnim korisnicima, neinformatičarima i analitičarima, a s druge strane dovoljno detaljni za opis kompleksnih procesa i formalni u cilju njihove implementacije pomoću IS. Sledeći pristupi definišu jezike za modelovanje poslovnih procesa u okviru opšteg koncepta menadžmenta poslovnih procesa.

- Događajem vođeni lanci procesa (EPC – *Event-driven Process Chains*) predstavljaju jedan jezik za modelovanje poslovnih procesa prvenstveno zastupljen u kompanijama zemalja nemačkog govornog područja[16] [15]. Jezik EPC se u osnovi takođe bazira na konceptima Petrijevih mreža. Osnovna ideja pri modelovanju procesa je sledeća: događaji u sistemu pokreću izvršavanje procesa, i obratno, procesi proizvode događaje u sistemu, čime se formira lanac procesa vođenih događajima. Tokom godina je ova metoda pretrpela poboljšanja u vidu proširenja za modelovanje organizacione strukture i uloga radnika, modelovanje podataka, kao i integraciju sa jezikom UML.
- ADONIS – standardni jezik[17], je metoda za modelovanje poslovnih procesa i još jedan predstavnik „nemačke škole“, a deo je sveobuhvatnog menadžment koncepta BMPS (*Business Process Management Systems*) [16][17]. Ovaj jezik nudi koncepte za modelovanje, dokumentaciju i optimizaciju kako procesa tako i strukture organizacionih sistema.
- LOVEM (*Line Of Visibility Enterprise Modeling*) je metoda za modelovanje poslovnih procesa razvijena unutar firme IBM [18].
- BPMN. Najnoviji i najznačajniji jezik za modelovanje poslovnih procesa u vidu standarda u oblasti menažmenta poslovnih procesa je BPMN[18] (*Business Process Modeling Notation*) [19]. Kao najnoviji standard, BPMN je razvijen sa ciljem da obuhvati sve prednosti ranijih pristupa u modelovanju poslovnih procesa sa posebnim osvrtom na jednostavne grafičke koncepte namenjene poslovnim korisnicima. Istovremeno, baziran na matematičkim formalizmima, BPMN pruža mogućnost izgradnje modela

[16] EPK (*Ereignissteuerte Prozesskette*) u originalnom nazivu, je jezik razvijen na Institutu za Poslovnu informatiku Univeziteta Saarland u saradnji sa softerskom kućom SAP za potrebe modelovanja poslovnih procesa.

[17] ADONIS standardni jezik je razvijen na Institutu za Poslovni i inženjenring znanja, Univerziteta u Beču.

[18] BPMN je inicijalno razvijen u okviru BPMI (*Business Process Management Initiative*) a trenutno je u fazi standardizacije u okviru OMG (*Object Management Group*), neprofitnog tela za specifikaciju standarda u oblasti informatike.

procesa koji se lako mogu transformisati u izvršni jezik podržan od strane specijalizovanih IS za automatizaciju poslovnih procesa.

Napomenimo još na kraju ovog razmatranja da su svi ovi pristupi za modelovanje funkcija sistema, kako konvencionalni, a posebno moderni podržani moćnim softverskim alatima za vizuelno modelovanje. Modeli poslovnih procesa dobijeni ovim alatima obično predstavljaju početnu kariku u lancu modela koji čine osnovu savremenog, modelima vođenog pristupa razvoju softvera.

3.5. Primer – studija slučaja

U primeru koji sledi daje se analiza sistema e-prodavnice primenom metode SSA. Logički model procesa ovog sistema rezultat je primene direktnog modelovanja, postupka modelovanja objašnjenog u odeljku 3.3.6.2. Kao teorijski model poslužio je klasični model trgovinskog poslovanja, koji je zatim prilagođen uzimajući u obzir specifičnosti online prodaje roba i usluga, kao vid e-poslovanja. Dati primer je „fiktivna“ studija slučaja, jer se ne odnosi ni na jedan realan slučaj, pa zbog toga ne uzima u obzir specifične karakteristike konkretnog realnog sistema.

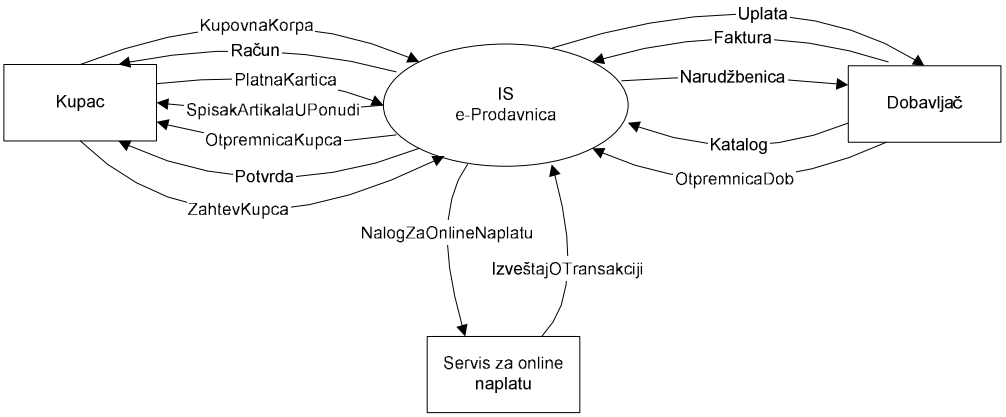
Studija slučaja pretpostavlja sledeću situaciju: Novoformirano trgovinsko preduzeće želi da izađe na tržište prodajom roba savremenim načinom poslovanja, putem interneta. Osnovna ideja leži u prodaji robe kupcima preko firminog veb-sajta, dok se nabavka proizvoda trenutno vrši klasičnim načinom kataloškog naručivanja od dobavljača. Prodaja se vrši na sledeći način: Kupac poručuje proizvode na veb-sajtu preduzeća tako što iz liste artikala u ponudi bira željene proizvode, tj. „puni“ kupovnu korpu, koju zatim predaju na naplatu. Naplata se vrši putem platnih kartica preko eksternog servisa za online-naplatu. Naplaćeni proizvodi se otpremaju poštom na adresu kupca. Radi sigurne kupovine, ali i prepoznavanja specifičnih zahteva kupaca, vodi se evidencija o kupcima.

3.5.1. Opis okruženja sistema

3.5.1.1. Speficikacija svrhe informacionog sistema

Svrha uvođenja informacionog sistema e-prodavnice sastoji se u informacionoj podršci poslovnog procesa nabavke proizvoda od dobavljača i poslovnog procesa online-prodaje proizvoda kupcima. Nabavka artikala podrazumeva obradu kataloga dobavljača, naručivanje artikala, njihov prijem i plaćanje. Prodaja proizvoda preko veb-sajta treba da registrovanim kupcima omogući poručivanje artikala preko kupovne korpe, naplatu porudžbine platnom karticom kupca preko eksternog sistema za naplatu i otpremu naplaćenih proizvoda.

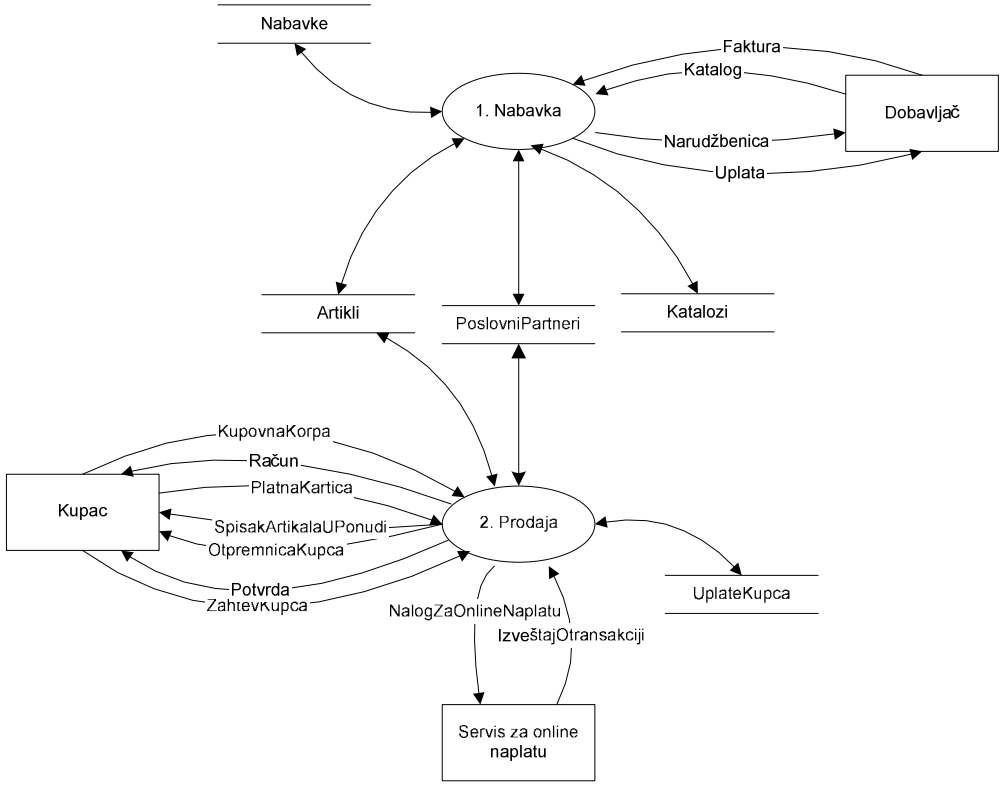
3.5.1.2. Dijagram konteksta



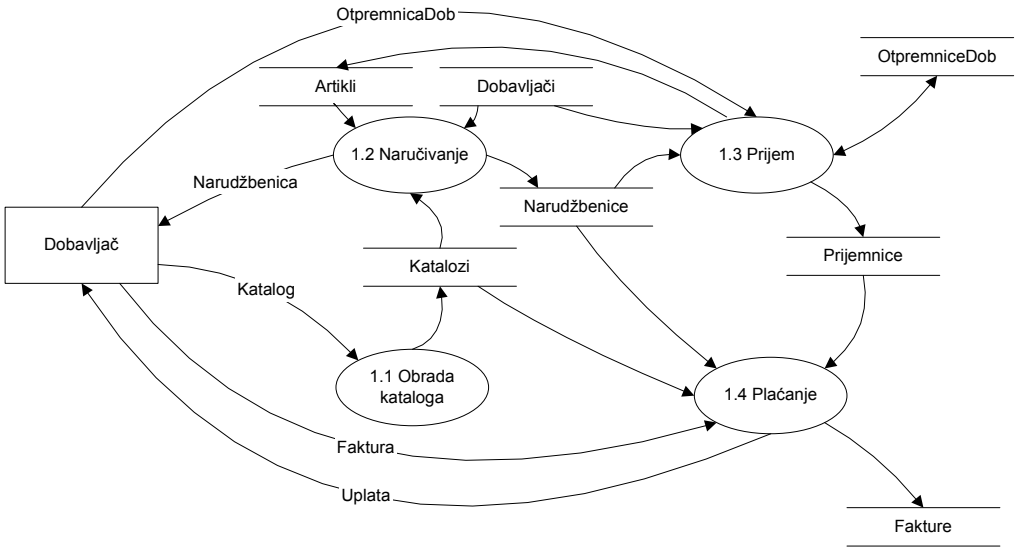
Slika 17 Dijagram konteksta IS e-prodavnice

3.5.2. Opis ponašanja sistema

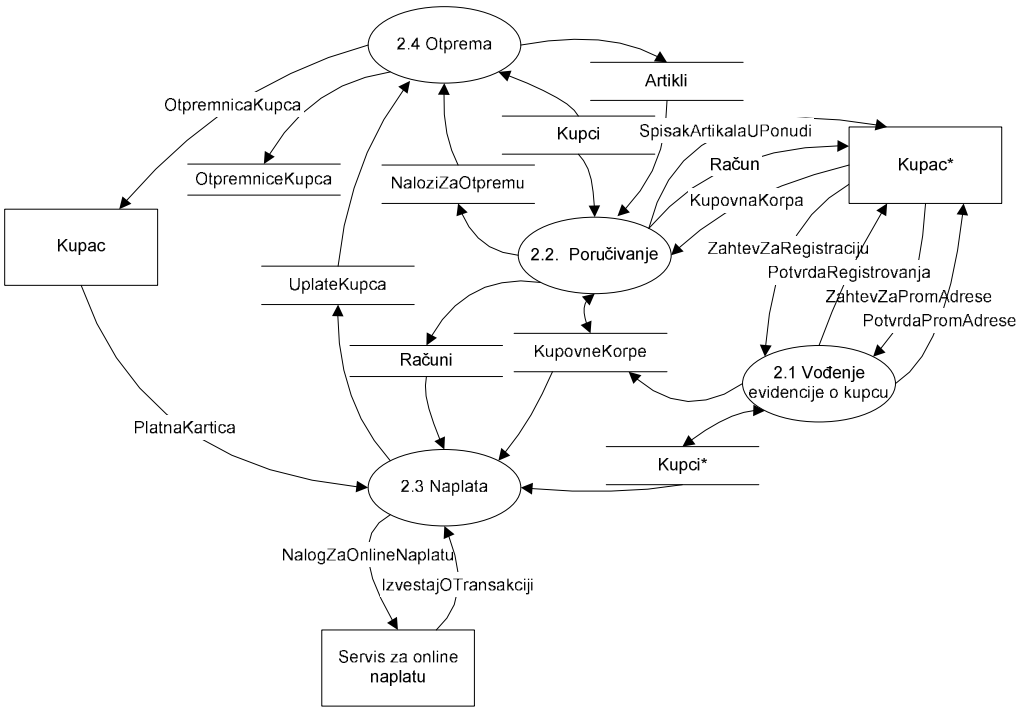
3.5.2.1.Dijagrami tokova podataka



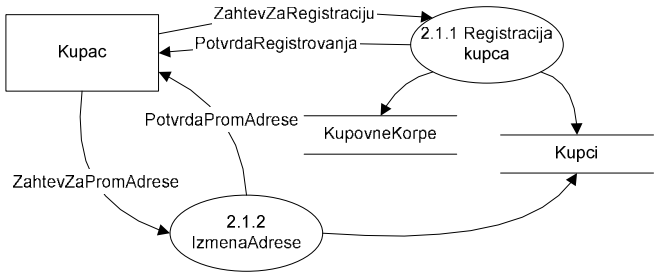
Slika 18 Dijagram prvog nivoa dekompozicije



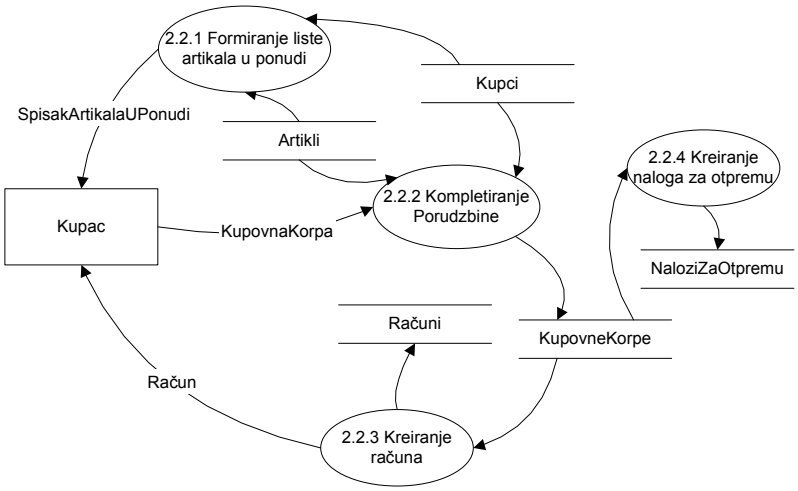
Slika 19 DTP 1 - Nabavka



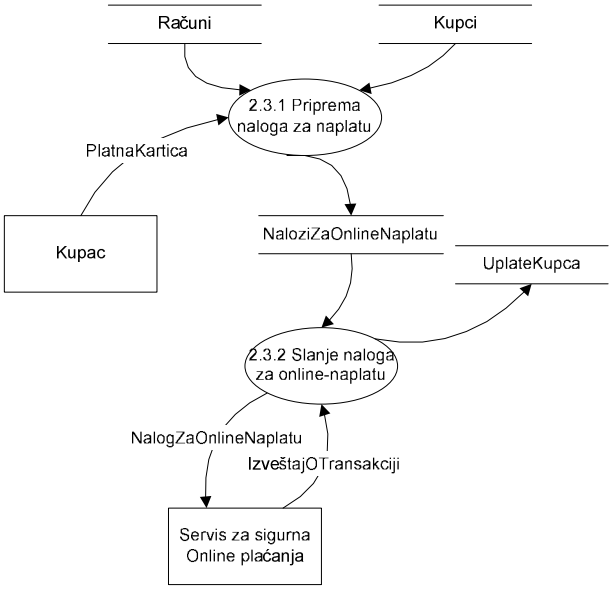
Slika 20 DTP 2 – Prodaja



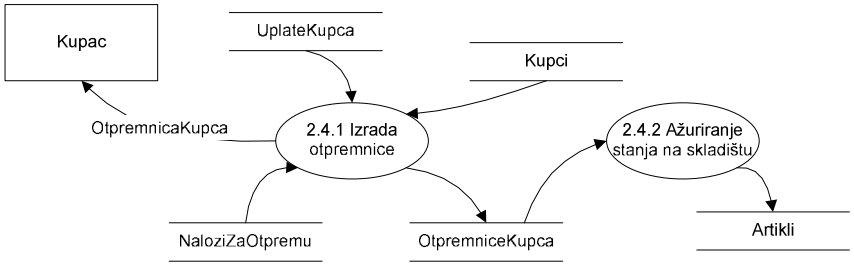
Slika 21 DTP 2.1 - Vođenje evidencije o kupcu



Slika 22 DTP 2.2 - Poručivanje

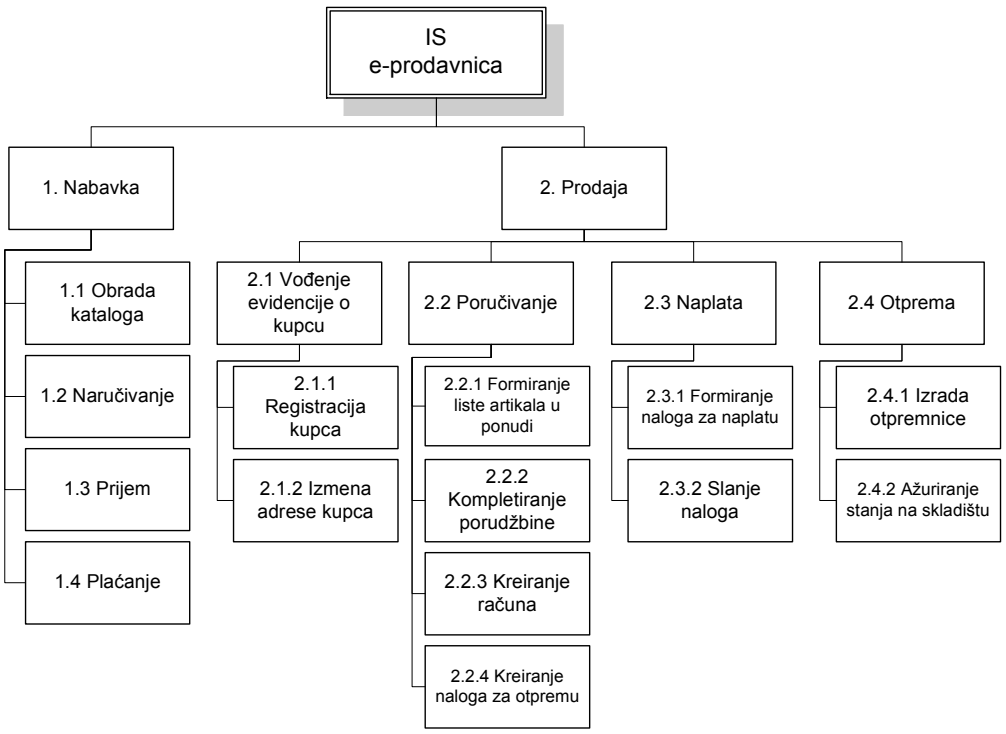


Slika 23 DTP 2.3 - Naplata



Slika 24 DTP 2.4 - Otprema

3.5.2.2.Dijagram dekompozicije



Slika 25 Dijagram dekompozicije IS "e-Prodavnica"

3.5.2.3. Rečnik podataka

OtpremnicaDob: <ŠifraDob, NazivDob, AdresaDob, BrojOtpr, DatumOtpr, BrojNar, DatumNar, {<RB, ŠifraArtikla, NazivArtikla, OtprKoličina, Vrednost>}>

Prijemnica: <BrojPrijem, DatumPrijem, ŠifraDob, NazivDob, AdresaDob, BrojOtpr, DatumOtpr, {<RB, ŠifraArtikla, NazivArtikla, PrimKoličina, Vrednost>}>

Katalog: <ŠifraDob, NazivDob, AdresaDob, BrKataloga, DatumIzdavanja, {<RB, {<ŠifraArtikla, NazivArtikla, OpisArtikla, CenaVP, CenaMP>}>

Narudžbenica: <BrojNarudžbenice, DatumNaručivanja, ŠifraDob, NazivDob, AdresaDob, {<RB, ŠifraArtikla, NazivArtikla, NarKol>}>

Faktura: <ŠifraDob, NazivDob, AdresaDob, BrojFakture, DatumFakture, BrojOtprem, DatumOtpr, OpisFakture, Iznos, RokPlaćanja>

Uplata: <BrojUplate, Datum, ŠifraDob, NazivDob, AdresaDob, BrojFakture, IznosUplate, RokPlaćanja>

PoslovniPartneri: <[Dobavljači, Kupci]>

Dobavljači: <ŠifraDob, NazivDob, AdresaDob, Delatnost>

Kupci: <ŠifraKup, ImeKup, PrezimeKup, AdresaKup, AdresaIspor, Telefon>

Artikal: <ŠifraArtikla, NazivArtikla, OpisArtikla, VrstaArtikla>

Račun: <BrojRač, DatumRač, ŠifraKupKorpe, DatumPorudžb, ŠifraKupca, ImeKup, PrezimeKup, AdresaKup, AdresaIspor, IznosRač, Napomena >

KupovnaKorpa: <ŠifraKupKorpe, DatumPorudžb, ŠifraKupca, ImeKup, PrezimeKup, AdresaKup, AdresaIspor, Telefon, {<RB, ŠifraArtikla, NazivArtikla, NarKol>}>

NaloziZaOtpremu: <ŠifraNalOtpreme, DatumNalOpreme, Opis, RokIsporuke, {<RB, ŠifraArtikla, NazivArtikla, NarKol>}>

PlatnaKartica: <BrPlatneKartice, DatumIsteka, TipKartice, ŠifraKup, ImeKup, PrezimeKup, AdresaKup, Telefon>
NalogZaOnlineNaplatu: <ŠifraNalOnlineNap, DatumNalOnlineNap, BrRačunaPrimaoca, BrPlatneKartice, DatumIsteka, TipKartice, ŠifraKup, ImeKup, PrezimeKup, AdresaKup, Telefon, BrojRač, DatumRač, IznosZaNaplatu, Napomena>
IzveštajOTransakciji: <ŠifraIzveštajaTrans, DatumTransakcije, ŠifraNalOnlineNap, DatumNalOnlineNap, BrRačunaPrimaoca, BrPlatneKartice, DatumIsteka, TipKartice, ŠifraKup, ImeKup, PrezimeKup, AdresaKup, Iznos, StatusTransakcije>
UplateKupca: <ŠifraUplateKup, ŠifraIzveštajaTrans, DatumTransakcije, ŠifraNalOnlineNap, DatumNalOnlineNap, ŠifraKup, ImeKup, PrezimeKup, AdresaKup, IznosUpl, Napomena >
OtpremnicaKupca: <ŠifraOtprKupca, DatumOtprKup, ŠifraUplateKup, DatumTransakcije, ŠifraKup, ImeKup, PrezimeKup, AdresaIspor, Telefon, ŠifraNalOtpreme, DatumNalOpreme, Opis, RokIsporuke, {<RB, ŠifraArtikla, NazivArtikla, OtprKol>}>
SpisakArtikalaUPonudi: <ŠifraSpiska, DatumIzdavanja, {<RB, ŠifraArtikla, NazivArtikla, OpisArtikla, Cena, Popust >}>
ZahtevKupca: <[ZahtevZaRegistraciju, ZahtevZaPromAdrese]>
ZahtevZaRegistraciju: <ŠifraZahtReg, DatumZahtReg, ImeKup, PrezimeKup, AdresaKup, AdresaIspor, Telefon, KorisničkoIme, Lozinka>
ZahtevZaPromAdrese: <ŠifraZahtAdr, DatumZahtAdr, ŠifraKupca, ImeKup, PrezimeKup, StaraAdresaKup, NovaAdresaKup>
Potvrda: <[PotvrdaRegistrovanja, PotvrdaPromAdrese]>
PotvrdaRegistrovanja: <ŠifraPotvrdeReg, DatumReg, ŠifraZahtReg, DatumZahtReg, ImeKup, PrezimeKup, AdresaKup, Status, OpisUslovaKorišćenja>
PotvrdaPromAdrese: <ŠifraPotvrdeAdr, DatumPromAdr, Status, ŠifraKupca, ImeKup, PrezimeKup, StaraAdresaKup, NovaAdresaKup >

3.6. Rezime

Strukturna sistemska analiza predstavlja jednu od metoda za analizu sistema, odnosno modelovanje funkcija sistema. U životnom ciklusu razvoja softvera se metoda SSA koristi u fazi analize sistema i specifikacije zahteva korisnika. Kao metoda modelovanja, ona jasno definiše osnovne koncepte za opis funkcija sistema i postupak modelovanja; drugim rečima, ona daje alat i uputstvo za njegovu upotrebu u cilju izgradnje modela. Osnovne karakteristike metode SSA su mali broj jednostavnih grafičkih koncepata i postupak hijerarhijske dekompozicije funkcija sistema predstavljenih dijagramima tokova podataka. Originalno ili u izmenjenoj formi, kao osnova drugih pristupa, ona predstavlja najšire korišćenu konvencionalnu metodu za modelovanje funkcija sistema.

Metode modelovanja funkcija sistema se sa jedne strane koriste za funkcionalnu specifikaciju softvera, a sa druge strane za modelovanja poslovanja, odnosno modelovanje poslovnih procesa u organizaciji. U domenu funkcionalne specifikacije softvera najznačajniji savremeni pristup definiše UML sa Dijagramima slučajeva korišćenja i Dijagramima aktivnosti. U domenu modelovanja poslovnih procesa, odnosno upravljanja proslovnim procesima najaktuelniji je BPMN standard.

3.7. Pitanja

1. Da li je poznavanje metoda za analizu sistema od važnosti isključivo informatičarima? Odgovor objasniti jednim primerom.
2. Šta je sistem? Opišite svojim rečima.

3. Navedite nekoliko primera za sisteme, sa kojima se svakodnevno susrećete. Opišite u grubim crtama sisteme na jedan od tri načina: verbalno, tekstualno, grafički. Definišite granicu sistema u odnosu na sisteme koji ga okružuju i veze sa njima, a zatim analizirajte u najgrubljim crtama njegovu strukturu, odnosno pronađite njegove elemente i osnovne veze između njih. Sa kog aspekta ste posmatrali sistem? Opišite sistem svom kolegi. Koji vam se način da se objasni sistem učinio najlakšim: verbalni, tekstualni, grafički?
4. Šta se podrazumeva pod pojmom analize sistema? Objasnite opšti postupak analize sistema.
5. Šta se podrazumeva pod pojmom analize sistema u informatici? Koje vrste sistema su pritom predmet analize sistema?
6. Definišite pojam modela i njegovu osnovnu svrhu?
7. Koja vrsta modela se izgrađuje u fazi analize sistema u procesu razvoja IS?
8. Šta se podrazumeva pod jednom metodom modelovanja?
9. Objasnite svojim rečima metodu Strukturne sistemske analize, njenu namenu, položaj u životnom ciklusu razvoja IS.
10. Opišite ukratko Dijagram tokova podataka. Kako se na DTP posmatraju procesi? Navedite osnovne koncepte koji se pojavljuju na DTP.
11. Na koji način se imenuju procesi na DTP? Koja od sledećih imena su ispravna: *Naručivanje proizvoda, Odeljenje prodaje, Ispitna prijava, Prijava ispita, Računovodstvo, Proračun ostalog*? Zašto je važno pravilo imenovati procese?
12. Čime se karakteriše izvršavanje procesa u organizaciji? Odgovor upotpuniti primerom.
13. Kako se tumači podatak predstavljen tokom, a kako skladištem podataka? U čemu se razlikuje imenovanje tokova u odnosu na imenovanje skladišta podataka?
14. Šta se predstavlja interfejsom? Navesti šta se obično može smatrati interfejsom jednog IS.
15. Da li proces na DTP može da ima samo ulazni, odnosno samo izlazni tok? Obrazložiti odgovor.
16. Zašto se ne preporučuje direktno povezivanje dva procesa tokom podataka?
17. Dva interfejsa, dva skladišta, ili interfejs i skladište ne mogu direktno biti povezani tokom podataka. Zašto?
18. Na koji način se održava jednostavnost u prikazu, preglednost DTP uprkos složenosti sistema, odnosno velikom broju procesa i drugih komponenata?
19. Objasniti postupak hijerhijskog opisa DTP? Na kom principu se bazira ovaj postupak?
20. Koja je uloga Dijagrama konteksta? U čemu se on razlikuje od ostalih DTP? Na koji način je moguće efikasno identifikovati komunikaciju sistema sa okruženjem?
21. Na vebsajtu neke kompanije pročitajte tekst o opisu kompanije i njenoj osnovnoj delatnosti (obično u odeljku „O nama“), a zatim pokušajte da skicirate Dijagram konteksta.
22. Šta su to primitivni procesi? Čemu služe mini-specifikacije primitivnih procesa?
23. Koje je najvažnije pravilo koje reguliše postupak dekompozicije DTP?
24. Kako se numerišu procesi na DTP? Da li numeracija procesa utvrđuje njihov redosled izvršavanja?
25. U kom slučaju je potrebno uvesti novo skladište podataka na DTP?

26. Da li je tačno da svaki DTP mora da sadrži sedam plus ili minus dva procesa?
27. Navedite sve kriterijume koji pomažu da se odredi kada je potrebno prestati sa dekompozicijom jednog procesa – Kako znamo da je jedan proces primitivan?
28. Šta prikazuje Dijagram dekompozicije?
29. Koja je uloga Rečnika podataka u metodi SSA?
30. Navedite primer polja koje uzima vrednosti iz semantičkog domena?
31. Šta čini strukturu jednog toka, odnosno skladišta podataka koji se opisuje u Rečniku podataka?
32. Koje se konstrukcije koriste za opis strukture tokova i skladišta podataka?
33. Šta znači saglasnost DTP i Rečnika podataka?
34. Kako se zove model koji procese opisuje nezavisno od tehnoloških i organizacionih ograničenja, a kako model koji ta ograničenja uključuje?
35. Koji od ova dva modela se izgrađuje u postupku modelovanja sistema metodom SSA? Kako se zove takav postupak?
36. Navedite postupak modelovanja metodom SSA?
37. Ako se „top-down“ dekompozicijom procesa stiglo do nivoa N, da li se smatra greškom vršenje izmena, tj. vraćanje na nivo N-1?
38. Kako se mogu podeliti svi pristupi za modelovanje funkcija sistema?
39. Koji je najzačajni savremeni pristup za modelovanje funkcija u smislu specifikacije softvera, a koji u smislu modelovanja poslovanja, tj. modelovanja poslovnih procesa?

3.8. Glosar

3.9. Korišćena i preporučena literatura

[1] Daenyer W.: *Systems Engineering*, 7. Auflage, Verlag Industrielle Organisation, Zürich, 1992.

[2] Yourdon, E., Constantine, L.: *Structured Design – Fundamentals of a Discipline of Computer Program and Systems Design*, 2nd ed, Yourdon Press, New York, 1978.

[3] DeMarco, T.: *Structured Analysis and System Specification*, Yourdon Press Computing Series – Prantice Hall, New Jersey, 1978.

[4] Yourdon, E.: *Moderne strukturierte Analyse: ein Standardwerk zur modernen Systemanalyse*, 1. Auflage, Wolfram's Fachverlag, Freiburg, 1992. – Titel der englischen Originalausgabe: *Modern Structured Analysis*, Prentice Hall, 1989.

[5] Lazarević B., Marjanović Z., Aničić N., Babarogić S.: *Baze podataka*, FON, Beograd, 2003.

[6] Lazarević B., et al: *Strukturna sistemska analiza*, Laboratorija za informacione sisteme, 1995.

[7] Miller G., *The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information*, Psychological Review, Vol. 63, 1956, str. 81-97.

[8] Goodland, M., Ashworth, C.: *An Introduction to SSADM Version 4,* McGraw Hill, 1993.

[9] *Integration Definition for Function Modelling (IDEF0)*, FIPS Publication 183, National Institute of Standards and Technologies, 1993.

[10] *An Introduction to SADT (Strctured Analysis and Design Technique),* SoftTech, Inc, Waltham, Mass, 1976.

[11] *Petri Nets World: Online Services for the International Petri Nets Community*, http://www.informatik.uni-hamburg.de/TGI/PetriNets/, odziv od 18.12.2005.

[12] Van der Aalst, W.M.P., Van Hee K.: *Workflow Management – Models, Methods and Systems,* MIT Press, Cambridge, MA, 2004.

[13] Booch, G., Rambaugh, J., Jacobson, I: *The Unified Modeling Language User Guide*, Addison – Wesley Object Technology Series, 1999.

[14] *UN/CEFACT Modeling Methodology*, United Nation Center for Trade Facillitation and Electronic Business, Technical Modeling Working Group, February 2001, http://www.unece.org/cefact/, odziv od 19.12.2005.

[15] Keller, G., Nüttgens, M., Scheer, A.-W.: *Semantische Prozeßmodellierung auf der Basis „Ereignisgesteuerter Prozeßketten (EPK),* Veröffentlichungen des Instituts für Wirtschaftsinformatik, Heft 89, Universität Saarbrücken, 1992, http://www.iwi.uni-sb.de/Download/iwihefte/heft89.pdf, odziv od 19.12.2005.

[16] Karagiannis, D.: *BPMS: Business Process Management Systems*, In: ACM SIGOIS Bulletin, Vol.16, Nr 1, 1995, S. 10-13.

[17] Kühn, H., Karagiannis, D.: *Modellierung und Simulation von Geschäftsprozessen*. In: WISU Das Wirtschaftsstudium, 30.Jg., Ausgabe 8-9, Lange Verlag, 2001, S. 1161 - 1170.

[18] IBM Corporation: *LOVEM – Line of Visibility Engineering Methodology – User's Guide, Version 3*, Third Edition, IBM Canada Ltd., 1999.

[19] *Business Process Modeling Notation – Version 1.0,* Business Process Management Initiative (BPMI), 2004. http://www.bpmn.org/, odziv od 13.04.2005.