Fakultet za kompjuterske nauke, Megatrend Univerzitet

Predmet:

UPRAVLJANJE INFORMACIJAMA I ZNANJEM

Predmetni nastavnik:

Prof. dr Branimir Trenkić

Materijal za pripremu ispita iz oblasti SSA (Strukturna Sistemska Analiza)

Beograd 2012.

UVOD – Proces razvoja informacionog sistema

Tema ovog materijala je Strukturna Sistemska Analiza (SSA) kao metodologije za potpunu, tačnu, formalnu i jasnu specifikaciju informacionog sistema. Ovu temu možemo posmatrati kao deo jedne šire teme kojom se mi bavimo u okviru ovog predmeta a to je razvoj informacionog sistema.

Proces razvoja informacionog sistema je vrlo složen proces i najčešće se obavlja u više faza. Kako podeliti jedan tako složen proces na faze (korake)? – za to u teoriji postoji više različitih pristupa. Jedna kategorija tih pristupa se bazira na fazama životnog ciklusa IS (informacionog sistema). Drugim rečima, podeliti sam process razvoja IS u faze koje odgovaraju životnim ciklusima samog IS. Na taj način dobijamo sledeće faze u razvoju iS:

1. Planiranje razvoja
2. Analiza i specifikacija zahteva
3. Projektovanje
4. Implementacija
5. Održavanje

Sami pristupi u okviru ove kategorije se razlikuju po tome kako je definisan međusobni odnos između ovih faza. Pa tako, između ostalih, postoje

1. Linearni pristup (konvencionalni model životnog ciklusa)
2. Prototipski (evolutivni model životnog ciklusa)
3. Transformacioni pristup

Svaka od navedenih faza ima svoje ciljeve (rezultate) i metodologije koje se primenjuju u datoj fazi kako bi se postigli rezultati koji su definisani u okviru te faze. U poslednje vreme, tendencija je standardizacija metodologija koje se koriste u pojedinim fazama radi ostvarenja cilja. Standardizacija metodologija koje se koriste u pojedinim fazama ima za cilj i formalizaciju prosedura koje se koriste kao i rezultata koji se dobijaju njihovom primenom. To omogućuje primenu odgovarajućih CASE alata na različitim nivoima procesa razvoja informacionog sistema. Tako, na primer, u realizaciji prve faze (Planiranje razvoja) vrlo često se koristi metod BSP (*Business System Planning*).

Tema ovog materijala je takva jedna standardizovana metodologija koja se može primeniti u različitim razvojnim procesima – za nas je interesantna njena primena u drugoj fazi životnog ciklusa informacionog sistema – analiza i specifikacija zahteva. Time, ovaj material obrađuje tematiku u okviru predmeta “Upravljanje informacijama i znanjem” na trećoj godini Osnovnih akademskih studija na Fakultetu za kompjuterske nauke koji se većim delom bavi procesom razvoja informacionog sistema sa nekoliko različitih aspekata.

1. U V O D

Strukturna sistemska analiza (SSA) je jedna potpuna metodologija za specifikaciju informacionog sistema, odnosno softvera. Ona se na različite načine može povezati sa metodama drugih faza u neku specifičnu metodologiju celokupnog razvoja IS. Tako na primer, ona može biti polazna osnova za metodu Strukturnog projektovana programa, ili projektovanja logičke strukture baze podataka metodom normalizacije, ili se može tretirati kao metodološki postupak dekompozicije nekog sistema na podsisteme sa ciljem da se, nalaženjem modela podataka podsistema i njihovom integracijom, dođe do potpunog modela podataka posmatranog sistema. Upravo zbog mogućnosti njene raznovrsne primene, metoda SSA se ovde tretira kao jedinstvena, samosvojna metoda, dok se u drugim materijalima pokazuje kako se ona koristi u pojedinim koracima Standardne metodologije razvoja informacionih sistema.

Potpuna, tačna, formalna i jasna specifikacija IS, ili kako se to obično kaže, specifikacija zahteva korisnika, zahteva koje budući sistem treba da zadovolji, predstavlja bitan preduslov za uspešno dalje projektovanje i implementaciju sistema. Očigledno je zbog čega specifikacija IS treba da bude potpuna i tačna. Zahtev da specifikacija bude formalna iskazuje se zbog toga što je formalna specifikacija osnov za "transformaciono" projektovanje i implementaciju, za atomatizovano generisanje baze podataka i programa iz nje, odnosno za korišćenje CASE sistema. Zahtev da specifikacija bude jasna iskazuje se zbog toga što u specifikaciji IS u velikoj meri učestvuju korisnici sitema, neinformatičari, pa jezik specifikacije mora biti i njima prihvatljiv. Originalna SSA čiji su tvorci Yourdon i njegovi saradnici (DeMarco i drugi) poseduje veoma jednostavne, grafičke, pa samim tim i jasne koncepte. Ovde su svi ovi koncepti zadržani, a strožija formalizacija je dodata samo za opis strukture tokova i skladišta podataka, da bi se obezbedio specifičan transformacioni razvoj IS koji Standardna metodologija zagovara.

Kao što je već ranije rečeno, specifikacija IS treba da prikaže (potpuno, tačno, formalno i jasno) šta budući informacioni sistem treba da radi. Veoma je bitno odmah istaći da specifikacija IS prikazuje **ŠTA** IS treba da da, a ne i **KAKO** to treba da ostvari. Očigledno je da prerano definisanje "kako", odnosno davanje nekih projektantskih rešenja u okviru specifikacije, ograničava kasniji mogući izbor (optimizaciju) načina implementacije sistema. Odgovor na pitanje "**kako**" daje se za konkretno okruženje, za definisanu tehnologiju i organizaciju u kojoj se sistem implementira. Da specifikacija ne bi sadržala tehnološki i organizaciono ograničena rešenja, obično se kaže da ona treba da opiše funkcionisanje IS u "idealnoj tehnologiji", gde praktično nikakva ograničenja ne postoje. Ako je specifikacija ovako zadata, onda je, pre prelaska na dalje projektovanje, neophodno da se definišu sva ograničenja koja nameće okolina u kojoj se sistem implementira.

Zbog toga specifikacija IS treba da poseduje sledeća dva dela:

(**I**) funkcionalnu specifikaciju u kojoj se opisuje budući IS u "idealnoj tehnologiji" i

(**II**) nefunkcionalnu specifikaciju koja definiše sva ograničenja implementacione okoline.

SSA u potunosti obuhvata samo funkcionalne specifikacije, dok nefunkcionalne samo delimično pokriva prikazujući tokove podataka u novoimplementiranom sistemu. Ostali deo nefunkcionalnih specifikacija obično predstavlja samo nabrajanje zahtevanih performansi budućeg IS i ograničenja implementacione okoline.

SSA posmatra informacioni sistem kao funkciju (proces obrade) koja, na bazi ulaznih, generiše izlazne podatke. Ulazni podaci se dovode u proces obrade, a izlazni iz njega odvode preko tokova podataka. Tok podataka se tretira kao vod ili kao pokretna traka kroz koji stalno teku ili 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. Imajući u vidu zahtev da specifikacija treba da se oslobodi svih implementacionih detalja od interesa su samo sadržaj i struktura ulaznog toka, a ne i medijum nosilac toka.

Izvori ulaznih, odnosno ponori izlaznih tokova podataka mogu biti objekti van IS koji sa IS komuniciraju i koji se u SSA nazivaju interfejsi, drugi procesi u sistemu, ili tzv skladišta. Skladišta podataka se posmatraju kao "tokovi u mirovanju", odnosno odloženi, akumulirani tokovi, različite vrste evidencija, arhiva, kartoteka i datoteka. I za skladišta kao i za tokove od interesa su isključivo njihov sadržaj i struktura.

Osnovni koncepti za specifikaciju IS u SSA su, znači, funkcije, odnosno procesi obrade podataka, tokovi podataka, skladišta podataka i interfejsi. Njihov međusobni odnos se prikazuje preko dijagrama toka podataka koji prikazuje vezu interfejsa, odnosno skladišta kao izvora odnosno ponora podataka, sa odgovarajućim procesima, kao i međusobnu vezu procesa. Na slici 1 prikazan je jedan opšti primer dijagrama toka podataka koji ima za cilj i da uvede sledeće grafičke simbole:

(**I**) krug ili elipsa pretstavlja funkciju ili proces obrade podataka,

(**II**) pravougaonik predstavlja interfejs,

(**III**) usmerena linija predstavlja tok podataka,

(**IV**) dve paralelne linije ("otvoreni" pravougaonik) predstavlja skladište podataka.

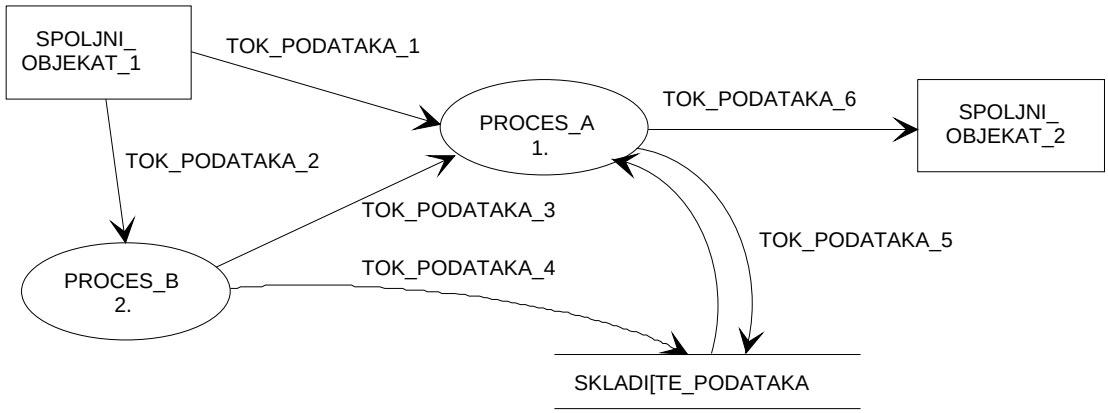
Očigledno je da se jedan IS sastoji iz mnoštva procesa, interfejsa, tokova i skladišta podataka. Specifikacija IS treba da bude potpuna (detaljna) i jasna. Kada bi se jedan sistem detaljno opisao i prikazao jednim dijagramom toka podataka, dobio bi se veoma nejasan opis sistema, paukova mreža procesa, tokova, skladišta i interfejsa. Istovremeno detaljan i jasan opis sistema zahteva opis na "različitim nivoima apstrakcije", odnosno hijerarhijski opis u kome se na višim nivoima sistem opisuje opštije, a na nižim, postepenim i organizovanim uvođenjem detalja, potpuno i detaljno. Hijerarhijski opis sistema u tehnici 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, pretstavi novim dijagramom toka podataka.

Dijagram toka podataka na vrhu ovakve hijerarhije naziva se dijagram konteksta, a procesi na najnižem nivou (procesi koji se dalje ne dekomponuju) nazivaju se primitivni procesi.

Imajući u vidu sve rečeno, jednu potpunu specifikaciju IS čine:

(1) Hijerarhijski organizovan skup dijagrama toka podataka;
(2) Rečnik podataka koji opisuje sadržaj i strukturu svih tokova i skladišta podataka;
(3) Specifikacija logike primitivnih procesa;

Nadalje, u sledeća tri poglavlja detaljno se opisuju ove tri komponente, odnosno tri osnovna alata SSA. Međutim metodologiju specifikacije IS ne čine samo alati i tehnike opisivanja budućeg IS. Metodologija treba da obuhvati i mnogo složeniji aspekat, metode i postupke kojim se do specifikacije IS, preko pomenutih alata, može da dođe. Zato se u petom poglavlju diskutuju metode strukturne sistemske analize.



Slika 1. Osnovni koncepti DTP

2. DIJAGRAMI TOKA PODATAKA

U prethodnom delu su definisani osnovni koncepti dijagrama toka podataka (Slika 1). Dijagram toka podatka (DTP) predstavlja model sistema koji sadrži četiri osnovne komponente: procese obrade podataka (aktivne komponente sistema), objekte okruženja (interfejse) sa kojima sistem komunicira, skladišta podataka koje procesi koriste i/ili ažuriraju i tokove podataka koji povezuju ostale komponente sistema u celinu.

Osnovne karakteristike DTP-a su:
- jasna grafička specifikacija, pogodna za komunikaciju sa korisnikom,
- istovremeno jasan i detaljan opis sistema, primenom metode apstrakcije tako da se sistem na višim nivoima apstrakcije opisuje uopšteno, a na nižim detaljno.

2.1. Pravila kreiranja (sintaksa) dijagrama toka podataka

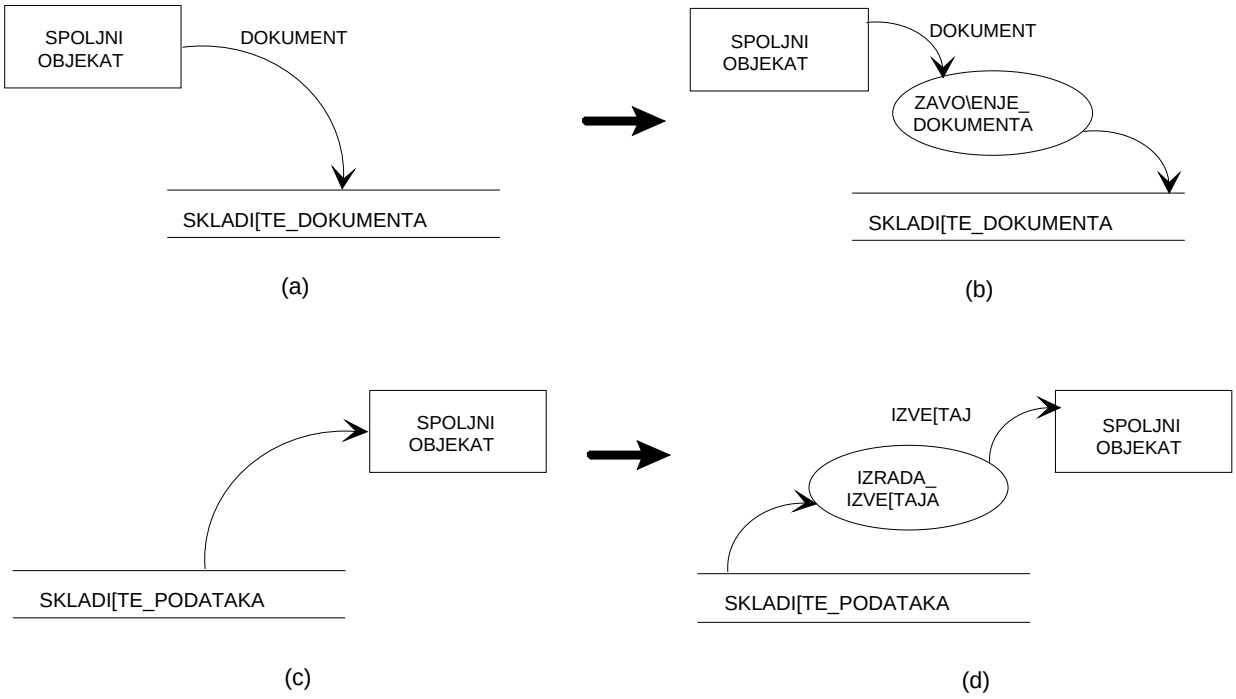
Pri kreiranju dijagrama tokova podataka moraju se poštovati sledeća pravila:

(**I**) Kao što je rečeno, tok podataka se može zamisliti kao pokretna traka u fabrici, ili cev kojom do skladišta podataka, procesa ili spoljneg objekta teku struktuirani podaci (različita dokumenta, formulari, tekstovi, knjige, časopisi i slično). Tok podataka ostvaruje vezu između ostalih komponenti sistema i na DTP-u se predstavlja imenovanim, orijentisanom linijom.

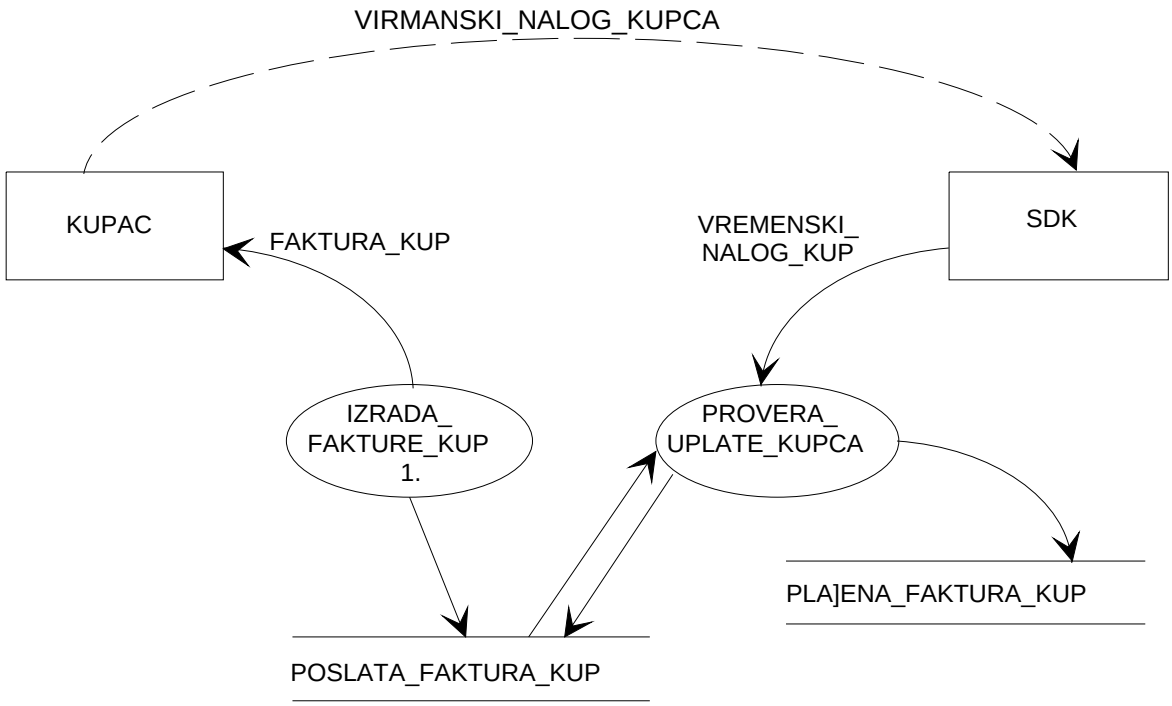
(**II**) Tok podataka mora da ima izvor i ponor. Bilo koja druga komponeneta DTP može da bude izvor ili ponor. Međutim, za jedan tok, bilo izvor, bilo ponor (bilo oba) mora da bude proces. Drugim rečima, tokovima se ne mogu neposredno povezati dva skladišta, dva interfejsa, ili skladište i interfejs. Na slici 2 i slici 3 prikazani su primeri ovih nedozvoljenih situacija, za koje se mogu dati sledeći komentari:

- Nekoreketni delovi DTP-a na slikama 2a i 2c su istovremeno i nelogični, jer u realnom sistemu spoljni objekti nemaju direktan pristup skladištima podataka, već to obavljaju procesi sistema. Slike 2b i 2d pretstavljaju korektne reprezentacije situacija sa slika 2a i 2c, respektivno.

- Direktno povezivanje interfejsa, objekata van sistema, nekim tokom podataka (Slika 3) nije dozvoljena, jer pretstavlja opis odnosa objekata van posmatranog sistema, pa nije od interesa. Ako bi takva veza bila od interesa, odnosno ostvarivala se preko posmatranog sistema, tada bi se morao definisati proces u posmatranom sistemu koji takvu vezu uspostavlja.



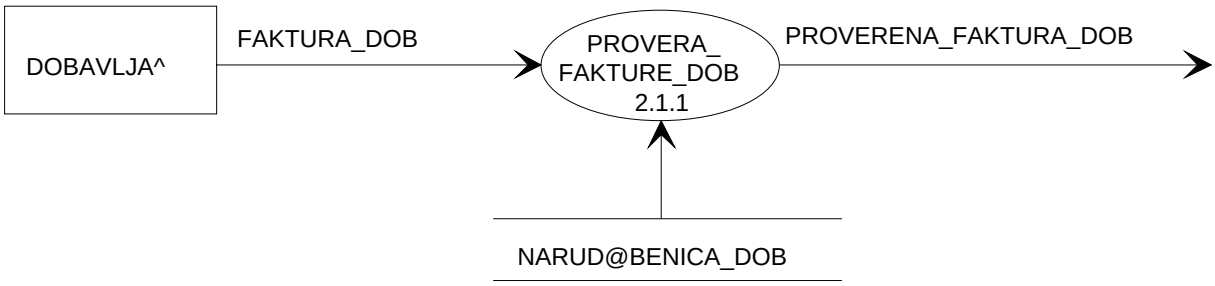
Slika 2. Primer nekorektnih i korektnih DTP



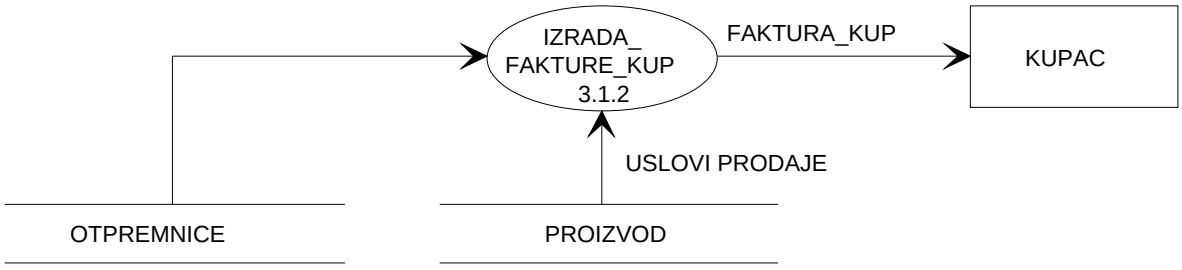
Slike 3. Primer nekorektnog DTP

(**III**) Svaki tok podatka na DTP-u mora imati ime, koje treba da odražava značenje podataka koje on nosi. Da bi se ostvarila čitljivost i razumljivost DTP-a, ova imena treba da budu prirodna, a ne neka specifična, kodirana, preterano skraćena imena. Izuzetak su tokovi koji idu ka, odnosno od skladišta

podataka koji ne moraju biti imenovani. Ako tok između procesa i skladišta nije imenovan, podrazumeva da tok nosi celokupan sadržaj i strukturu podataka tog skladišta. Ukoliko to nije slučaj, tj. ako tok sadrži samo deo strukture podataka, treba ga imenovati. Konvencije o imenovanju prikazane su na slikama 4 i 5. Na Slici 5. pretpostavljeno je da proces IZRADA_FAKTURE_KUP preuzima ceo sadržaj i strukturu podataka OTPREMNICE, dok od svih podataka u skladištu PROIZVOD preuzima samo podatke USLOVI_PRODAJE. Ako je tok između skladišta i procesa imenovan, imenivana struktura mora biti deo strukture skladišta, što treba da bude specifikovano u Rečniku podataka.

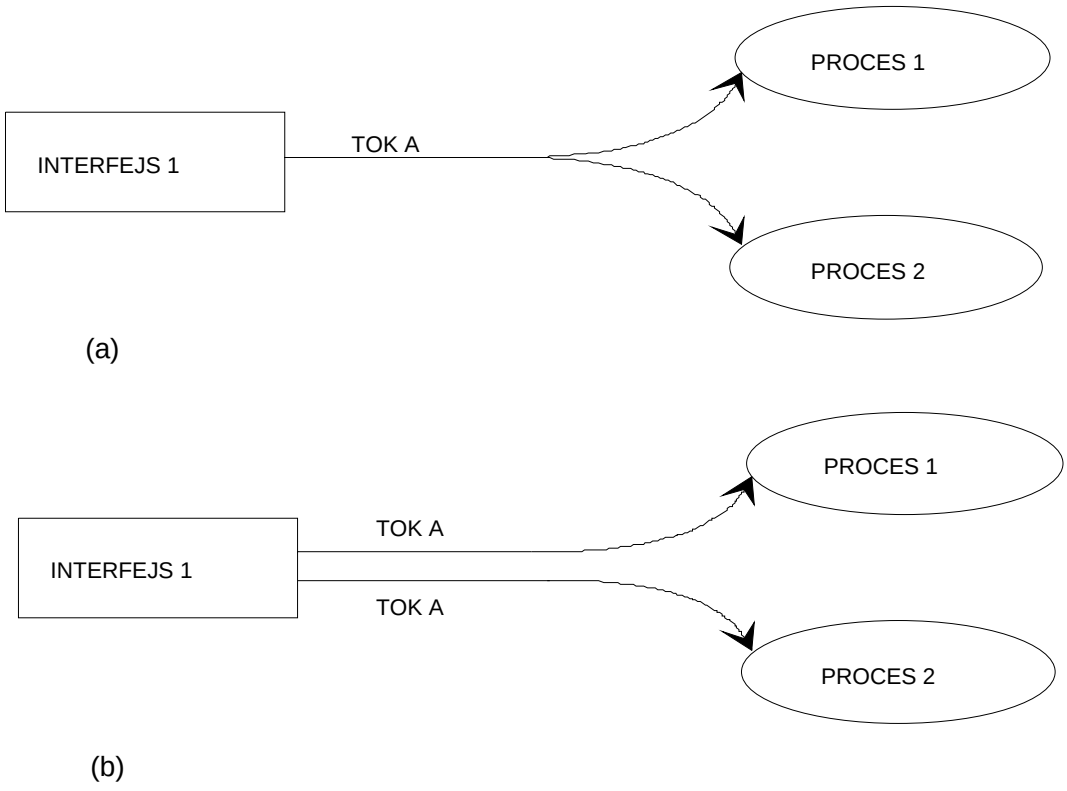


Slika 4. Neimenovani tok proces-skladište



Slika 5. Imenovani tok proces-skladište

(**IV**) Tok podataka se može granati, kako je to pokazano na primeru na Slici 6. Slika 6a eksplicitno prikazuje grananje jednoga toka, dok je na Slici 6b, ista struktura, umesto preko grananja prikazana sa dva istoimena toka koja imaju isti izvor a različite ponore. Drugim rečima, istoimeni tokovi na DTP u suštini pretstavljaju grananje jednog toka, pa moraju imati zajednički izvor, a mogu imati različite ponore.



Slika 6. Grananje tokova

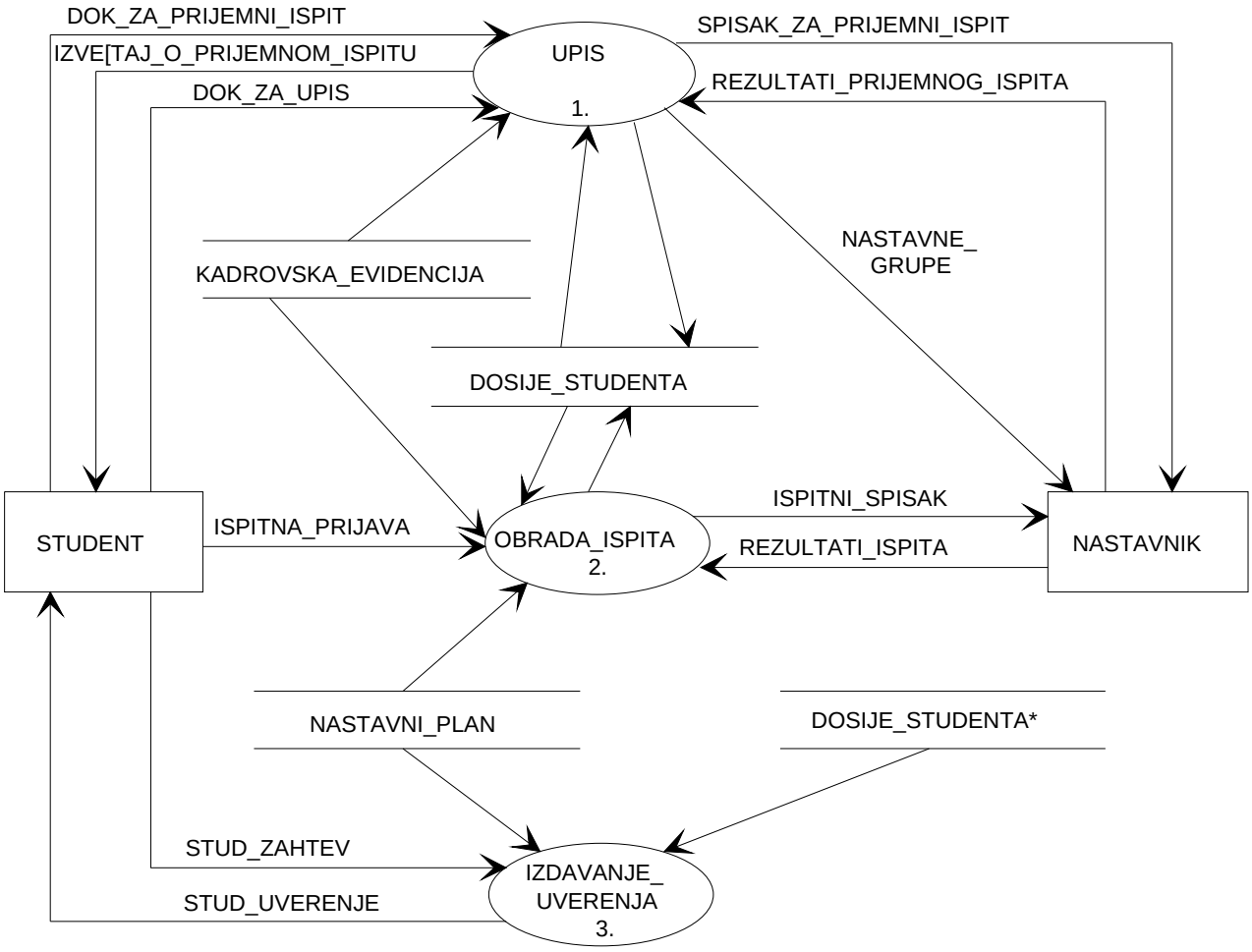
(**V**) Proces obrade podataka je aktivna komponenta sistema koja vrši tranformaciju strukture i sadržaja ulaznog (ili ulaznih) tokova u izlazni (ili izlazne) tok podataka. Svaki proces ima naziv i oznaku. Naziv procesa treba da precizno označava funkciju koju on obavlja. Nepisano pravilo kaže da, ako analitičar nije u stanju da dodeli ime procesu to samo znači da ne razume funkciju koju proces obavlja. Brojna oznaka procesa služi za referenciranje procesa. O načinu dodeljivanja brojne oznake govoriće se u odeljku o dekompoziciji procesa.

(**VI**) Svaki proces mora da ima barem jedan ulazni i barem jedan izlazni tok podataka. Proces bez ulaznog toka generisao bi izlaz niizčega, a proces bez izlaznog toka je nesvrsishodan.

(**VII**) Po pravilu, svako skladište takođe treba da ima barem jedan ulazni i barem jedan izlazni tok. Međutim, dozvoljava se da skladište nema ulazni tok, podrazumevajući da se formira i ažurira u nekom drugom sistemu (mada bi ga tada, možda, logičnije bilo prikazati kao tok od nekog interfejsa), odnosno da nema izlazni tok, podrazumevajući da posmatrani sistem formira i ažurira skladište koje se koristi u nekom drugom sistemu.

(**VIII**) Svaki interfejs mora da ima barem jedan, bilo ulazni, bilo izlazni tok podataka, inače bi bio izolovan od ostalog dela sistema.

(**IX**) Da bi se DTP-ovi lakše crtali, odnosno izbeglo nepotrebno presecanje linija, dozvoljava se da se jedno skladište ili interfejs, na jednoj slici višestruko ponove. Primer je dat na Slici 7, gde je ponovljeno skladište DOSIJE_STUDENTA.

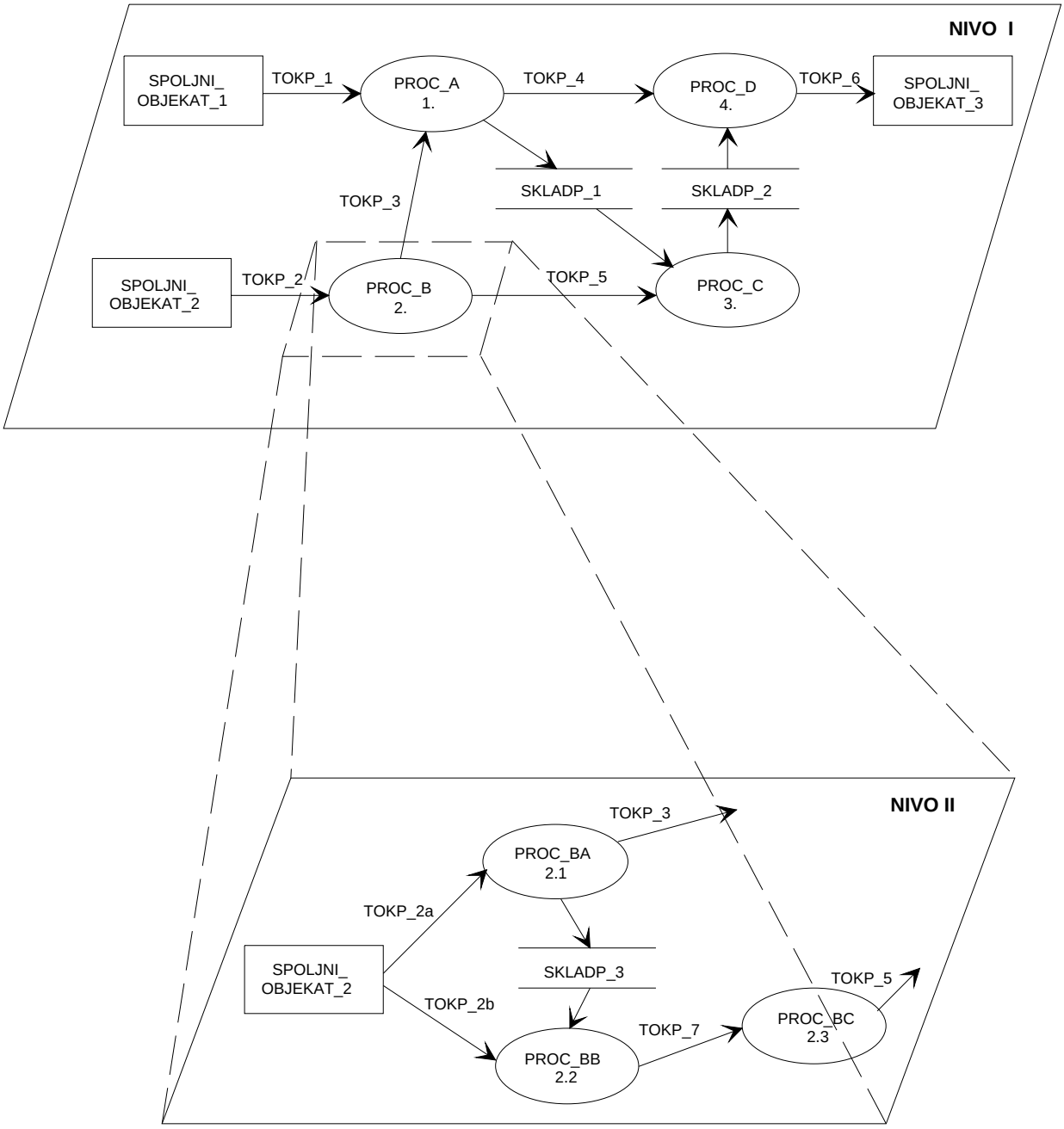


Slika 7. Primer dijagrama toka podataka

Ovo su osnovna, jednostavna pravila za konstrukciju DTP. Jedan korektno konstruisan DTP dat je na Slici 7. Znatno kompleksnija pravila proizilaze iz potrebe hijerarhijske dekompozicije dijagrama.

2.2. Hijerarhijska dekompozicija dijagrama toka podataka

Jedan informacioni sistem može biti veoma složen i može sadržati veliki broj procesa, tokova podataka, skladišta podataka i spoljnih objekata. Istovremeno jasna i detaljna specifikacija sistema zahteva da se i na predstavljanje sistema pomoću DTP-a primeni metoda apstrakcije. To se postiže primenom hijerarhijske dekompozicije DTP. Hijerarhijska dekompozicija DTP se izvodi na taj način što se jedan proces višeg nivoa apstrakcije dekomponuje i prikazuje pomoću novog celokupnog DTP-a na nižem nivou apstrakcije. Tehnika hijerarhijske dekompozicije prikazana je na Slici 8, gde je proces PROC_B, označen sa brojem 2, dekomponovan u novi dijagram toka na nižem nivou.



Slika 8. Dekompozicija DTP

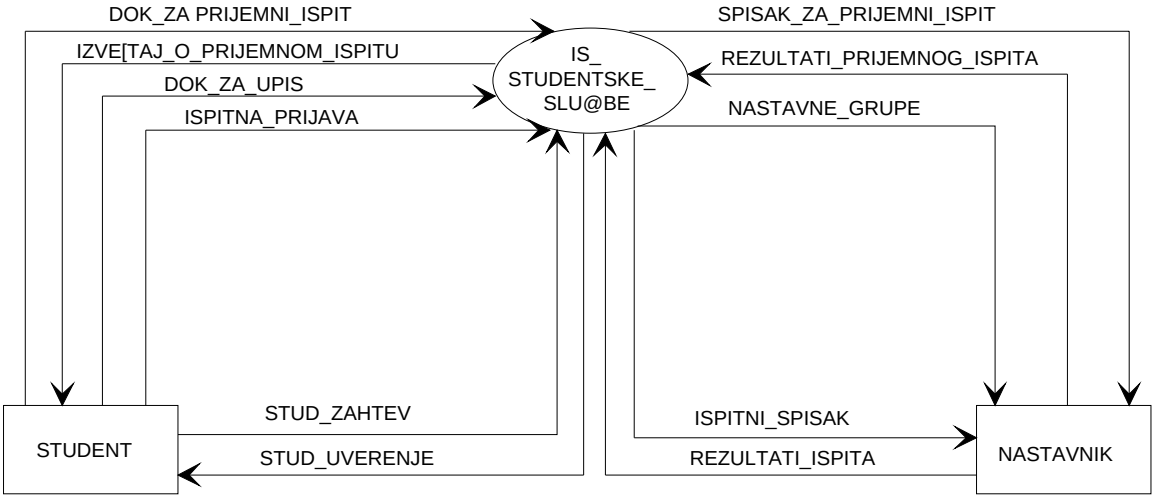
Pri dekompoziciji dijagrama toka podataka moraju se poštovati sledeća pravila i konvencije:

(**I**) Dijagram najvišeg nivoa, koji po pravilu sadrži samo jedan proces koji pretstavlja ceo IS, interfejse sa kojima IS komunicira i odgovarajuće tokove podataka naziva se dijagram konteksta. Primer dijagrama konteksta za Informacioni sistema studentske službe na jednom fakultetu dat je na slici 9.

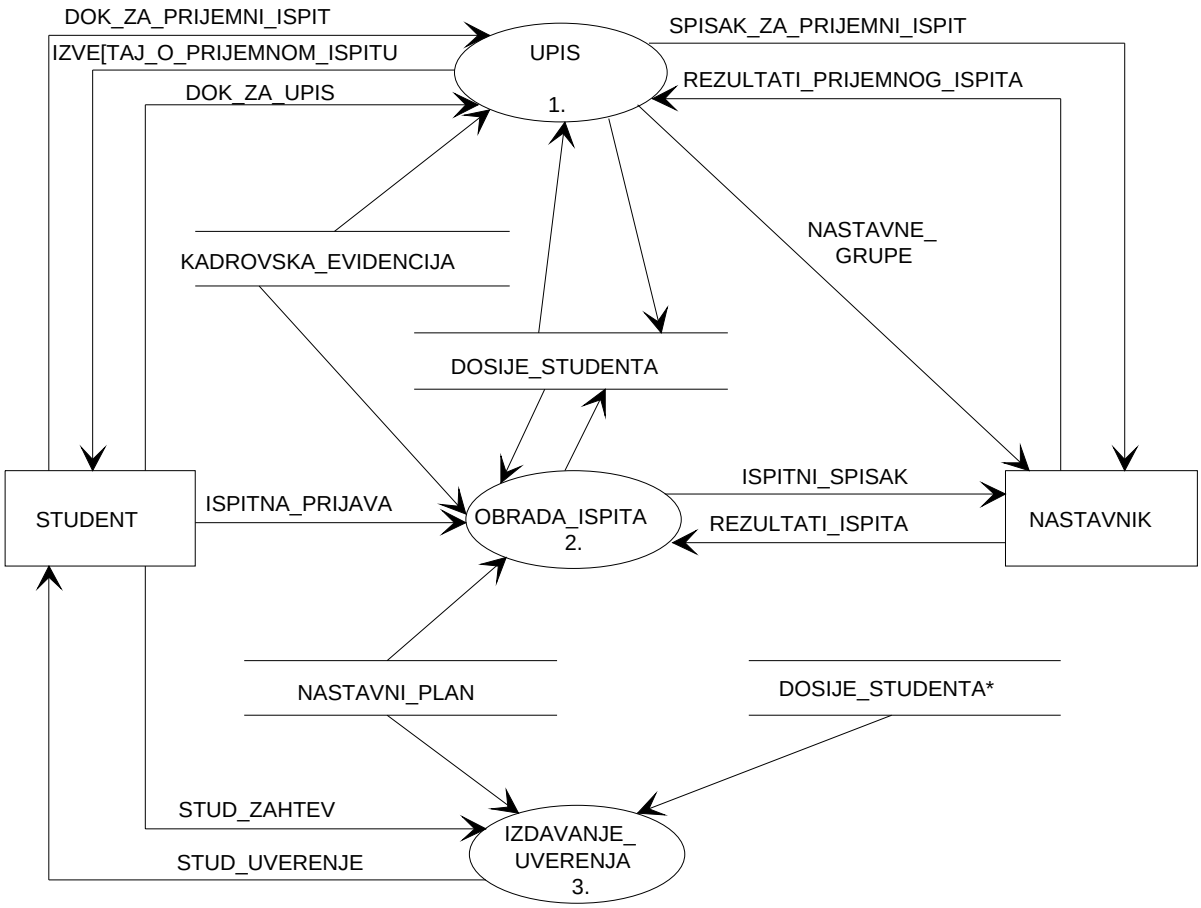
(**II**) Dijagram prvog nivoa pretstavlja dekompoziciju dijagrama konteksta. Procesi na njemu označavaju se brojevima 1, 2, 3, ..., kako je to prikazano na Slici 10. Dijagrami nižih nivoa, kao celina, su označeni sa oznakom procesa čije detalje pretstavljaju, a procesi na njima povlače sa sobom brojnu oznaku nadređenog procesa, odnosno posmatranog dijagrama (Slike 11 i 12).

(**III**) Uobičajeno je da se u dokumentaciji za specifikaciju IS pomoću SSA, celokupan skup, ili neki podskup hijerarhiski dekomponovanih dijagrama, pretstavi dijagramom dekompozicije.

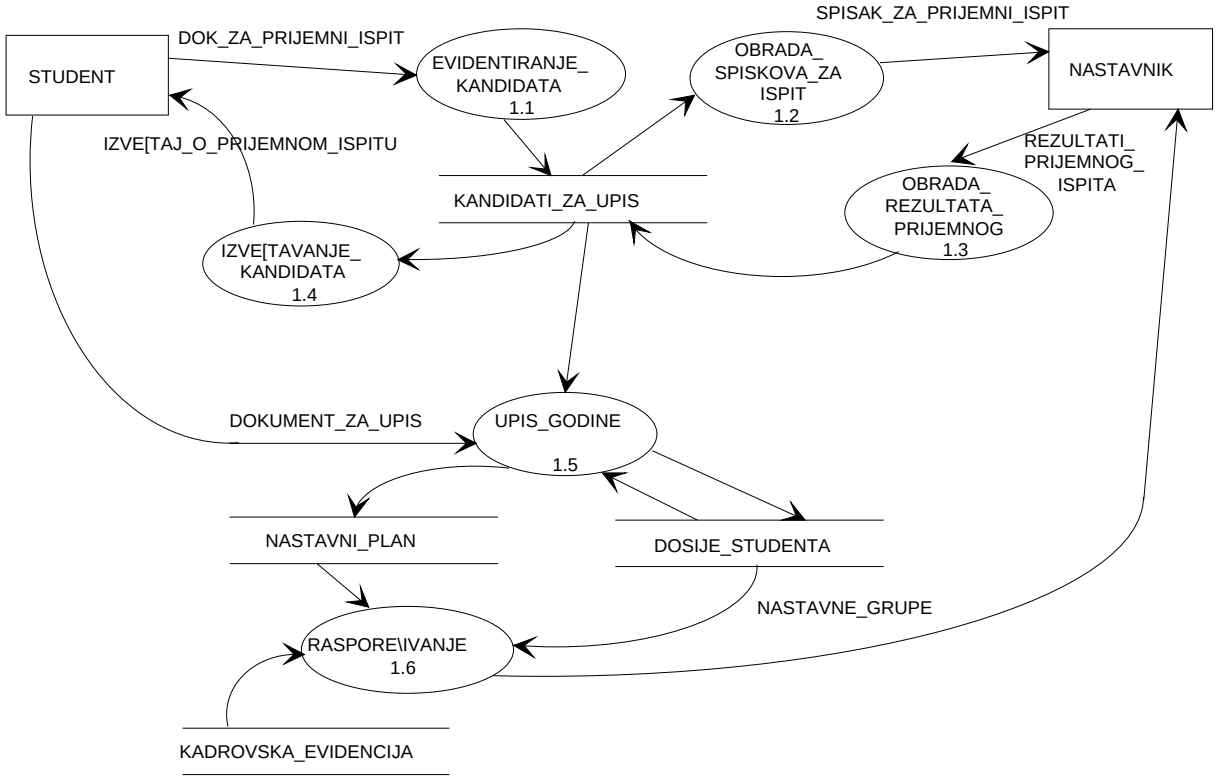
Opšti primer dijagrama dekompozicije prikazan je na Slici 14, a za IS studentske službe na slici 15.



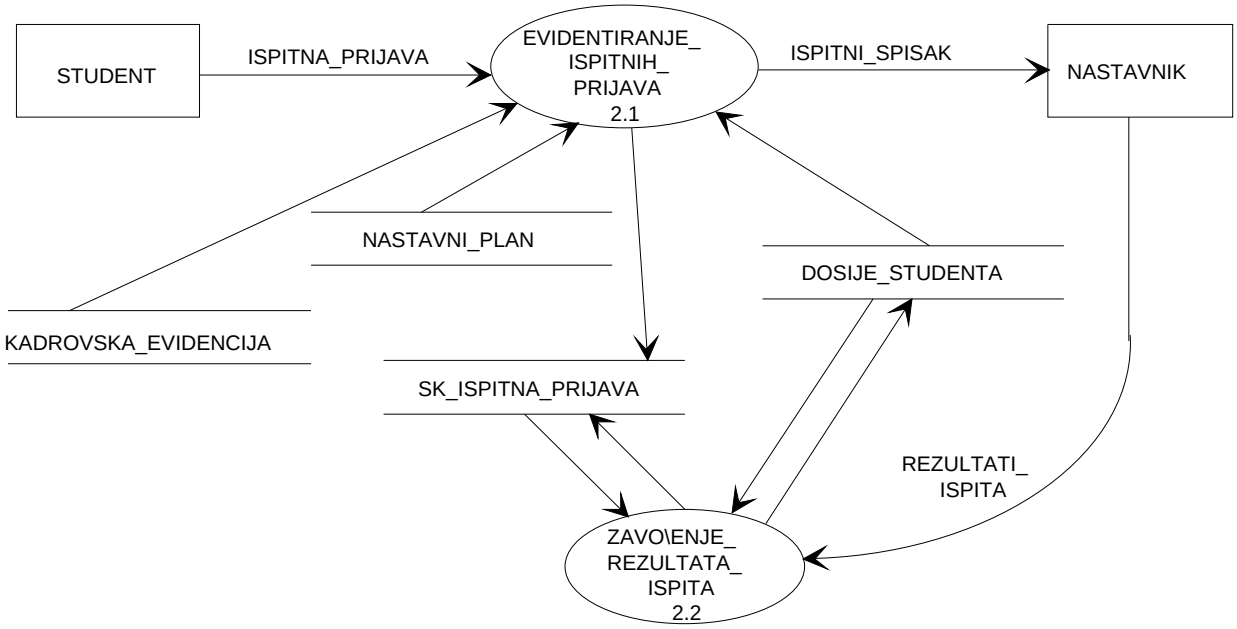
Slika 9. Dijagram konteksta IS studentske službe



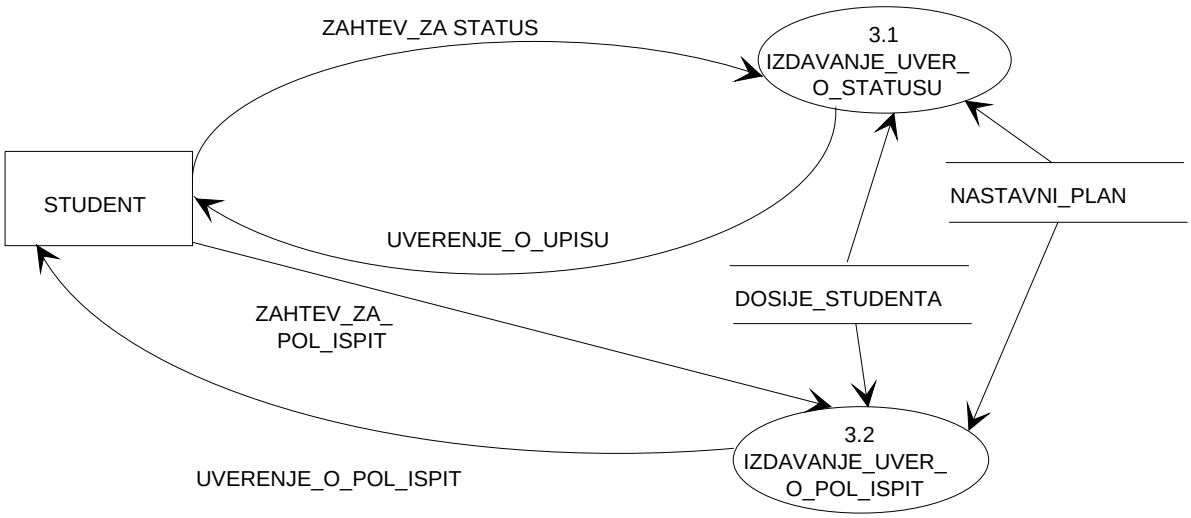
Slika 10. Dijagram prvog nivoa IS studentske službe



Slika 11. Dekompozicija procesa Upis (1)



Slika 12. Dekompozicija procesa Obrada ispita (2)



Slika 13. Dekomozicija procesa Izdavanje uverenja (3)

(**IV**) Procesi koji se dalje ne dekomponuju (na primer procesi 1.1 do 1.5, 2.1 i 2.2, kao i procesi 3.1 i 3.2 za IS Studentske službe) se nazivaju primitivni procesi i za njih se daje specifikacija logike njihovog odvijanja. Opis logike primitivnih procesa naziva se mini-specifikacija sistema. O sredstvima za minispecifikaciju govori se u posebnom poglavlju.

(**V**) Pored procesa, mogu se dekomponovati i tokovi i skladišta. Dekompozicija tokova i skladišta se ne prikazuje na DTP-u, već u Rečniku podataka, pomoću sintakse za opis strukture podatka, o čemu će kasnije biti više reči. Mogu se dekomponovati samo oni tokovi i skladišta koji u sebi sadrže nezavisne komponente, komponente čija unija čini tok ili skladište koje se dekomponuje. Na primer:

- Zahetvi studenata i uverenja, prikazana kao jedinstveni tokovi na dijagramu konteksta i dijagramu prvog nivoa, razlažu se na odgovarajuće komponente na dijagramu 3, s tim što se u rečnik podataka unose stavke

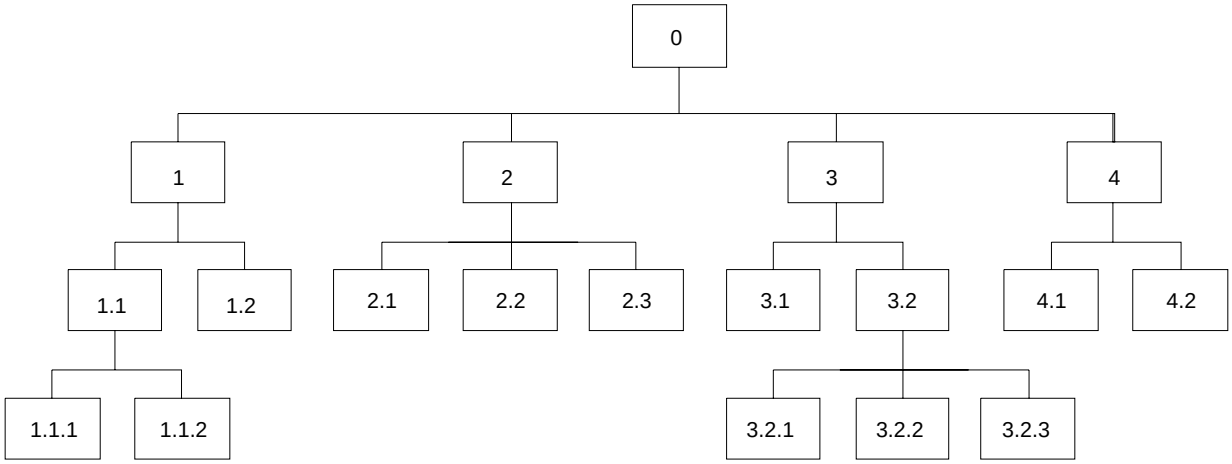
STUD_ZAHTEV: [ZAHTEV_ZA_STATUS, ZAHTEV_ZA_POL_ISPITE]

STUD-UVERENJE: [UVERENJE_O_UPISU, UVERENJE_O_POL_ISPIT]

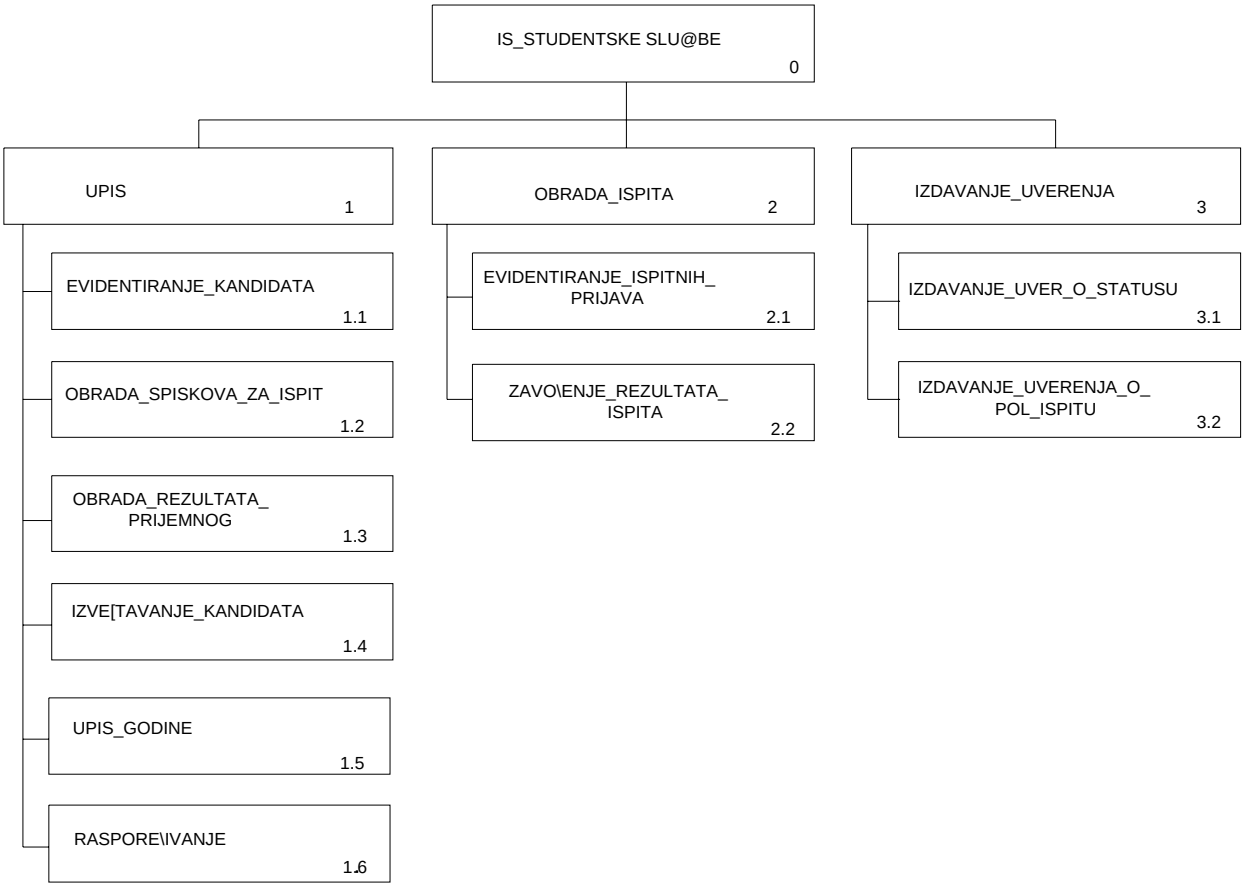
(Srednje zagrade označavaju "eksluzivnu uniju". STUD_ZAHTEV je bilo ZAHTEV_ZA_STATUS ili ZAHTEV_ZA_POL_ISPITE. STUD_UVERENJE je bilo UVERENJE_OUPISU ili UVERENJE_O_POL_ISPIT. Detaljna specifikacija sintakse rečnika podataka se daje u sledećem poglavlju.)

- Međutim tokovi, Dokumenta za upis i Dokumenta za prijemni ispit, koji su prikazani na dijagramu konteksta i dijagramu prvog nivoa kao celine, ne mogu se, kako to na prvi pogled izgleda, dalje dekomponovati na odgovarajućim dijaramima tokova nižih nivoa, jer pretstavljaju nedeljivu celinu u smislu da se ni jedan od njih ne može samostalno pojaviti ni obrađivati.

(**VI**) Najznačajnije pravilo koje se mora poštovati pri dekompoziciji procesa je pravilo balansa tokova: Ulazni i izlazni tokovi na celokunom DTP-u koji je dobijen dekompozicijom nekog procesa P moraju biti ekvivalentni sa ulaznim i izlaznim tokovima toga procesa P na dijagramu višeg nivoa. Pri tome se uzima u obzir dekompozicija tokova pretstavljena u rečniku podataka.



Slika 14. Dijagram dekompozicije



Slika 15. Dijagram dekompozicije za IS studentske službe

Pravilo balansa tokova drugim rečima iskazuje se na sledeći način. Svi tokovi koji ulaze, odnosno izlaze iz jednog procesa, moraju se pojaviti kao ulazni, odnosno izlazni tokovi na dijagramu u koji je posmatrani proces dekomponovan. Na tom dijagramu ne može se pojaviti nijedan drugi ulazni i izlazni tok. Na primer, svi tokovi podataka sa dijagrama konteksta na Slici 9, pojavljuju se kao ulazni i izlazni

tokovi na dijagramu prvog nivoa dekompozicije, na Slici 10. Proces OBRADA_ISPITA na dijagramu na Slici 10 ima ulazne tokove ISPITNA_PRIJAVA i REZULTATI_ISPITA i izlazni tok ISPITNI_SPISAK. Ovi tokovi su ulazni i izlazni tokovi i na dijagramu na Slici 12, koji pretstavlja dekompoziciju procesa OBRADA_ISPITA.

Na dijagramu na Slici 10 ulazni tok u proces IZDAVANJE_UVERENJA je STUD_ZAHTEV, a izlazni je STUD_UVERENJE. Ovi tokovi se ne pojavljuju na dijagramu na Slici 13 koji pretstavlja dekompoziciju ovog procesa, već se pojavljuju neki novi tokovi ZAHTEV_ZA_STATUS, ZAHTEV_ZA_POL_ISPITE UVERENJE_O_UPISU i UVERENJE_O_POL_ISPIT, pa izgleda da je pravilo balansa tokova narušeno. Međutim, u Rečniku podataka postoje stavke

STUD_ZAHTEV: [ZAHTEV_ZA_STATUS, ZAHTEV_ZA_POL_ISPITE]
STUD-UVERENJE: [UVERENJE_O_UPISU, UVERENJE_O_POL_ISPIT]

preko kojih se balanas tokova uspostavlja.

(**VII**) 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. Međutim, ako se pojavi na jednom nivou, uz jedan proces, mora se nadalje pojavljivati na svim nižim nivoima dekompozicije toga procesa. Uobičajeno je da se skladišta prikazuju po prvi put na onom DTP-u na kome predstavljaju interfejs između dva ili više procesa. I skladišta se mogu dekomponovati preko Rečnika podataka. Međutim, to nije neophodno, jer se nazivima tokova od i ka skladištu mogu iskazati komponente skladišta koje dati proces ažurira ili koristi.

(**VIII**) O metodološkim aspektima dekompozicije, o tome kako se dekomponuje, na koliko nivoa, na kom nivou se prestaje sa dekompozicijom, biće kasnije mnogo više reči. Za sada se navodi samo opšta preporuka, da jedan DTP, po pravilu, treba da sadrži 2 - 7 procesa. Veći broj procesa od ovoga značio bi da je "preskočen" jedan nivo apstrakcije.

2.3. Primer dijagrama toka podataka

Kao što je već diskutovano, na slikama 9 - 13 pretstavljen je primer dijagrama toka podataka za obradu podataka u studentskoj službi jednog fakulteta.

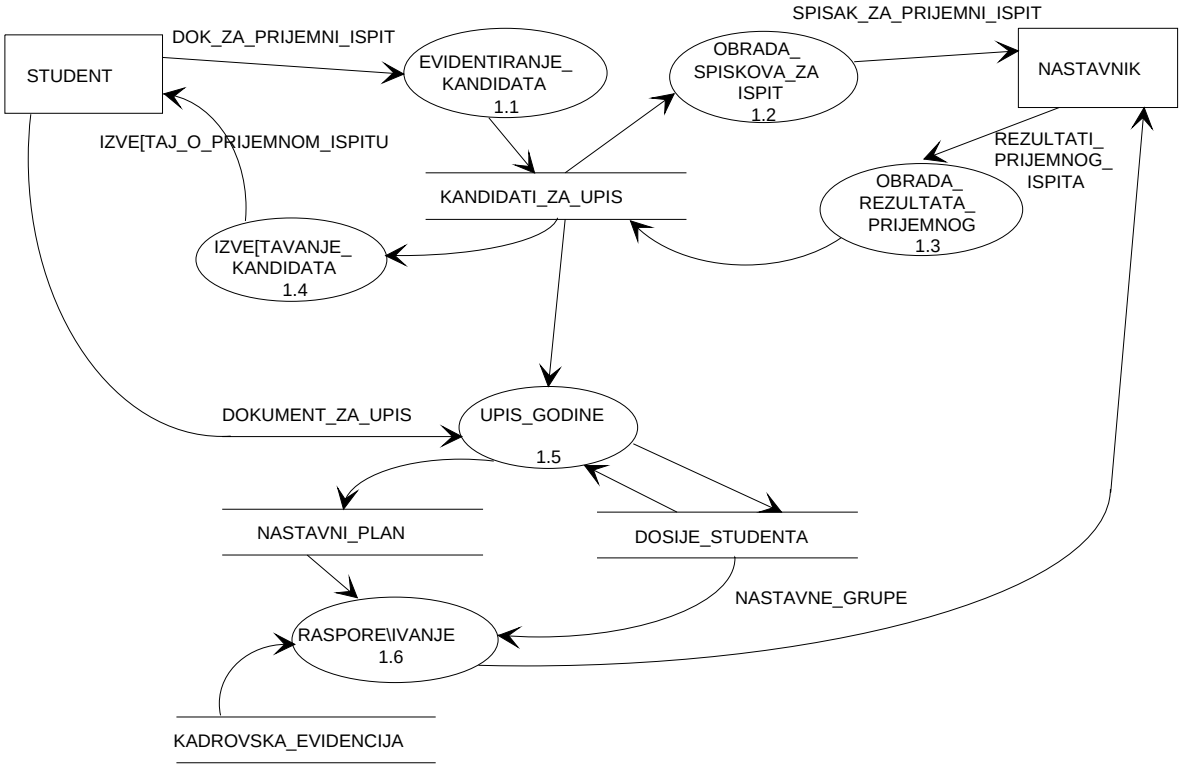
Na dijagramu konteksta definisana su dva interfejsa, objekta van sistema sa kojima IS studentske službe komunicira, STUDENT i NASTAVNIK. Zatim je dat dijagram prvog nivoa dekompozicije. Svaki proces sa dijagrama prvog nivoa, UPIS, OBRADA_ISPITA, IZDAVANJE_UVERANJA, pretstavljen je posebnim dijagramom toka podataka.

Primitivni procesu u ovoj hijerarhijskoj strukturi su:

EVIDENTIRANJE-KANDIDATA, OBRADA_REZULTATA-PRIJEMNOG, UPIS_GODINE, RASPOREĐIVANJE (studenata u grupe za predavanja i vežbe), IZVEŠT_KANDIDATA, zatim EVIDENTIRANJE_ISPITNIH_PRIJAVA, ZAVOĐENJE_REZULTATA_ISPITA i na kraju, IZDAVANJE_UVERENJA_O_POL_ISPIT i IZDAVANJE_UVERENJA_O_STATUSU.

Za ove primitivne procese treba dati minispecifikaciju.

Mada je značenje prikazanih DTP očigledno, navešćemo kao primer "čitanje" jednog dela DTP koji pretstavlja dekompoziciju procesa UPIS (Slika 16):



Slika 16. Dekompozicija procesa Upis

- Proces EVIDENTIRANJE_KANDIDATA, na osnovu ulaznih dokumeneta DIPLOMA_SREDNJE_ŠKOLE, SVEDOČANSTVO i NAGRADE formira skladište podataka KANDIDATI_ZA_UPIS i generiše SPISAK_ZA_PRIJEMNI_ISPIT koji se šalje nastavniku koji treba da obavi prijemnni ispit. Po Obavljanju ispita nastavnik vraća REZULTATI_PRIJEMNOG_ISPITA koji se obrađuju u procesu OBRADA_REZULTATA_PRIJEMNOG i ažurira se skladište KANDIDATI_ZA_UPIS (registruju se rezultati kandidata). Koristeći se podacima iz ovog skladišta proces IZVEŠTAVANJE_KANDIDATA generiše IZVEŠTAJ_OPRIJEMNOM_ISPITU koji se šalje kandidatima (studentima).

- Proces UPIS_GODINE obuhvata kako upis prve, tako i upis svih drugih godina (semestara). Student podnosi standardna dokumenta za upis. Za studente koji upisuju prvu godinu proveravaju se rezultati prijemnog ispita u skladištu KANDIDATI_ZA_UPIS, a za sve ostale, ispunjenost uslova u skladištu DOSIJE_STUDENTA. U oba slučaja proces UPIS_GODINE ažurira skladište DOSIJE_STUDENTA.

- Na osnovu NASTAVNOG_PLANA u kome se daju predmeti u semestru i broj časova nastave i vežbi, kao i opštih pravila o veličini grupa za pojedine predmete i DOSIJEA_STUDENATA, kao i KADROVSKE_EVIDENCIJE u kojoj su dati podaci o nastavnicima i asistentima po predmetima, proces RASPOREĐIVANJE, raspoređuje studente po grupama i šalje odgovarajuće spiskove (NASTAVNE_GRUPE) nastavniku.

3. REČNIK PODATAKA STRUKTURNE SISTEMSKE ANALIZE

Rečnik podataka, kao što je ranije rečeno, daje opis strukture i sadržaja svih tokova i skladišta podataka. Bez obzira šta tok ili skladište podataka pretstavljaju, papirni dokumenat, niz karaktera kao ulaz sa terminala, "paket" informacija dobijen telekomunikacionom linijom, kartoteku ili datoteku, kao logička struktura podataka oni pretstavljaju neku kompoziciju polja. Da bi precizno definisali logičku strukturu sladišta i tokova i definisali sintaksu rečnika neophodno je da uvedemo definicije svi koncepata rečnika.

3.1. Definicije i primeri koncepata rečnika

3.1.1. Polja i domeni

(**I**) Polje je elementarna (atomska) struktura koja se dalje ne dekomponuje i koja ima svoju vrednost. Na primer, u indeksu, polja su BROJ_INDEKSA, IME_I_PREZIME, OCENA, STATUS i slično.

(**II**) Polja svoje vrednosti uzimaju iz skupova vrednosti koji se nazivaju domenima. Domeni mogu biti:

- "predefinisani", odnosno standardni programsko-jezički domeni, kao što su INTEGER, CHARACTER, REAL, LOGICAL i DATE.

- "semantički", kada se definišu posebno, preko svoga imena, predefinisanog domene i, eventualno, ograničenja na mogući skup vrednosti predefinisanog domena. Na primer, semantički domen SEMESTRI se definiše kao

  SEMESTRI DEFINED_AS INTEGER (2) BETWEEN 1,10

Ključna reč DEFINED_AS vezuje imenovani i predefinisani domen, a iskaz BETWEEN 1, 10 ograničava vrednosti semestara na ovaj opseg prirodnih brojeva. Opšta sintaksa za iskazivanje ograničenja na predefinisani skup vrednosti za semantički domen je:

definicija_semantičkog__domena ::
naziv_domena 'DEFINED AS' predefinisani_domen [ograničenje]

(Kao što je uobičajeno, uglasta zagrada znači opcioni deo iskaza.)

O definiciji predefinisanih domena i drugih ograničenja, osim ovog navedenog u primeru, govoriće se kasnije.

(**III**) Činjenica da polje uzima vrednost iz nekog domena označava se na sledeći način:

naziv polja : domen [ograničenje]

Na primer:

BI: CARACTER(7)
SEMESTAR: SEMESTRI
OCENA: INT(2) IN (5,6,7,8,9,10)
SEMESTVŠ: SEMESTRI IN (1,2,3,4) *Semestar više škole*

(**IV**) Osnovni razlog za uvođenje semantičkih domena je jasno iskazivanje semantičke sličnosti dva polja. Naime dva polja su semantički slična samo ako su definisana nad istim domenom.

Drugim rečima, semantički domeni uspostavljaju razliku između pojedinih istovrsnih predefinisanih domena koji nemaju semantičku sličnost. Na primer, ako bi se i polje SEMESTAR definisalo direktno nad predefinisanim domenom INTEGER, kao

SEMESTAR: INTEGER (2) BETWEEN 1,10

tada bi polja OCENA i SEMESTAR bila semantiki slična, mogla bi se povezivati operatorima definisanim nad predefinisanim domenom INTEGER, na primer, moglo bi se pisati

IF SEMESTAR > OCENA THEN ....

što očigledno nema smisla. Međutim, ako je polje SEMESTAR definisano nad semantičkim domenom SEMESTRI, a polje OCENA nad predefinisanim domenom INTEGER, prethodni izraz bi bio i sintaksno nekorektan. Različita polja se mogu povezati nekim operatorom samo ako su definisana nad istim domenom i ako je operator definisan u tom domenu.

Očigledno je da bi u opštu strukturu i tokova trebalo težiti što je moguće većem korišćenju semantičkih domena. Međutim, treba imati u vidu da retko koji računarski jezik podržava koncept semantičkog domena.

(**V**) Predefinisani domeni su standardni progrmsko-jezički domeni, koji se ovde definišu na sledeći način:

INTEGER(dužina)

CHARACTER(dužina)

REAL(dužina celokupnog broja, dužina iza zareza)

LOGICAL

DATE

(**VI**) Pored ograničenja na vrednosti polja, odnosno vrednosti domena koja su data u primerima definišu se i druga. Ograničenja mogu biti prosta i složena. Lista dozvoljenih prostih ograničenja je:

(a) Θ *konstanta*, gde je Θ bilo koji operator poređenja koji se na datom domenu može definisati (na primer, <, >, =, <=, >= za brojne domene), a *konstanta* je neka definisana vrednost iz datog domena. Na primer:

STAROST: INT(2) < 65

(b) BETWEEN konstanta , konstanta, gde su konstante vrednosti iz datog domena. Na primer:

SEMESTAR: INTEGER (2) BETWEEN 1,10

(c) IN (*lista vrednosti*), gde se lista formira od konstanti iz odgovarajućeg domena. Na primer:

OCENA INT(2) IN (5,6,7,8,9,10)

(d) NOT NULL, kada dato polje ne može da dobije "nulla vrednost", odnosno mora uvek da ima vrednost. Na primer:

BROJ_INDEKSA: CHARACTER (7) NOT NULL

Složena ograničenja se formiraju od prostih ili drugih složenih ograničenja vezujići ih logičkim operatorima AND, OR i NOT. Na primer:

STAROST: INT(2) < 65 AND NOT NULL

Bilo prosto, bilo složeno ograničenje se može imenovati, odnosno posebno definisti kao Bullova (logička funkcija) i samo ime navesti kao ograničenje. Na primer, pretpostavimo da polje ŠIFRA_PR

(šifra_proizvoda) ima kontrolu "po modulu 11" Tada bi se mogla definisati funkcija MODUO_11 koja dobija vrednost TRUE ako je pomenuta kontrola zadovoljena, a FALSE ako nije. Tada se ograničenje defiše kao

ŠIFRA_PR INT(13) MODUO_11

Bez obzira što je polje atomska, nedeljiva komponenta, ponekad je, za potrebe definisanja ograničenja, potrebno ući i u njegovu strukturu. Zbog toga je uvedena funkcija SUBSTRING koja ima za cilj da izvuče deo polja i prikaže neke karakteristike toga dela.

Funkcija SUBSTRING se definiše kao

SUBSTRING (položaj prvog člana, položaj poslednjeg člana)

Na primer:

MLBR CHARACTER(13) SUBSTRING(1,2) < 31 AND SUBSTRING(3,4) < 12
AND SUBSTRING(5,7) BETWEEN 000, 999 ....

(Prva dva karaktera predstavljaju datum u mesecu, druga dva mesec u godini itd.)

(**VII**) Ograničenja koja se definišu su samo ograničenja na domene, odnosno polja. Čak i u složenijim ograničenjima i logičkim funkcijama preko kojih se ponekad iskazuju, jedini argumenat može biti domen, odnosno polje (ili neki njihov deo dobijen funkcijom SUSTRING) na koga se ograničenje odnosi. Složenija, tzv "vrednosna ograničenja", koja povezuju vrednosti više polja (na primer da prosečna ocena na svedočanstvu mora da bude jednaka sumi ocena po predmetima podeljenoj sa brojem predmeta), iskazuju se u SSA proceduralno, u mini specifikacijama, preko sredstava za opis logike.

3.1.2. Strukture

Kao što je ranije rečeno, struktura tokova podataka i skladišta pretstavlja neku kompoziciju polja, odnosno konstrukciju čije su komponente polja. Očigledno je da se kao komponenta jedne strukture može, pored polja, može pojaviti i druga definisana struktura. Konstrukcija kojom se od komponenata gradi struktura može biti:

(a) Agregacija komponenti, koja se pretstavlja kao lista komponenti koje je čine u "špicastim" zagradama - <a,b,c.>, na primer. Agregacija pretstavlja složenu strukturu n komponenti. Vrednost agregacije je n-toka u kojoj svaki elemenat ima vrednost odgovarajuće komponente. Na primer,

```
ISPITNA_PRIJAVA:   <      BROJ_INDEKSA,
                               IME_STUDENTA,
                               NAZIV_PREDMETA,
                               DATUM_POLAGANJA,
                               OCENA,
                               IME_NASTAVNIKA
                          >
```

(b) Eksluzivna specijalizacija (unija) komponenti, koja se pretstavlja kao lista komponenti u uglastim zagradama - [a,b,c], na primer i koja označava da se u strukturi pojavljuje eksluzivno jedna od navedenih komponenti, ili a ili b ili c. Ako se u uglastoj zagradi pojavi samo jedna komponenta, kao [a], to znači da se u strukturi ova komponenta javlja ili ne javlja. Primer za eksluzivnu specijalizaciju komponeneti je:

```
PROIZVOD:    <    ŠIFRA_PR,
                  NAZIV_PR,
                  [ STOPA_AMORT, KOLIČ_NARUČ ]
             >
```

Podaci o proizvodu pretstavljaju agregaciju polja ŠIFRA_PR i NAZIV_PR i eksluzivne specijalizacije [STOPA_AMORT, KOLIČ_NARUČ] koja kaže da se u strukturi javlja bilo polje STOPA_AMORT (ako je proizvod osnovno sredstvo), bilo KOLIČ_NARUČ (ako je proizvod materijal za proizvodnju)

(c) Neeksluzivna specijalizacija (unija) komponenti, koja se pretstavlja kao lista komponenti u kosim zagradama - /a,b,c/, na primer i koja označava da se u odgovarajućoj srtukturi pojavljuje bilo samo jedna, komponenta, bilo dve, bilo sve. Primer za neeksluzivnu specijalizaciju je:

```
STUD_ZAHTEV:    / ZAHTEV_ZA_UVER_STATUS,
                  ZAHTEV_ZA_UVER_POL_ISPIT /
```

Student može da podnese bilo zahtev za uvrenje o statusu (redovan ili vanredan) bilo zahteva za uverenje o položenim ispitima, bilo oba.

(d) Skup komponenti (preciznije skup više vrednosti jedne komponente), koji se pretstavlja u vitičastim zagradama, na primer {a}, i koja kaže da se u odgovarajućoj strukturi komponenta može da pojavi više puta. Primer za skup komponenti je:

```
UVERENJE_O_POL_ISPIT:    <    BROJ_INDEKSA,
                                   IME_STUDENTA,
                                   {< NAZIV_PREDMETA, OCENA>},
                                   PROSEČNA_OCENA
                              >
```

UVERENJE_O_POL_ISPIT je agregacija polja BROJ_INDEKSA i IME_STUDENTA, zatim skupa agregacije polja NAZIV_PREDMETA, OCENA (naziv predmeta i ocena se na uverenju više puta ponavljaju) i polja PROSEČNA_OCENA.

3.2. Sintaksa za specifikaciju Rečnika podataka

Na osnovu prethodno definisanik koncepata i primera ovde se, u BCNF notaciji, daje potpuna sintaksa za specifikaciju Rečnika podataka. Tekst između zvezdica je objašnjenje sintakse.

* Ključne reči sintakse su bilo date velikim slovima, bilo između znakova navoda " ".*

```
Rečnik podataka :: STRUCTURES tačka-zarez-lista-struktura
FIELDS tačka-zarez-lista-polja
DOMAINS tačka-zarez-lista-domena
CONSTRAINT_FUNCTIONS tačka-zarez-lista-logič-funkcija .
```

* Rečnik podataka sardrži četiri osnovna segmenta: (i) za opis struktura skladišta i tokova (ii) za opis polja, (iii) za opis semantičkih domena i (iv) za definiciju logičkih funkcija preko kojih se iskazuju složenija ograničenja. Kao što će se docnije videti, po principu "ortogonalnosti jezika" i polja i semantički domeni mogu se definisati u okviru dela za opis struktura tokova i skladišta)*

tačka-zarez-lista-struktura :: struktura
| struktura ";" tačka-zarez-lista-struktura

* tačka-zarez lista struktura sastoji se od struktura razmaknutih sa ;*

struktura :: naziv_strukture ":" opis_strukture
opis_strukture :: "<" zarez-lista-komponenti ">"* agregacija *
| "[" zarez-lista-komponenti "]* eksl. specijal.*
| "/" zarez-lista-komponenti "/" * neeksl. specijal.*
| "{" zarez-lista-komponenti"}"* skup *

zarez-lista- komponenti:: komponenta
| komponenta "," zarez-lista-komponenti

komponenta :: naziv_polja * naziv polja koje je opisano u Rečniku *
| naziv_strukture * Naziv strukt. koja je opisana u Rečn. *
| opis_strukture
| polje* potpuna definicija polja sa nazivom i domenom *
| struktura * potpuna definicija strukture sa nazivom i opisom *

* Poslednje dve mogućnosti u opisu komponente mogu da posluže da se u okviru definicije jedne strukture, definiše i druga struktura i neko polje koji neće biti posebno definisani u Rečniku *

tačka-zarez-lista-polja :: polje
| polje ";" tačka-zarez-lista-polja

polje :: naziv_polja ":" opis_polja
| naziv_polja ":" naziv_domena
| naziv_polja ":" naziv_domena DEFINED_AS opis_polja

* Druga mogućnost se koristi kada je polje definisano nad semantičkim domenom, a treća kada se definicija semantičkog domena daje uz definiciju atributa, a ne posebno *

opis_polja :: tip_polja
| tip_polja ograničenje

tip_polja :: INTEGER "(" dužina ")"
| CHARACTER "(" dužina ")"
| REAL "("ukupna_dužina "," dužina_posle_zapete ")"
| LOGICAL
| DATE

* Polje se može opisati samo sa tipom ili mu se može dodati i ograničenje *

ograničenje :: prosto_ograničenje | složeno_ograničenje

prosto_ograničenje :: □ vrednost_iz_domena
| BETWEEN vrednost_iz_domena "," vrednost_iz_domena
| IN "("skup_vrednosti")"
| NOT NULL

□ :: < | | > | | = |

složeno_ograničenje :: ograničenje AND ograničenje

| ograničenje OR ograničenje
| NOT ograničenje
| IF ograničenje THEN ograničenje
| naziv_logičke_funkcije

logička_funkcija ::naziv_logičke_funkcije"=" logički_izraz

* Pored ranije opisanih prostih i složenih ograničenja, ograničenje se može dati i preko neke logičke funkcije koja se posebno definiše. Logički izraz je bilo koja funkcija čija vrednost može biti samo TRUE i FALSE. Napomenimo da svi argumenti funkcije moraju biti vezani za polje ili domen za koje se ograničenje definiše. Drugim rečima, oni pretstavljaju bilo celo polje ili neki njegov deo dobijen funkcijom SUBSTRING *

tačka-zarez-lista-domena :: domen
| domen ";"tačka-zarez-lista-domena

domen :: naziv_domena DEFINED_AS opis_polja

tačka-zarez-lista-logič_funkcija :: logička_funkcija
| logička_funkcija ";"tačka-zarez-lista-logič_funkcija

3.3. Primeri opisa struktura u Rečniku podataka

Ovde se na nekoliko primera još jednom diskutuje sintaksa rečnika podataka, a posebno različite opcije za iskazivanje istih činjenica, da bi se u raznim primenama istakle njihove prednosti i nedostaci.

STRUCTURES

```
DOK_ZA_PRIJEMNI_ISPIT:  <       DIPLOMA,
                                {<SVEDOČANSTVO>},
                                {<NAGRADA>}
                        >;


DIPLOMA:    <       NAZIV_ŠKOLE:CHAR (20),
                    VRSTA_ŠKOLE:VRSTE_ŠKOLA,
                    IME_KAND,
                    DATUM_DIPL:DATE
            >;

SVEDOČANSTVO:   <       NAZIV_ŠKOLE,
                        VRSTA_ŠKOLE,
                        IME_KAND,
                        DATUM_SVED,
                        {<      NAZIV_ŠKOL_PRED,
                                OCENA_ŠKOL_PRED:INT(1) IN (1,2,3,4,5)
                        >},
                        PROSEK: REAL(1,2) < 5.00
                >;
```

FIELDS

```
NAZIV_ŠKOLE: CHAR(20);
VRSTA_ŠKOLE: VRSTE_ŠKOLA;
IME_KAND: CHAR(25);
DATUM_DIPL: DATE;
DATUM_SVED: DATE;
NAZIV_ŠKOL-PRED: CHAR(15);
OCENA_ŠKOL_PRED: INT(1) IN (1,2,3,4,5);
PROSEK: REAL(1,2) < 5.00

DOMAINS

VRSTE_ŠKOLA: CHAR(20)
                    IN ('GIMNAZIJA', 'SREDNJETEHNIČKA', 'OSTALE')
```

Očigledno je da ovakvi "*mešoviti*" zapisi u Rečniku, gde se ponekad daje samo naziv polja, ponekad uz naziv i domen, a ponekad uz semantički domen i njegova definicija, otežavaju čitanje i razumevanje Rečnika. Mogućnost de se odmah uz naziv polja definiše i njegov domen i ograničenje, a uz semantički domen odmah da i njegova definicija, veoma je pogodna kada se Rečnik kreira pomoću CASE alata. Tada se na istom ekranu, pri definiciji polja daju i njegov domen, definicija domena i ograničenje, nije neophodno ići kroz poseban postupak. Međutim, pri ručnom formiranju i prikazivanju Rečnika treba izbegavati "*mešovite*" zapise, odnosno treba usvojiti sledeću praksu:

- u opisu struktura navode se samo nazivi polja, bez njihovih domena i bez definicije domena,
- definiše se posebna tabela za opis polja sa kolonama NAZIV POLJA, DOMEN (predefinisani ili semantički) i OGRANIČENJE,
- definiše se posebna tabela za opis semantičkih domena sa kolonama NAZIV DOMENA PREDEFINISANI DOMEN i OGRANIČENJE

Ako se usvoji ovakva praksa gornji primer Rečnika bi izgledao:

```
STRUCTURES

DOK_ZA_PRIJEMNI_ISPIT:  <       DIPLOMA,
                                {<SVEDOČANSTVO>},
                                {<NAGRADA>}
                        >;

DIPLOMA:    <       NAZIV_ŠKOLE,
                    VRSTA_ŠKOLE,
                    IME_KAND,
                    DATUM_DIPL
            >;


SVEDOČANSTVO:   <       NAZIV_ŠKOLE,
                        VRSTA_ŠKOLE,
                        IME_KAND,
                        DATUM_SVED,
                        {<      NAZIV_ŠKOL_PRED,
                                OCENA_ŠKOL_PRED
                        >},
                        PROSEK
                >;
```

FIELDS

NAZIV POLJA	DOMEN	OGRANIČENJE
NAZIV_ŠKOLE	CHAR(20) -	
VRSTA_ŠKOLE	VRSTE_ŠKOLA-	
IME_KAND	CHAR(25) -	
DATUM_DIPL	DATE -	
DATUM_SVED	DATE -	
NAZIV_ŠKOL-PRED	CHAR(15);-	
OCENA_ŠKOL_PRED	INT(1)	IN (1,2,3,4,5)
PROSEK	REAL(1,2)	< 5.00

DOMAINS

NAZIV DOMENA	PREDEFINISANI DOMEN	OGRANIČENJE
VRSTE_ŠKOLA	CHAR(20)	IN ('GIMNAZIJA', 'SREDNJE TEHNIČKA', 'OSTALE')

4. MINI SPECIFIKACIJE - SPECIFIKACIJA LOGIKE PRIMITIVNIH PROCESA

Primitivni procesi su procesi na najnižem nivou dekompozicije. Oni su po pravilu sekvencijalni, redosled aktivnosti u okviru njih je definisan, pa se za njihovu specifikaciju moraju koristiti neki alati za specifikaciju sekvencijalnih procesa, ili kako se obično zovu alati za opis logike procesa.

Postoji čitav skup ovih alata, počev od Dijagrama toka programa ("Flowchart"), preko Nassi Shneiderman-dijagrama, tabela odlučivanja, stabla odlučivanja, raznih vrsta strukturnih jezika i pseudokodova. Najčešće se koristi neka vrsta strukturnog prirodnog jezika ("Structured English" ili "Strukturni srpski") ili pseudokoda.

Ponekad se pravi razlika između strukturnog prirodnog jezika i pseudokoda, mada se oni u osnovi baziraju na istim principima. Naime, strukturni prirodni jezik koristi rečnik nekog prirodnog jezika (srpskog, na primer), a za konstrukciju rečenica i složenijih sklopova koristi strukture struktuiranog programiranja, sekvenciju, selekciju i iteraciju, predstavljajući ove strukture uobičajenim "ključnim rečima" (BEGIN, END, DO WHILE, IF ... THEN.... ELSE i drugim sličnim). Pseudokod je bliži programskim jezicima jer više koristi rečnik (ključne reči) nekog izabranog programskog jezika. Ovde se neće praviti razlika između pseudokoda i strukturnog prirodnog jezika.

U ovom materijalu, za opis logike primitivnih procesa koristiće se pseudokod.

4.1. Pseudokod

Zanemarujući eventualne manje razlike između pojmova "strukturni prirodni jezik" i "pseudokod", možemo reći da je pseudokod stuktuirani prirodni jezik, jezik koji koristi rečnik prirodnog jezika, a čije su konstrukcije struktuirane pomoću koncepata struktuiranog programiranja. Prirodni jezik nije pogodno sredstvo za specifikaciju logike procesa zbog svoje nepreciznosti i višeznačnosti i zato ga je neophodno struktuirati. Na primer iskaz u prirodnom jeziku:

"Svaki komitent banke koji ima na računu više od 10.000, čiji je srednji mesečni bilans veći od 5.000 ili koji poseduje račun više od pet godina ...."

može se interpretirati barem na dva načina:

(a) "Svaki komitent (koji ima na računu više od 10.000 AND srednji mesečni bilans veći od 5.000) OR poseduje račun više od pet godina.."

(b) "Svaki komitent koji ima na računu više od 10.000 AND (srednji mesečni bilans veći od 5.000 OR poseduje račun više od pet godina..)"

Zbog toga je za formiranje složenijih konstrukcija u specifikaciji logike primitivnih procesa neophodno koristiti pseudokod.

Osnovne strukture za struktuiranje prirodnog jezika su:

(**I**) Sekvencija. Aktivnosti u sekvencijalnom bloku se odvijaju po redosledu navođenja. Ponekad je pogodno da se i sekvencijalni blok akcija ograniči sa ključnim rečima BEGIN i END. Na primer, grubi pseudokod za proces Evidentiranje kandidata bi mogao da bude

BEGIN
Unesi podatke o diplomi;
Unesi podatke sa svedočanstava;
Unesi podatke o nagradama;
Ažuriraj datoteku KANDIDATI_ZA_UPIS;
END;

(**II**) Selekcija. Redosled aktivnosti zavisi od nekog uslova. Ako je uslov ispunjen obavlja se jedan, a ako nije drugi blok akcija. Opšti iskaz selekcije je

IF uslov THEN blok_akcija_1 ELSE blok_akcija_2;

Na primer,

IF BROJ POENA > 85 Upiši kandidata ELSE Odbi kandidata;

(**III**) Case struktura je specijalni slučaj selekcije kada se u zavisnosti od vrednosti jednog parametra može izvršavati više različitih blokova akcija. Opšti iskaz za ovu strukturu je:

CASE parametar OF
vrednost_parametra_1: bloka_akcija_1
vrednost_parametra_2: bloka_akcija_2
...
...
vrednost_parametra_n: bloka_akcija_n

Na primer,

CASE SEMESTAR OF
1: Stavi_studenta u grupu za prvu godinu
3: Stavi_studenta u grupu za drugu godinu
5: Stavi_studenta u grupu za treću godinu
7: Stavi_studenta u grupu za četvrtu godinu;

(**IV**) Iteracija. Blok akcija se ponavlja dok se neki uslov ne ispuni ili dok se akcije ne obave za sve objekte nekog skupa. Osnovni oblici ove strukture su:

DO WHILE uslov blok_akcija - uslov se ispituje pre svakog izvršenja bloka akcija, moguće je da se blok akcija ne izvrši nijednom,

DO UNTIL uslov blok_ akcija - uslov se ispituje na kraju svakog izvršenja bloka akcija, blok akcija se izvršava barem jedanput.

FOR EACH naziv_ skupa_ objekata DO blok_akcija

Ponekad se ključna reč DO zamenjuje sa REPEAT, PERFORM ili LOOP. Uopšte ključne reči za pseudokod mogu se izabrati kao standard u pojedinim organizacijama, tako da budu bliske rečima programskog jezika koji se koristi.

Na primer, sračunavanje prosečne ocene za sve studente bi bilo opisano pseudokodom

```
FOR EACH STUDENT DO
Učitaj rekord studenta;
SUMA := 0.;
BROJPRED := 0;
DO WHILE Postoje položeni predmeti
SUMA := SUMA + OCENA;
BROJPRED := BROJPRED + 1;
END WHILE;
PROSOCENA := SUMA/BROJPRED;
Štampaj ime studenta i prosečnu ocenu;
END FOR;
```

Nadalje slede primeri opisa logike primitivnih procesa za IS studentske službe.

EVIDENTIRANJE KANDIDATA

```
Prihvati DIPLOMU_SRED_ŠKOLE;
IF VRSTA_ŠKOLE ne odgovara THEN formiraj NEPRIHVATLJIV_DOKUMENTI;
FOR EACH SVEDOČANSTVO DO
Prihvati SVEDOČANSTVO
* Provera da li je PROSEK u svedočanstvu dobro sračunat*
SUMA := 0.; BROJPRED := 0;
DO WHILE Postoje predmeti u SVEDOČANSTVO
SUMA := SUMA + OCENA;
BROJPRED := BROJPRED + 1;
END WHILE;
PROSOCENA := SUMA/BROJPRED;
IF PROSEK NOT EQUAL PROSOCENA THEN
formiraj NEPRIHVATLJIV_DOKUMENTI;
Ubaci podatke sa SVEDOČANSTVO u KANDIDATI_ZA_UPIS;
END FOR;
FOR EACH NAGRADE DO
IF IME_NAGRADE odgovarajuće THEN Ubaci podatke sa NAGRADE u
KANDIDATI_ZA_UPIS;
END FOR;
```

5. METODOLOGIJA MODELIRANJA PROCESA

U prethodnim poglavljima prikazani su osnovni koncepti metode SSA, odnosno alati kojima se opisuju informacioni procesi u nekom realnom sistemu. Ovo poglavlje ima za cilj da diskutuje mogućnosti i način primene ovih alata u praksi modeliranja procesa.

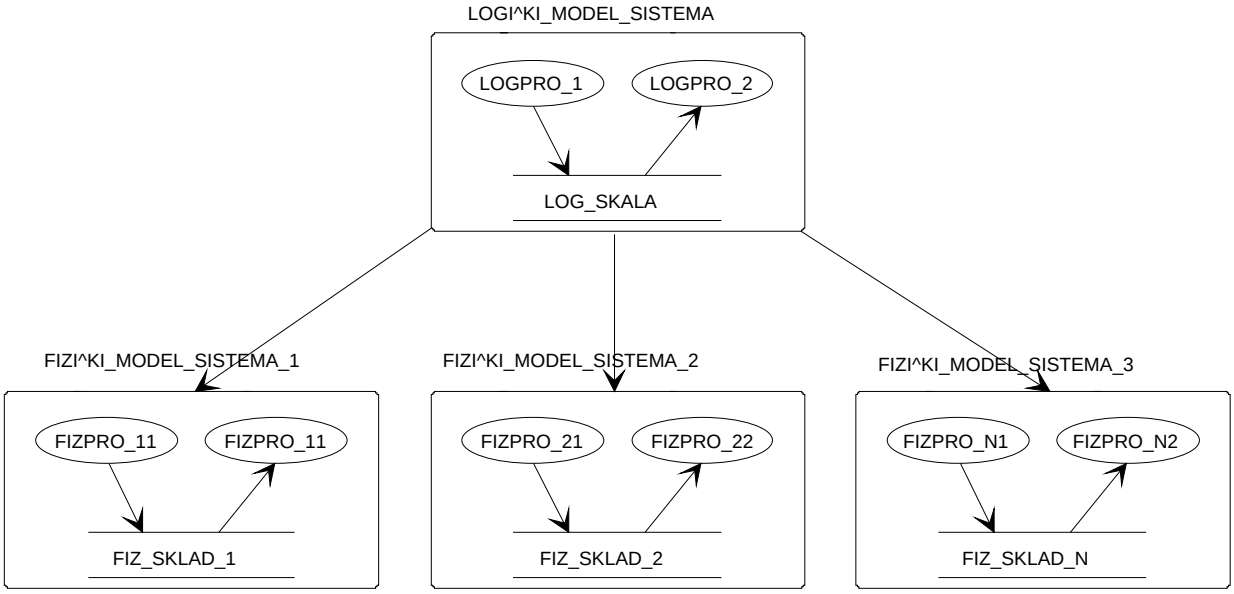
Modeliranje procesa, kao i svako drugo modeliranje u nauci i tehnici, je visoko kreativna delatnost u kojoj analitičar pretstavlja svoje znanje o sistemu koji se modelira, preko koncepata "modela" (metode) koji se koristi (metode SSA). Zbog toga uspešno modeliranje zahteva, s jedne strane, dobro poznavanje realnog sistema, a sa druge dobro poznavanje metoda i tehnika koje se koriste. Pored toga rezultat koji se dobija zavisi, naravno, i od iskustva i sposobnosti analitičara, od vremena kojim se raspolaže, organizovanosti sistema koji se analizira, komunikacije sa korisnicima sistema i drugih sličnih faktora.

Upravo zbog toga što je modeliranje kreativna delatnost ne mogu se dati definitvne metode, odnosno "recepti" u kojima bi se, korak po korak, u potpunosti definisao celokupan postupak. Umesto toga, ovde se daje niz metodoloških preporuka koje mogu značajno da olakšaju nalaženje adekvantnog modela infomacionih procesa nekog realnog sistema.

5.1. Fizički i logički modeli procesa

Cilj projektovanja i uvođenja nekog informacionog sistema nije da se automatizuju (uvođenjem računara) postojeći informacioni procesi, već da se na najbolji mogući način, uz definisana ograničenja, informaciono podrže suštinski procesi u postojećem realnom sistemu. Strukturna sistemska analiza treba da identifikuje i precizno opiše logički model procesa, odnosno suštinske procese realnog sistema i na taj način da definiše ŠTA budući informacioni sistema treba da da, dok sledeće faze razvoja sistema treba da odgovore na pitanje KAKO će se ti zahtevi realizovati u zadatom tehnološkom i organizacionom okruženju, odnosno da definišu novi fizički model procesa. Drugim rečima, logički model procesa pretstavlja specifikaciju IS, dok fizički model opisuje implementaciju zadate specifikacije u zadatom tehnološkom i organizacionom okruženju. U tom smislu postojeći informacioni sistem predstavlja prethodnu implementaciju, odnosno prethodni fizički model logičkog modela nekog informacionog sistema.

Odnos logičkog modela, fizičkoh modela pretstavljen je na Slici 17.

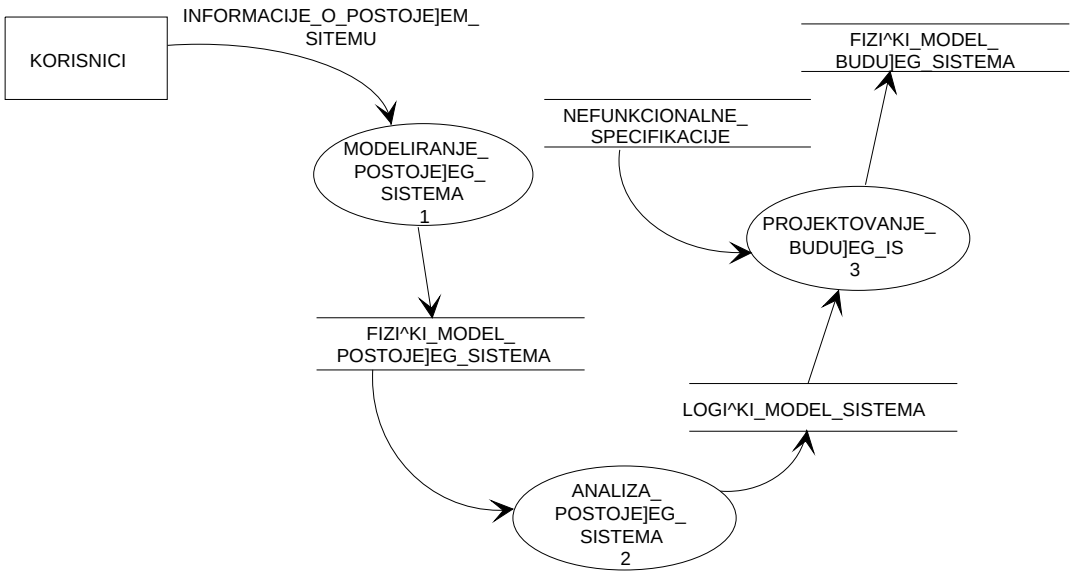


Slika17. Odnos logičkog i fizičkih modela IS

Osnovni cilj metode SSA, nalaženje logičkog modela IS, može se ostvariti bilo "direktnim modeliranjem", na osnovu poznavanja suštinskih procesa realnog sistema, bilo "snimanjem", odnosno izvlačenjem logičkog iz postojećeg fizičkog modela IS. U praksi se, naravno, kombinuju ova dva pristupa - polazeći od nekog opšteg "teorijskog" modela određenog tipa (vrste) realnog sistema (proizvodnog preduzeća, trgovine, banke i sl.), analizira se konkretan postojeći sistem, da bi se utvrdile njegove specifičnosti i posebni zahtevi.

5.2. Izdvajanje logičkog iz fizičkog modela procesa

Potpuni postupak SSA u ovom pristupu prikazan je DTP-om na Slici 18.

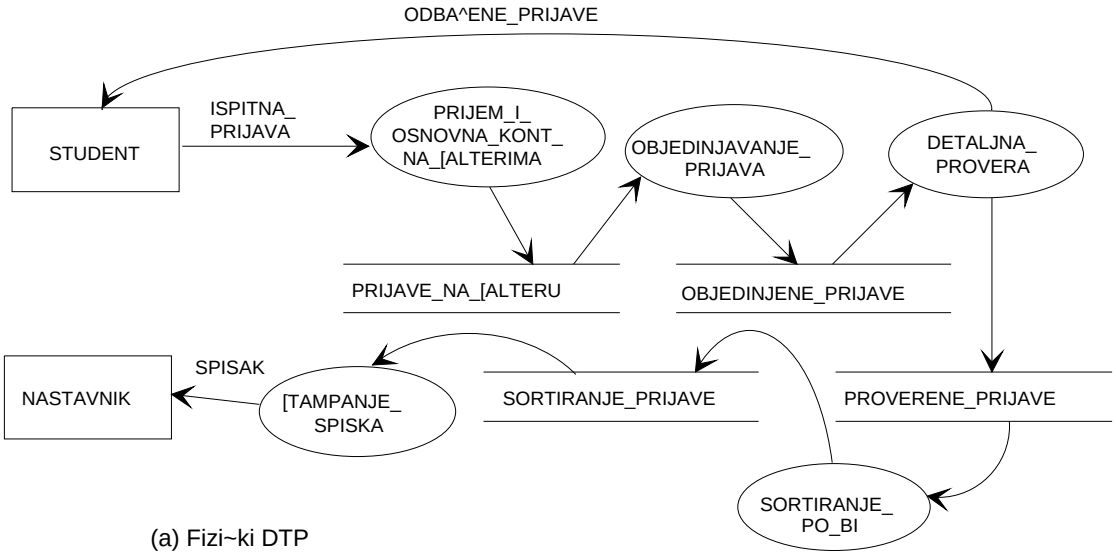


Slika 18. Specifikacija IS na osnovu snimanja postojećeg sistema

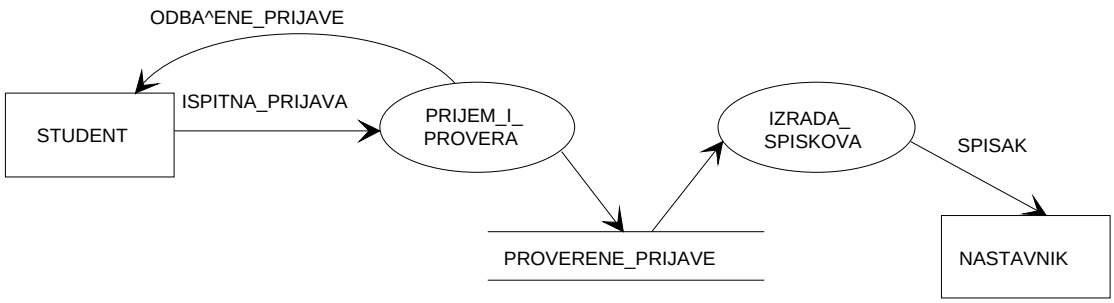
Na osnovu "snimanja" postojećeg IS definiše se fizički model procesa u postojećem IS. Ovaj model sadrži ne samo suštinske procese u sistemu i odgovarajuće tokove i skladišta podataka, već i dopunske ("posredničke") procese, skladišta i tokove koji su posledica konkretne implementacije, konkretnog tehnološkog i organizacionog okruženja. Ključna aktivnost u ovoj strategiji je analiza postojećeg sistema ("snimka") i formiranje logičkog modela sistema i ona se svodi na uklanjanje procesa, tokova i skladišta podataka koji su rezultat postojeće organizacije i tehnologije. Na osnovu ovog logičkog modela sistema i ograničenja koja nemeće buduća tehnološka i organizaciona rešenja (koja se nazivaju "nefunkcionalne specifikacije" i o kojima će kasnije biti viće reči), definiše se fizički model budućeg sistema.

Osnovni nedostaci ovog pristupa ogledaju se u tome što, sa jedne strane kompletno i detaljno snimanje postojećeg stanja traži mnogo truda i vremena, a sa druge strane, što postupak izdvajanja logičkog od fizičkog modela sistema praktično najviše zavisi od sposobnosti i veštine analitičara, kao i sposobnosti korisnika da apstrahuje suštinu sistema i da svoje neposredne zahteve ne bazira na postojećoj ili pretpostavljenoj tehnologiji.

Primer fizičkog DTP prikazan je na Slici 19a. Očigledno je da su procesi Prijem i osnovna kontrola prijava na šalterima, Objedinjavanje prijava, Sortiranje po broju indeksa, Štampanje spiska, fizički procesi. Odgovarajući DTP sa samo suštinskim procesima i skladištima dat je na Slici 19b.



(a) Fizi~ki DTP



(b) Logi~ki DTP

Slika 19. Primer fizičkog i logičkog DTP

5.3. Direktno modeliranje logičkog sistema

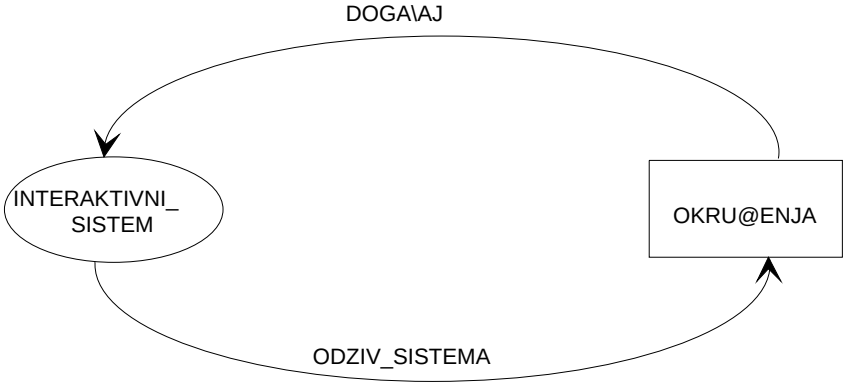
Metoda direktnog modeliranja logičkog sistema zasniva se na dobrom poznavanju vrste sistema koji se modelira. Pri tome se polazi od nekog opšteg "teorijskog" modela te vrste sistema, a zatim se analiziraju specifične karakteristike konkretnog informacionog sistema.U metodi direktnog modeliranja obično se koriste sledeća dva dopunska metodološka pristupa:

(i) analiza odziva sistema na specifične događaje,

(ii) analiza "životnog ciklusa" osnovnih delatnosti i resursa u sistemu.

5.3.1. Analiza odziva sistema na specifične događaje

Informacioni sistem je interaktivan sistem, on deluje na okruženje i okruženje deluje na njega. Svaka promena u okruženju predstavlja dogadjaj koji deluje na sistem tako što izaziva niz akcija koje predstavljaju reagovanje, odziv sistema na dati dogadjaj (Slika 20).



Slika 20. Odziv sistema na specifične događaje

Većina sistema za obradu podataka spada u interaktivne sisteme sa planiranim odzivom za koje je unapred definisan skup dogadjaja koji deluju na sistem i skup akcija, odziva sistema na te dogadjaje.

Dogadjaji koji deluju na sistem mogu biti spoljni dogadjaji koje izazivaju objekti iz okruženja , vremenski (temporalni) dogadjaji koji nastaju jer je nastupio odredjeni vremenski trenutak, ili interni događaji koji se "okidaju" kada sistem dođe u neko specifično stanje. U primeru studentske službe iz prethodnog poglavlja spoljni dogadjaji su prijavljivanje ispita i ispostalvjanje zahteva za nekim uverenjem od strane studenata, dok je vremenski dogadjaj datum početka novog semestra kada sistem treba da formira nastavne grupe. Primer internog događaja mogao bi da bude pad zaliha nekog proizvoda ispod predviđenog nivoa, što za posledicu ima otpočinjanje svih aktivnosti naručivanja.

Realni sistemi su implementirani u nekoj konkretnoj tehnologiji, koja je okarakterisana procesorima koji izvršavaju aktivnosti sistema i skladištima podataka u kojima se čuvaju podaci neophodni za izvršavanje procesa. S obzirom da svaka realna tehnologija ima i odredjena ograničenja u realnom sistemu postoje dve vrste procesa (aktivnosti) i dve vrste skladišta podataka.

Prvu vrstu procesa čine oni procesi koje bi sistem morao da izvršava čak i u slučaju da je implementiran u "idealnoj" tehnologiji, tj. takvoj tehnologiji koja nema ni vremenska ni prostorna ograničenja. Takvi procesi odražavaju suštinu, svrhu posmatranog sistema i nazivaju se suštinski (esencijalni) procesi. Za razliku od suštinskih procesa koje bi morali da postoje u bilo kojoj implementaciji sistema, posrednički procesi čine drugu vrstu procesa koji zavise od konkretne tehnologije u kojoj se vrši implementacija.

Na sličan način možemo sada definisati i pojam suštinskog skladišta podataka (esencijalne memorije) koje sadrži samo one podatke koji su neophodni za izvršavanje esencijalnih procesa, za razliku od fizičkih skladišta podataka čiji je broj i sadržaj delom diktiran tehnologijom. S tim u vezi razlikujemo dve vrste esencijalnih procesa: osnovne (fundamentalne) procese koji, sa stanovišta okruženja opravdavaju postojanje sistema, tj. generišu izlazne tokove ka okruženju i procese održavanja koji prihvataju i skladište podatka proizvedene od strane okruženja i/ili fundamentalnih procesa u esencijalnoj memoriji. Suštinski procesi jednog sistema se, medjutim retko javljaju u "čistom" obliku (kao isključivo osnovni procesi ili procesi održavanja) već su obično složeni procesi koji objedinjuju obe vrste aktivnosti. U primeru studentske službe proces izdavanje-uverenja predstavlja čist osnovni proces, proces UPIS-GODINE je proces održavanja, dok proces IZRADA-ISPITNIH-SPISKOVA predstavlja složen proces.

Metoda direktnog modeliranja logičkog sistema sastoji se u identifikaciji suštinskih procesa, raščlanjivanju sistema na suštinske procese. Svaki suštinski proces sastoji se od skupa akcija kojim sistem reaguje na jedan i samo jedan dogadjaj (spoljni,vremenski ili interni). Skup akcija grupisanih u jednom procesu mora biti takav da sistem, u idealnoj tehnologiji, po njihovom izvršenju prelazi u stanje mirovanja dok ne nastupi neki drugi dogadjaj. Svi suštinski procesi u ovako dekomponovanom sistemu komuniciraju isključivo preko suštinske memorije. Suštinska memorija pretstavlja osnovne objekte (fizičke ili koncetualne) posmatranog sistema i sadrži samo podatke koji opisuju te objekte.

Na osnovu izloženog, očigledno je da su snovni koraci u ovakvoj strategiji direktnog modeliranja sistema:

(1) Identifikacija osnovnog cilja, svrhe postojanja posmatranog sistema
(2) Identifikacija osnovnih (fundamentalnih) procesa sistema
(3) Identifikacija neophodnih informacija
(4) Identifikacija procesa održavanja

U principu, svrha svakog realnog sistema je da u interakciji sa okruženjem na odredjen, planiran način ostvari svoje ciljeve. Prema tome, na svaki spoljni ili vremenski dogadjaj sistem reaguje na način kojim najbolje ostvaruje svoje ciljeve, preko planiranog odziva. Kako se odziv na dogadjaje ostvaruje putem fundamentalnih procesa, to je neophodno da se identifikuju dogadjaji koji izazivaju fundamentalne procese sistema i da se ti procesi modeliraju tako da u najvećoj mogućoj meri ostvaruju ciljeve sistema.

U sledećem koraku treba identifikovati sve podatke koji su neophodni za odvijanje fundamentalnih procesa. To je najbolje uraditi tako što se prvo definišu podaci koje fundamentalni proces treba da proizvede a zatim se utvrde podaci na osnovu kojih to radi. Sistem takve, neophodne podatke dobija iz okruženja prilikom nastanka nekog dogadjaja ili ih stvaraju fundamentalni procesi u toku generisanja odziva sistema na neki dogadjaj. Po identifikaciji podataka, neophodnih za odziv sistema treba utvrditi koje od tih podataka treba čuvati u suštinskoj (esencijalnoj) memoriji. Konačno, u poslednjem koraku treba definisati procese održavanja čiji je zadatak da skladište i ažuriraju podatke u skladištima podataka.

Može se reći da je možda najveća vrednost metode opisane u ovom odeljku precizna i jasna klasifikacija komponenti koje čine jedan informacioni sistem, što analitičaru pomaže da svoja istraživanja i komunikaciju sa korisnikom kanališe u cilju što efikasnije identifikacije i specifikacije zahteva korisnika.

5.3.2. Analiza "životnog ciklusa" osnovnih delatnosti i resursa u sistemu

Procese u nekom sistemu pogodno je podeliti na sledeće grupe:

(i) Procesi organizovanja, planiranja i upravljanja koji definišu parametre, uslove i pravila pod kojima se drugi procesi u sistemu obavljaju;

(ii) Procesi obavljanja osnovnih delatnosti, odnosno procesi čiji je rezultat neki "proizvod rada" (fizički proizvod, usluga ili informacija);
(iii) Procesi upravljanja resursima kojima se obezbeđuju resursi potrebni za obavljanje osnovnih delatnosti.

Pri analizi nekog složenog sistema, pogodno je, na prvom nivou dekompozicije upravo definisati ove tri grupe procesa.

Procesi organizovanja, planiranja i upravljanja su obično "slabo struktuirani" procesi u postojećem sistemu i u konkretnoj situaciji, teško ih je precizno definisati i pretstaviti dijagramima tokova podataka. Upravo je projekat novog IS prilika da se i ovi procesi precizno definišu, specifikuju podaci na osnovu kojih se obavljaju i podaci koje proizvode kao rezultat. Procesi organizovanja, planiranja i upravljanja obično obuhvataju:

- definisanje organizacione strukture, odnosno sistematizacija poslova i zadataka,
- izrada različitih pravilnika o načinu i postupcima obavljanja pojedinih delatnosti. (Na primer pravilnici i postupci kontrole kvaliteta u svim poslovnim funkcijama u organizaciji),
- standardizacija,
- strategijsko, odnosno dugoročno i globalno planiranje. Operativno planiranje u pojedinim poslovnim funkcijama obično se pridružuje opisu tih funkcija.

Procesi obavljanja osnovnih delatnosti identifikuju se i dalje dekomponuju na bazi analize "životnog ciklusa" proizvoda rada koji se tom delatnošću dobija. Pri tome se posebno analiziraju različiti tipovi proizvoda ili usluga koje posmatrana organizacija obavlja. Svaki tip proizvodnje ili usluge prolazi kroz sledeće četiri faze "životnog ciklusa":

(1) Priprema "rađanja";
(2) "Rađanje";
(3) Razvoj (život)
(4) Nestanak;

Analiziraju se procesi svake faze i opisuju odgovarajućim dijagramima tokova podataka. Tako se za jedu tipičnu proizvodnu radnu organizaciju, mogu definisati sledeći procesi u "životnom ciklusu proizvoda":

Priprema rađanja
- Analiza tržišta prodaje,
- Analiza konkurencije,
- Analiza tržišta nabavke,
- Analiza postojećih sopstvenih kapaciteta,
- Definisanje novog proizvodnog programa,
- Planiranje realizacije novog proizvodnog programa.

Rađanje proizvoda
- Projektovanje proizvoda (konstrukcija proizvoda)
- Razvoj tehnologije proizvodnje (definisanje tehnoloških postupaka).

Život proizvoda
- Operativno planiranje,
  - Planiranje kapaciteta,
  - Planiranje nabavke,
  - Planiranje prodaje,
- Nabavka,
  - Izbor ponuđača,
  - Ugovaranje nabavke,
  - Nabavljanje
    - Naručivanje,

- Prihvatanje,
    - Kvalitativna i kvantitativna kontrola,
    - Skladištenje,
- Upravljanje proizvodnjom,
    - Terniniranje i lansiranje,
    - Praćenje proizvodnje,

Nestanak proizvoda
Prodaja
Obrada narudžbenica,
Otprema.

Procesi upravljanja resursima se takođe struktuiraju na osnovu analize "životnog ciklusa" svakog resursa organizacije. Osnovni resursi u organizaciji su kadrovi, osnovna sredstva (oprema) i kapital. Ponekad se i informacija tretira kao resurs organizacije. Primeri procesa po "životnom ciklusu" resursa kadrovi su:

Priprema rađanja
- Periodično planiranje kadrova,
- Stipendiranje,
- Regrutovanje
    - Raspisivanje konkursa,
    - Izbor kandidata,

Rađanje
- Zasnivanje radnog odnosa,
- Zasnivanje ugovornog odnosa,

Život
- Praćenje kadrova,
    - Evidentiranje izvršenja radnih obaveza,
    - Organizacija odmora i rekreacije,
    - Praćenje odsustvovanja,
    - Praćenje školovanja i osposobljavnja,
    - Raspoređivanje na poslove i zadatke,

Nestanak
- Odlazak radnika,
- Penzionisanje.

Na sličan način se može analizirati i životni ciklus stalih resursa.

Napomenim da se u direktnom modeliranju, metoda analize "životnog ciklusa" koristi za opisivanje procesa na višim, a metoda analize odziva sistema na specifične događaje,na nižim nivoima dekompozicije

5.4. Kriterijumi dekompozicije

Verovatno najveći problem analitičara u okviru SSA je vezan za pitanje kako vršiti dekompoziciju, kada prestati sa dekompozicijom, odnosno na osnovu čega možemo zaključiti da jedan proces na DTP-u ne treba dalje dekomponovati. U literaturi se daju različite preporuke, koje su

najčešće veoma opšte i neprecizne i zasnivaju se na kriterijumu "kada se dođe do procesa čija se logika može jednostavno opisati, na primer pseudokodom ne većim od jedne stranice teksta". Međutim, ovo pitanje ima suštinski značaj i odgovor na njega se izvodi iz karaktera procesa koje dijagrami tokova podataka opisuju.

Dijagrami tokova podataka opisuju suštinske, paralelne procese u nekom sistemu. Suštinski procesi, kao što je rečeno, su procesi koji se moraju obaviti u svakoj tehnologiji i oni međusobno komuniciraju preko suštinske memorije. Samim tim što komuniciraju isključivo preko suštinske memorije, oni su i paralelni (nesekvencijalni), odnosno mogu se obavljati istovremeno, nikakav međusobni sekvencijalni odnos između njih se ne podrazumeva. Paralelizam među procesima je karakteristika odvijanja procesa u organizaciji. Na primer, u nekom preduzeću istovremeno se odvija i nabavka i prodaja i lansiranje proizvodnje i sama proizvodnja, na fakultetu istovremeno i prijem prijava i nastava i raspisivanje konkusa za nastavnike i slično. Dijagram toka podataka je sredstvo (alat) za opisivanje suštinskih paralelnih procesa u organizaciji.

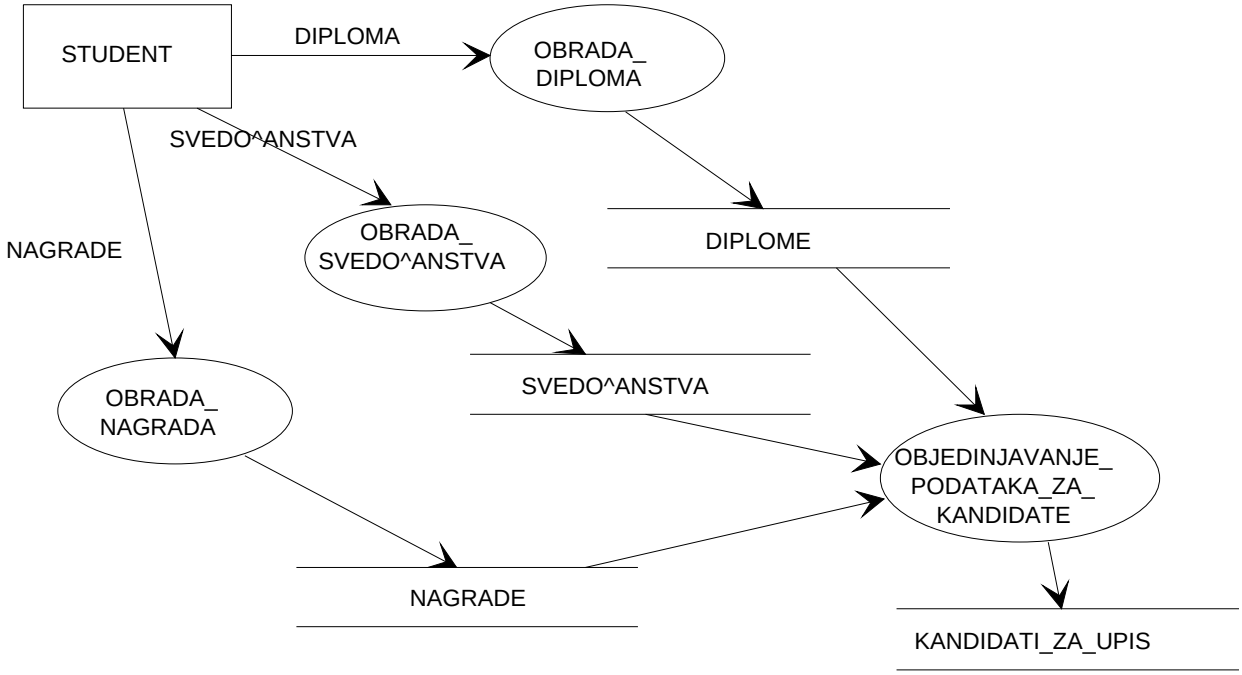
Za razliku od procesa u organizaciju, procesi koji se odvijaju u računaru su sekvencijalni, redosled njihovog obavljanja je unapred definisan. Sredstva za opis sekvencijalnih procesu su najčešće Dijagram toka programa ("Algoritam", "Flowchart") ili Pseudokod. Mada se ponekad to čini, ove alate ne treba koristiti za opis paralelnih procesa u organizaciji.Razlika između sekvencijalnih i paralelnih procesa najbolje se ilustruje kroz sledeći citat:

"Ako ikada nađete organizaciju u kojoj jedan čovek obrađuje jedan dokumenet, dok svi ostali spavaju, pa kad on završi obradu, probudi suseda, preda mu dokumenat da ovaj nastavi svoj deo posla, a on ode na spavanje, tada slobodno možete koristiti Dijagram toka programa (ili Pseudokod) za opis procesa u toj organizaciji".

Razlika između paralelnih i sekvencijalnih procesa diktira osnovni kriterijum dekompozicije u SSA - dekompoziciju treba vršiti dok se neki proces prirodno može dekomponovati na suštinske paralelne procese koji 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.

Sledeći dopunski kriterijum obično podržava prethodno osnovno pravilo: dekompozicija se vrši dok se ne dobiju procesi sa jednim ulaznim i/ili izlaznim tokom podataka. Naime, uslov da jedan proces istovremeno treba da prihvati više ulaznih ili generiše više izlaznih tokova podataka je "fizički uslov", uslov koji proizilazi iz postojeće tehnologije obrade podataka. Ovakvi tokovi u suštini pretstavljaju jedan "logički" tok, njihovu strukturu ne treba pretstavljati na DTP, već u rečniku podataka.

Na primer u opisu IS Studenste službe definisan je jedinstveni tok DOKUMENTI_ZA_PRIJEMNI_ISPIT, i na osnovu toga jedinstveni proces UPIS_GODINE. Ako bi se ovaj tok razbio na posebne tokove DIPLOMA, SVEDOČANSTVA, NAGRADE, a zatim izvršila dalja dekompozicija sistema na bazi takvih tokova, morao bi se uvesti novi pomoćni (fizički) proces kiji bi objedinjavao podatke prikupljene iz razloženog jedinstvenog logičkog toka podataka, kako je to na Slici 21 i prikazano.



Slika 21. Fizički dijagram kao rezultat nekorektne dekompozicije toka podataka.

Kao što je ranije pokazano, u dekompoziciji DTP može se vršiti i dekompozicija tokova podataka. Na osnovu odnosa strukture podataka i strukture programa koji ih obrađuju:

- agregaciji podataka <a,b,...> odgovara istovremena (sekvencijalna) obrada,
- skupu podataka {a} odgovara iteracija obrada nad elementima skupa,
- specijalizaciji (uniji) podataka [a,b,...], ogvovara grananje obrade u moguće paralelne procese,

očigledno je da se dekompozicija tokova može vršiti samo na strukturama tipa specijalizacije.

Pretodno navedeni osnovni kriterijum dekompozicije može dovesti i do složenih primitivnih procesa, procesa koji se po klasičnim kriterijumima dekompozicije ne mogu opisati "pseudokodov veličine jedne strane teksta". Međutim, metodologija specifikacije programa i njihovog automatskog generisanja na osnovu prethodno napomenutog odnosa strukture podataka i strukture programa, dozvoljava da se, bez daljih većih teškoća zadržimo i na takvom nivou dekompozicije, ne prejudicirajući buduća fizička rešenja.