# 2 Objektno orijentisana analiza

PROCES OBJEKTNO ORIJENTISANA ANALIZA (OOA) KORISTI SE ZA DEFINISANJE ključnih koncepata (ključnih apstrakcija) i veza između koncepata (mehanizama) vezanih za domen problema modeliranja realnog sistema objekata.

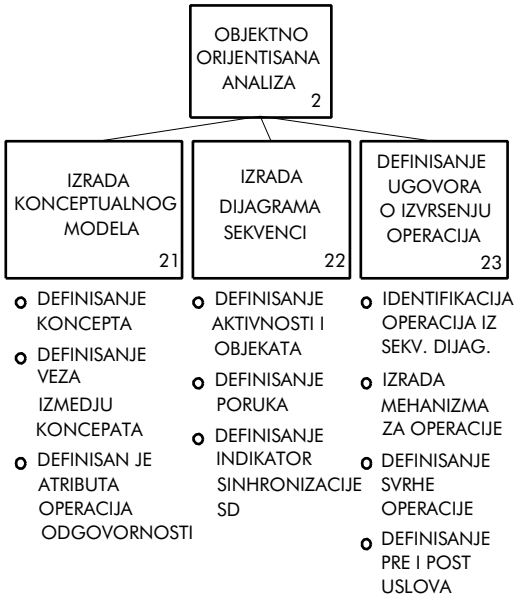
Realni sistem objekata i njihove veze modeliraju se predstavljanjem određenog broja precizno definisanih koncepata. U softverskom inženjerstvu, koncepti služe za formiranje jasnog, preciznog i potpunog modela realnog sistema (njegove specifikacije), kao uslov za implementaciju modela na računaru.

Za definisanje ključnih apstrakcija ili koncepata potrebno je uočiti termine koji opisuju karakteristične entitete domena. U okviru aktivnosti *Izrada konceptualnog modela* definišu se koncepti, asocijacije između koncepata, atributi i rečnik podataka. Za radni primer koji se posmatra, ključne apstrakcije su: Osoba, Isplata, Sertifikat i Šifrarnici (slika 2.2). Definisan koncept odgovara modelu objekti veze, koji se definiše u tradicionalnom postupku modeliranja podataka.

Za aktivnost *Izrada dijagrama sekvenci* potrebno je izvršiti generisanje i dokumentovanje dijagrama sekvenci za odgovarajuće slučajeve upotrebe po kojima se realizuju interakcije između programskih objekata.

Dijagram sekvenci je dobar za shvatanje scenarija u početnim fazama razvoja sistema, pre nego što se uoče protokoli pojedinačnih klasa. Dijagram sekvenci se *fokusira na događaje, jer se na taj način mogu uočiti i granice sistema*.

Treći bitan element je aktivnost *Definisanje ugovora o izvršenju operacija*, kojom se definišu odnosi dva ili više koncepata, kao i opis interakcija dva ili više objekata koji kroz tu iteraciju ostvaruju neki od tih zahteva. Identifikovani mehanizmi se koriste u okviru sledeće faze, za objektno orijentisan dizajn, kao metoda ili procedura (funkcija članica budućih klasa). Granica između objektno orijentisane analize (OOA) i objektno orijentisanog dizajna (OOD) pomalo je maglovita. Iako obe analize posmatraju sistem iz različitog ugla, između njih dolazi do malog preplitanja. Na slici 2.1. je prikazano stablo aktivnosti vezano za fazu *Objektno orjenitisana analiza*.




**Slika 2.1.**
*Stablo aktivnosti faze Objektno orijentisana analiza*

# Izrada konceptualnog modela

Konceptualni model opisuje domen realnog sistema i ne predstavlja model softverskog rešenja. Formalno gledano, koncept je ideja, predmet, objekt i dr. Konceptualni model se u notaciji UML-a opisuje dijagramom klasa bez operacija (što odgovara modelu "objekti veze") ili sa samo nekim elementarnim operacijama. Tako se često i naziva - dijagram klasa bez operacija.

Konceptualni model ilustruje različite kategorije stvari u domenu ne samo uloge ljudi već i stvari od interesa. Radi analize, kojoj i pripada, on doprinosi dekompoziciji domena problema, identifikujući koncepte, atribute i veze. Konceptualni model nije opis softverske komponente, naprotiv on predstavlja koncepte u realnom svetu domena problema. Zato je pogrešno i metodološki nekorektno reći da je dijagram klasa samo dijagram dopunjen metodama klase. Činjenica je da on podseća na konceptualni model i da deo svoga izgleda duguje njemu, ali o znaku jednakosti ne može biti govora. Klase u dijagramu klasa su prave softverske klase koje će u kodu programa predstavljati deklaracije klasa. Klase u konceptualnom modelu su samo rezultat pokušaja da se sagledavanjem realnog sistema prepoznaju neki entiteti koji bi mogli da konkurišu za softverske klasu, te da se prepoznaju neke veze i njihova kardinalnost. Konceptualni model će morati da pretrpi krupne promene kako bi kroz ostatak faze analize i faze dizajna rezultovao dijagramom klasa.

Izrada konceptualnog modela obuhvata [2]:

- Definisanje koncepta;
- Definisanje asocijacija između koncepata;
- Definisanje atributa, operacija i odgovornosti.

## Definisanje koncepta

Konceptom se opisuju stvari u realnom sistemu i na osnovu njih se kasnije, u fazi *Objekto orijentisanog dizajna*, definišu odgovarajuće klase i objekti koji definišu odgovarajuća softverska rešenja. Koncept se definiše preko odgovarajućih simbola, tekstualnog opisa i primera na kojima se koncept primenjuje (slika 2.2).

| Simbol koncepta | Tekstualni opis | Primer |
|---|---|---|
| OSOBA | Osobe su prevodioci | Ruski<br>Engleski<br>Francuski |

**Slika 2.2.**
*Definisanje koncepata*

Prethodna faza *Definisanje zahteva* treba da izvrši dekompoziciju funkcija tako da složeni problem izgradnje softvera podeli u jednostavnije probleme kroz definisanje primitivnih funkcija, tj. primitivnih procesa. Primenom klasične strukturne sistem-analize, sprovedenom u fazi *Definisanje zahteva*, definišu se *funkcije*, a objektno orijentisanom analizom definišu se slučajevi upotrebe koji se opisuju odgovarajućim sistemskim dijagramom sekvenci ili, ako su u pitanju složeniji procesi, dijagramom aktivnosti. U okviru definisanih dijagrama sekvenci ili dijagrama aktivnosti, za defininisani slučaj upotrebe definišu se i odgovarajući objekti koji su osnova za definisanje *koncepata* (objekti). Primarni zadatak u fazi objektno orijentisane analize je identifikovanje različitih koncepata u problemskom domenu, kao i dokumenata i rezultata u konceptualnom modelu.

Na primeru prikazanom na slici 2.3, u problemskom domenu koji se razmatra, osnovni koncepti su OSOBA, ODELJENJE, JEZIK.

**Slika 2.3.**
*Primer razmatranih osnovnih koncepata*

Imajući u vidu dosadašnji razvoj tradicionalnih metoda vezanih za modeliranje podataka i bogato iskustvo u toj oblasti, pojavili su se i odgovarjući uzori za analizu (patern analysis), koji predstavljaju moguće šablone koji pomažu u postavljanju odgovarajućih konceptualnih modela.

S druge strane, preporučuju se prethodno definisan pristup i dve strategije za identifikaciju koncepta, i to:

- nalaženje koncepta iz liste konceptualnih kategorija i
- nalaženje koncepta identifikujući imenice u tekstualnom opisu.

Prvi način je nalaženje kandidata koncepta na osnovu sledeće okvirne liste konceptualnih kategorija (slika 2.4).

| Konceptualna kategorija | Primeri |
|---|---|
| Fizički ili opipljivi objekti | OSOBA |
| Specifikacija ili opisana stvar | CERTIFIKAT |
| Mesto | ODELJENJE |
| Transakcija | ISPLATA |
| Stavka transakcije | RADNA LISTA |
| Uloge ljudi | Prevodilac, Rukovodilac Knjigovođa |
| Spremište | Finansijsko knigovodstvo |
| Stvar u spremištu | Karton prevodilaca |
| Drugi komjuterski sistemi | Web server |
| Apstrakti imenični koncepti | Prevođenje |
| Organizacija | Knjigovodstvo Odelenje prevodilaca |
| Događaji | Ispatiti, Knjižiti |
| Procesi | Obračun isplata,<br>Održavanje osnovnih podataka |
| Pravila | Način isplate |
| Katalozi | Katalog posrednika |
| Ugovori | Ugovor o delu |
| Finansijski instrumenti | Kredit |
| Manueli, Knjige | Rečnici |

**Slika 2.4.**
*Definisanje kandidata koncepata na osnovu liste konceptualnih kategorija*

Drugi način je nalaženje koncepta na osnovu pojmova, tj. analize imenica u tekstualnom opisu. Ovaj način definisanja koncepta ima nedostatke, jer ne postoji tako definisana rečenička konstrukcija iz koje bi se mogao izvući konceptualni model. Rešenje je predložena primena IDEFO metodologije i strukturne sistem-analize (o čemu je bilo više reči u prethodnom poglavlju Definisanje zahteva), kao pretpostavka za definisanje elemenata konceptualnog modela.

Imajući u vidu prethodno, može se reći da ne postoje formalna pravila, već samo niz heuristika, posmatranih na sledeći način:

- Ukoliko je pojam dovoljno bitan za domen, on predstavlja koncept. Ovo je krajnje subjektivno pravilo.
- Ukoliko je pojam vezan sa drugim pojmovima, on predstavlja koncept koji će biti u relaciji sa drugima.
- Ukoliko je dati pojam apstraktan i predstavlja generalizaciju drugih pojmova, onda je on nezavisan koncept.

- Ukoliko dati pojam ima svoja svojstva, onda je on nezavisan koncept.
- Ukoliko pojam ima svoje ponašanje bitno za ponašanje celog sistema, onda je on nezavisan koncept.
- Ukoliko instance datog pojma mogu da žive nezavisno od instanci posmatranog koncepta, onda je on nezavisan koncept.
- Ukoliko ništa od prethodnog nije zadovoljeno, treba razmotriti upotrebu atributa, a ne nezavisnog koncepta.

Dakle, za razliku od UML-a, gde se koriste termini "klasa" i "tip", u okviru objektno orijentisane analize koristi se termin "koncept". Koncept se koristi u opisu realnih stvari, dok se klasa koristi za referisanje na softverskoj specifikaciji ili implementaciji.

Na osnovu definisanih koncepata potrebno je uspostaviti i odgovarajuće veze između njih, što predstavlja predmet razmatranja u sledećem poglavlju.

## Definisanje veza između koncepata

Definisanjem veza između koncepata uspostavljaju se asocijacije između prethodno definisanih koncepata. U okviru UML-a, asocijacije se opisuju kao "strukturne relacije" između objekata za različite tipove.

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

Asocijacije se mogu definisati na osnovu izbora iz zajedničke asocijativne liste, kao što je prikazano na slici 2.5.

| Kategorija | Primer |
|---|---|
| A je fizički deo od B | OSOBA – MUŠKARAC |
| A je logički deo od B | OSOBA – ISPLATA |
| A je fizički sadržan u B | PREDUZEĆE – ISPLATA |
| A je logički sadržan u B | R. LISTA – STAVKA RL |
| A je opis za B | OSOBA – DOSIJE |
| A je stavka transakcije ili izveštaj za B | OSOBA – ISPLATA |
| A je član za B | PILOT – AVION |
| A je organizaciona podjedinica za B | SEKTOR – ODELJENJE |
| A koristi ili služi B | PILOT – AVION |
| A komunicira sa B | KUPAC – PRODAVAC |
| A je povezan u transakciji B | PUTNIK – KARTA |
| A je sledeći za B | GRAD – GRAD |
| A je vlasnik B | AVION – AERODROM |

**Slika 2.5.**
*Definisanje asocijacija izborom iz zajedničke asocijativne liste*

Iz ovog spiska izvlače se tri asocijacije kao asocijacije najvišeg prioriteta:

- A je fizički ili logički deo od B;
- A je fizički ili logički sadržan u B;
- A je slog podataka (recorded) u B.

Predmet daljih razmatranja, definisanih u objektno orijentisanom modeliranju, jesu tri vrste odnosa, i to:

- *zavisnosti*, koji predstavlja korišćenje odnosa između koncepata;
- *generalizacije*, koji predstavlja uopštene koncepte sa njihovim specijalizacijama;
- *asocijacije*, koji predstavlja strukturalne odnose među objektima.

Ove tri vrste odnosa pokrivaju većinu važnih načina u kojima stvari sarađuju jedne s drugima. Svaki od ovih odnosa obezbeđuje drugačiji način kombinovanja modela i svaki će biti posebno opisan.

Zavisnost (*Dependency*) jeste relacija korišćenja koja govori da promena specifikacije jedne stvari (npr. klase A) može da utiče na drugu stvar koja je koristi (npr. klasu B), ali ne obavezno i suprotno. Kasnije će se videti da je zavisnost opšta relacija koja se može uspostaviti između svih stvari jezika UML, naročito između paketa.

Zavisnost se prikazuje isprekidanom linijom sa otvorenom strelicom na vrhu, koja polazi od stvari koja je zavisna i ide ka stvari od koje ova zavisi.

Grafička notacija prikazana na slici 2.6. čita se na sledeći način: koncept A zavisi od koncepta B.

**Slika 2.6.**
*Grafička prezentacija veze zavisnosti*

Dakle, kada instance koncepta A koriste usluge instance koncepta B tako što pozivaju njihove operacije i ne postoje čvršće, semantičke, strukturne veze, nego veze traju samo tokom poziva operacije (instanca koncepta A dobija kao parametar neke svoje operacije referencu na B), onda između njih postoji (samo) zavisnost: A zavisi od B.

*Generalizacija* je odnos između osnovne (generalne) stvari (nazvane kao superklasa ili roditelj) i još specifičnije vrste ove stvari (nazvane podklasa ili dete). Generalizacija se ponekad naziva "je-vrste" ("is-a-kind-of") odnos. Generalizacija znači da se objekti deteta mogu koristiti bilo gde, gde se može pojaviti roditelj, ali ne i obrnuto. Drugim rečima, generalizacija znači da je dete zamenljivo za roditelja. Dete nasleđuje osobine svojih roditelja, naročito njihove atribute i operacije. Često, ali ne i uvek, dete ima još neke atribute i operacije, pored onih koje je nasledilo od roditelja.

Operacija deteta koja ima istu signaturu koju ima operacija roditelja, redefiniše operaciju roditelja. Ova pojava poznata je kao polimorfizam.

Grafička notacija generalizacije je prikazana na slici 2.7. i čita se: koncept A je dete, a koncept B je roditelj.

**Slika 2.7.**
*Grafička prezentacija veze generalizacija*

*Asocijacija* je strukturalni odnos koji specificira da su objekti jedne stvari povezani sa objektima druge. Ova veza je jača nego zavisnost i tipično postoji duže nego samo tokom poziva jedne operacije, kao kod veze klijent-server, iako, naravno, asocijacija najčešće služi da se preko veze objekata razmenjuju poruke.

Asocijacija je relacija između koncepata. Instance tih koncepata vezane su instancama te asocijacije. Instanca asocijacije zove se veza (*Link*).

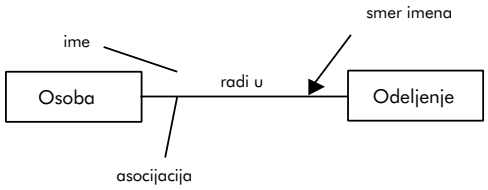
Kada između dva koncepta postoji asocijacija, onda to znači da se, na neki način, može doći od objekata jednog koncepta do objekata drugog koncepta koji su za njih vezani.

Asocijacija koja povezuje tačno dva koncepta naziva se binarna asocijacija. Asocijacija koja povezuje više od dva koncepta zove se n-arnom asocijacijom i poželjno ju je izbegavati zbog otežane realizacije. Asocijacija se grafički predstavlja kao puna linija koja povezuje koncepte.

Svaka asocijacija na nivou koncepta ima sledeća svojstva:

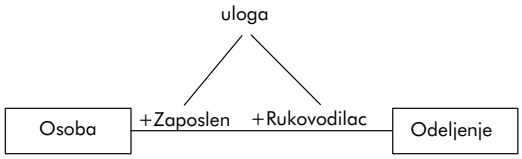
- ime asocijacije,
- ulogu,
- multiplikativnost i
- agregaciju.

*Ime* asocijacija opisuje prirodu odnosa. Da ne bi bilo dvosmislenosti u vezi sa njegovim značenjem, može se dati smer imenu pomoću trougla koji daje smer prilikom čitanja imena (čita se sleva nadesno ili odozgo nadole), kao što pokazuje slika 2.8. Grafički prikaz dat na slici 2.8. opisuje se rečenicom: "Osoba radi u Odeljenju".



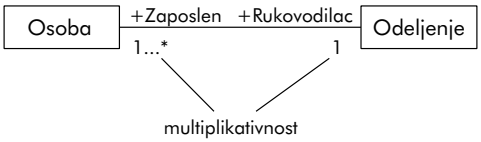

**Slika 2.8.**
*Ime asocijacije*

Uloga (*role*). Kada koncept učestvuje u asocijaciji, može da ima specifičnu ulogu u tom odnosu. Na slici 2.9. koncept Osoba ima ulogu Zaposlen, koji je u vezi sa konceptom Odeljenje, koja ima ulogu Rukovodilac.




**Slika 2.9.**
*Uloge*

*Multiplikativnost* (kardinalnost) definiše koliko instanci tipa A može da bude asocirano sa jednom instancom tipa B. U mnogim situacijama modeliranja veoma je važno da se utvrdi koliko objekata može biti povezano preko jedne instance asocijacije. Ovo "koliko" naziva se multiplikativnot (višestrukost) asocijacione uloge, a piše se kao izraz dat na slici 2.10. Kada se odredi višestrukost na jednom kraju, na drugom kraju moraju da budu objekti, pa se višestrukost određuje i na drugom kraju. Višestrukost se prikazuje kao: tačno jedan (1), više (0..*), ili jedan ili više (1..*), čak može da se stavi neki broj (recimo 3).




**Slika 2.10.**
*Prikaz multiplikativnosti*

*Agregacija* je samo specijalna vrsta asocijacije, definisana je romboidom i predstavlja strukturalni odnos, što znači da su oba koncepta na istom nivou. Agregacija definiše odnos "celina/deo", tj. odnos "ima" (has-a), što znači da objekt celine ima objekte delova.

Značenje agregacije je u potpunosti konceptualno. Ono samo govori da je jedan koncept pod imenom "celina" (pripada mu romboid), a drugi koncept pod imenom "deo", ništa ni više ni manje od toga. Ovo nikako ne znači da agregacija menja značenje - ni navigabilnosti ni multiplikativnosti asocijacije. Takođe, ona ne uvodi nikakva posebna ograničenja u pogledu ekskluzivne pripadnosti dela celini, niti vezanosti životnog veka objekta-celine i objekta-dela.

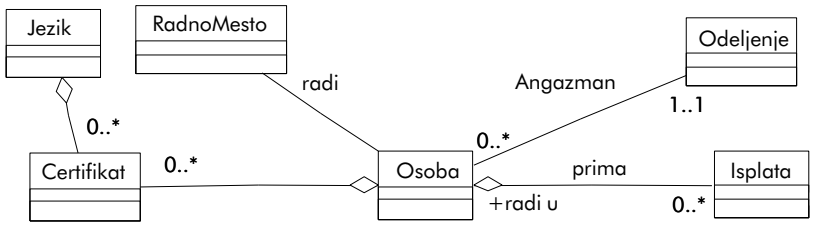
Kao što se može videti na slici 2.11, objekt tipa Osoba ima nijednu ili više isplata. Veza se implementuje preko referenci na Isplatu koje pamti Osoba.

**Slika 2.11.**
*Agregacija*

Posebna vrsta agregacije, tj. čvršća agregacija je *kompozicija* (*Composition*). Ona unosi posebnu semantiku u relaciju celine i dela. Kod nje je strogo definisano pitanje vlasništva objekta-dela i njegovog životnog dela. Kod kompozicije, životni vek objekta-dela je ugrađen u životni vek objekta-celine, što znači da on ne može da nadživi celinu. Dakle, sa nestankom objekta-celine nestaju i njegovi delovi. Ukoliko je kardinalnost na strani dela veća od nule (recimo tačno jedan), onda objekt-deo i nastaje sa objektom-celinom.

Kod kompozicije, objekti-delovi su u ekskluzivnom vlasništvu samo jednog objekta-celine, koji je odgovoran za njihovo kreiranje i uništavanje. Kod obične agregacije, objekte-delove može deliti više objekata-celina. Dakle, kod kompozicije je multiplikativnost na strani celine uvek implicitno tačno jedan. Kompozicija se prikazuje kao i agregacija, samo sa ispunjenim rombom.

Imajući u vidu rečeno, na slici 2.12. je prikazan realizovani primer koncepta definisan konceptima i vezama.




**Slika 2.12.**
*Primer realizovanih koncepata i veza*

Veza na nivou koncepta može i ne mora da bude implemetirana. O vezama će detaljno biti više reči u fazi Objektno orijentisan dizajn.

## Definisanje atributa, elementarnih operacija i odgovornosti

Konceptima se prave apstrakcije koje su izvučene iz problema koji se rešava. Svaka od ovih apstrakcija je deo rečnika sistema, tako da one zajedno predstavljaju stvari koje su važne i za korisnike i za one koji projektuju sistem.

Korisnicima nije teško da većinu apstrakcija identifikuju zbog toga što se njima tipično označavaju stvari, na način koji korisnici već koriste za opis svog sistema. Za one koji rešavaju problem, ove apstrakcije su samo stvari u tehnologiji koje su deo rešenja.

Sledeći koraci u definisanju rečnika sistema su:

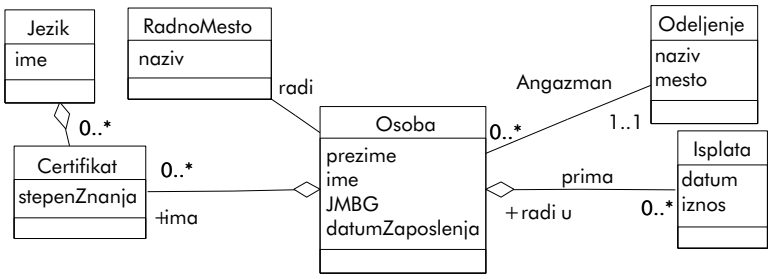
- identifikovati one stvari koje se koriste za opisivanje problema ili rešenja i
- za svaku apstrakciju identifikovati skup odgovornosti.

Kako je koncept opis stvari u realnom sistemu, iskaz se odnosi na atribute implementirane u kontekstu entiteta relanog sistema, odgovarajuće operacije i definisane odgovornosti.

## Definisanje atributa

Atribut je logička vrednost podataka za objekt. Atributi se mogu definisati u okviru slučajeva upotrebe sugerišući ili dajući nagoveštaj o potrebi za informacijom.

Atribute na konceptualnom nivou treba predstaviti jednostavnim tipovima vrednosti (Boolen, Date, Number, String, Time), a *ne treba predstavljati prenesene ključeve*.




**Slika 2.13.**
*Konceptualni model*

Na slici 2.13. su prikazani atributi kao osnova za formiranje rečnika sistema za dijagram koncepta prikazan na slici 2.12.

## Definisanje elementarnih operacija

Elementarne operacije koncepta nastale su na osnovu definisanog sistemskog dijagrama sekvenci i treba da skiciraju mogućnost budućeg menjanja stanja posmatranog objekta ili preuzimaju vrednosti atributa posmatranog koncepta. Svakom konceptu se pristupa preko skupa operacija gde *javni deo* koncepta predstavlja njegov id i operacije koje čine njegov protokol. S druge strane, *privatni deo* se odnosi na nivoe buduće implementacije odgovarajućih operacija, tj. uvodi se princip skrivanja informacija.

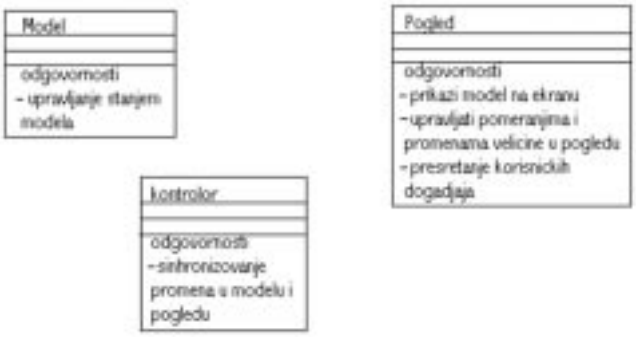
Imajući ove aspekte u vidu, u okviru objektno orijentisane analize definišu se elementarne operacije (koje će se kasnije detaljno razmatrati) i usvajaju u okviru objektno orijentisanog dizajna. Kako je u pitanju inkrementalno iterativan pristup, to će se prva korekcija elementarnih operacija izvesti u okviru definisanja dijagrama sekvenci, a kasnije i u okviru dijagrama saradnje sa završnim elementima u okviru definisanja potpunog dijagrama klasa.

## Definisanje odgovornosti

Posle definisanja elementarnih operacija, mogući korak koji je u sprezi sa definisanjem ugovora (odgovornost prethodi pri uslovima kojima se definiše stanje pre početka operacije) jeste definisanje odgovornosti (Responsobilities). O definisanju ugovora biće više reči u sledećem poglavlju.

*Odgovornost* je ugovor ili obaveza koncepta. Kada se kreira koncept, pravi se iskaz u kojem svi objekti imaju istu vrstu stanja i isto se ponašaju. Na višem nivou apstrakcije korespodentni atributi i operacije su samo karakteristike kojima se "nose" odgovornosti koncepta. Odgovornosti su jednostavno slobodne tekstualne forme. U praksi, jedna odgovornost je pisana kao fraza, rečenica, ili (u većini slučajeva) kao kratki paragraf.

Odgovornosti se mogu prikazati u posebnim odeljcima na dnu slike klase, kao na slici 2.14.




**Slika 2.14.**
*Odgovornosti*

Da bi se modelirala podela odgovornosti u sistemu, treba da se:

- identifikuje skup koncepata koji su u bliskoj saradnji, da bi izneli neko ponašanje;
- identifikuje skup odgovornosti za svaki od ovih koncepata;
- gleda na skup koncepata kao na celinu, a koncepti koji imaju previše odgovornosti podele na manje apstrakcije;
- razmotri način na koji koncepti sarađuju jedan sa drugim i ponovo dele svoje odgovornosti tako da nijedan koncept unutar saradnje ne radi ni previše ni premalo.

Atributi, operacije i odgovornosti su najčešće karakteristike koje će biti potrebne kada se budu stvarale apstrakcije. U stvari, za većinu modela glavna forma su ove tri karakteristike. Međutim, ponekad će biti potrebno specificiranje ili vizuelizacija nekih drugih karakteristika, na primer, vidljivost individualnih atributa i operacija; određene jezičke karakteristike operacija, kao što su polimorfizam ili konstantnost; ili čak izuzeci koje objekti klase mogu da stvore.

Ove i mnoge druge osobine mogu biti ispoljene u UML-u, iako su tretirane kao viši koncepti. Kod definisanja modela, skoro svaka apstrakcija je jedna vrsta koncepta.

Ukoliko modeli postaju složeniji, mnogi koncepti se grupišu u zajedničke grupe, tj. pakete. Kada se modelira u UML-u treba napraviti konceptualnu apstrakciju, takvu da:

- obezbeđuje apstrakciju nečega iz rečnika problema ili rešenja;
- oličava male, dobro definisane skupove odgovornosti i do kraja ih dobro vodi;
- obezbeđuje jasnu razdvojenost apstraktnih specifičnosti i njihove implementacije;
- bude razumljiva i jednostavna, a opet izražajna i prilagodljiva.

Kada se upisuju koncepti u UML, potrebno je:

- definisati samo one karakteristike koje su važne za razumevanje apstrakcije u svom kontekstu;
- definisati dugačke liste atributa i grupisati ih prema njihovim kategorijama;
- grupisati srodne koncepte.

Na osnovu definisanog dijagrama koncepta, u sledećem koraku pristupiće se definisanju dijagrama sekvenci.

## Izrada dijagrama sekvenci (Sequence Diagrams)

Dijagram sekvenci se koristi za specifikaciju vremenskih zahteva u opisu složenih scenarija, tj. za opis toka poruka između objekata kojima se realizuje odgovarajuća operacija u sistemu. Na dijagramu sekvenci se ne prikazuju veze između objekata, već daje odgovor na pitanje šta sistem radi, tj. identifikuju se sistemski događaji i sistemske operacije za odgovarajući slučaj upotrebe.

Dijagram sekvenci je jedan od dijagrama interakcije. Interakcija je ponašanje koje obuhvata skup poruka koje se razmenjuju između skupa objekata u nekom kontekstu sa nekom namenom. Poruka je specifikacija komunikacije između objekata koja prenosi informaciju. Od poruke se očekuje da će uslediti aktivnost. Prijem jedne poruke se može smatrati instancom jednog događaja. Kada se pošalje poruka, sledi akcija - izvršenje naredbe koja predstavlja apstrakciju metode. Na osnovu poslatih poruka UML predviđa sledeće vrste akcija:

| Akcija | Opis |
|---|---|
| Poziv (call) | Pokreće operaciju objekta primaoc (može biti i poziva sebi) |
| povratak (eturn) | Vraća vrednost pozivaocu |
| slanje (send) operacija | Asinhrono se šalje signal primaocu |
| kreiranje (create) <<create>> | Kreira se objekt |
| uništavanje (destory) <<destory>> | Uništava se objekt (objekt može biti i suicidan) |
| postajanje (become) <<become>> | Objekt menja prirodu (na obe strane veze je isti objekat) |

**Slika 2.15.**
*Vrste akcija na osnovu poslatih poruka*

Interakcija se koristi za modeliranje dinamičkih aspekata modela definisanih preko dijagrama interakcije koji čine dijagram sekvenci i dijagram saradnje (kolaboracije). Dijagrami sekvence naglašavaju vremensko uređenje interakcije, a dijagrami saradnje naglašavaju strukturu veza između učesnika u interakciji. Dijagram sekvenci je povoljniji za prikaz kompleksnih slučajeva upotrebe (scenarija) u interaktivnim sistemima i sistemima koji rade u realnom vremenu, dok su *dijagrami saradnje* povoljniji za opis složenijih algoritama.

Ove dve vrste dijagrama vizuelizuju na različit način iste informacije. Semantički su potpuno ekvivalentni i CASE alati Rational Rous i Pardigma Plus, i automatski konvertuju jedan u drugi.

U okviru izrade dijagrama sekvenci definišu se sledeće aktivnosti:

- definisanje objekata;
- definisanje poruka;
- definisanje indikatora sinhronizacije.

## Definisanje objekata

Ako su dijagrami slučajeva upotrebe prethodno definisani, onda je dijagram sekvenci jedna od realizacija slučajeva upotrebe, koja pokazuje redosled događaja koje generišu spoljni učesnici za *svaki slučaj upotrebe*. Slučaj upotrebe sugeriše način na koji se učesnik nalazi u interakciji sa softverskim sistemom, tj. učesnik generiše događaje. Imajući to u vidu, osnovna podela dinamike u dijagramu sekvenci je definisana kao:

- Događaj tj. spoljašnji ulazni događaj koji generiše učesnik u sistemu i
- Operacija (Trigger), koja je odziv na događaj u sistemu.

Analogija događaja i operacije, slično odnosu poruke (message) i metode

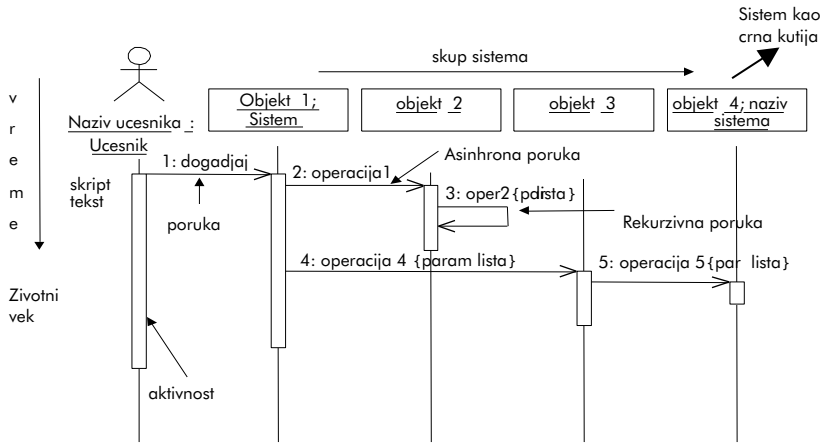
Treba naglasiti da je dijagram sekvenci prikaz dinamičke saradnje između objekata u vremenu, tj. da je to grafička ilustracija dinamičke interakcije, gde objekti komuniciraju preko sekvenci poruka. Takođe, dijagram sekvenci se koristi za prikaz korisničkog interfejsa, jer se definiše sekvenca događaja koje korisnik (interfejs, okolina) prosleđuje sistemu u jednom slučaju upotrebe.

Kao što je rečeno, dijagram sekvenci pokazuje interakciju između objekata u vremenskoj sekvenci. Oni poseduju dve dimenzije: vreme i kolekciju objekata. Uobičajeno je da se vreme prikazuje po vertikalnoj, a kolekcija objekata po horizontalnoj dimenziji (slika 2.16). Na vertikalnoj liniji se može, na pogodan način, predstaviti i vremenska skala. Isto tako je uobičajeno da se na dijagramu sekvenci prikazuju i granice sistema. Oni predstavljaju, obično, za jedan slučaj upotrebe, objekte koji su u interakciji i sekvencu događaja (poruka) koje oni razmenjuju. Uz levu ivicu dijagrama sekvenci moguće je opisati sekvencu operacija nad prikazanim objektima, koristeći strukturni prirodni jezik ili neki formalniji način.

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

Kao što slika 2.16. pokazuje, dijagram sekvenci se formira tako što se prvo na vrh dijagrama, duž ose X, postave objekti koji učestvuju u interakciji. Obično se objekti koji započinju interakciju stavljaju levo, a objekti koji slede redom nadesno. Posle toga se poruke koje ovi objekti šalju i primaju smeštaju duž ose Y, u rastućem nizu po vremenskom redosledu od vrha ka dnu. Ovo pruža jasnu asocijaciju na tok kontrole u proteku vremena.

Na slici 2.16. je prikazana notacija dijagrama sekvenci.




**Slika 2.16.**
*Notacija dijagrama sekvenci*

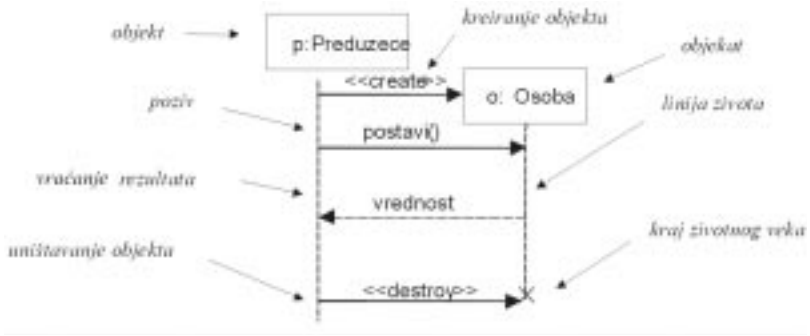
Dijagram sekvenci iz dijagrama slučajeva upotrebe preuzima učesnike, a ako se prvo definiše dijagram koncepta, preuzimaju se objekti. U okviru dijagrama sekvenci učesnici mogu biti korisnik ili veštački entitet (softver ili sistem). Objekt je instanca koncepta koja otprema i prima poruke. Moguće je da objekt otpremi poruku samom sebi. Objekti se definišu nazivom objekta i nazivom koncepta, koji su podvučeni i to na sledeći način:

Naziv objekta: Naziv koncepta

Kao što se može videti na slici 2.17. objekti se prikazuju u pravougaonicima i ime objekta se obavezno unosi.

Iz svakog objekta polazi nadole isprekidana linija koja predstavlja njegov životni vek (lifeline). Životni vek predstavlja postojanje nekog objekta u periodu vremena. Većina objekata koji se pojavljuju u dijagramu sekvenci postojaće i dok traje interakcija, pa su svi ti objekti poređani na vrhu dijagrama, sa svojim životnim vekom povučenim od vrha ka dnu dijagrama. Objekti se mogu praviti u toku interakcije. Njihov životni vek počinje nakon prijema poruke `create` (slika 2.17). Objekti mogu biti uništeni u toku interakcije. Njihov životni vek se završava po prijemu poruke `destroy` (još se naglašava velikim znakom X, kao krajem njihovog života).

Na slici 2.17. je dat primer dijagrama sekvenci i raznih vrsta poruka za objekte o:Osoba i p:Preduzece. Prvo se akcijom `<<create>>` kreira objekt o za koncept Osoba. Operacijom postavi() postavlja se osoba na neko radno mesto. Akcijom vrednost vraća se rezultat u objekt p koncepta Preduzece. Uništavanje objekta o koncepta Osoba izvodi se akcijom `<<destry>>`.



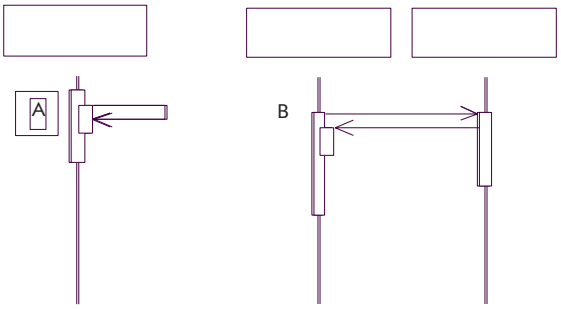

**Slika 2.17.**
*Primer dijagrama sekvenci*

## Definisanje poruka

Poruke (događaji) preko kojih objekti komuniciraju, prikazuju se usmerenim horizontalnim linijama. Ove linije, u osnovi, predstavljaju tranzicije, pa je moguć isti način označavanja kao i za dijagrame promene stanja: argumenti se navode običnim, a eventualni uslovi tranzicije (okidanja aktivnosti na drugom objeku) u uglastim zagradama.

Kao što je rečeno, postojanje aktivnosti koje objekt izvodi u vremenu predstavlja se tankim pravougaonikom (slika 2.18) koji prekriva vremensku osu u vertikalnom pravcu, i u veličini koja odgovara vremenu njegovog postojanja.

Dijagram sekvenci može da ima i rekurzivne poruke. To su poruke koje objekt upućuje samom sebi. Rekurzija je prikazana strelicom koja polazi i završava se u objektu. Drugim rečima, može se prikazati ugneždavanje (izazvano rekurzijom, pozivom same sebe) zbog povratnog poziva (call back) od pozivnog objekta.




**Slika 2.18.**
*Prikaz aktivnosti*

Dakle, objekti međusobno komuniciraju slanjem *poruka*. Poruke se predstavljaju strelicama koje polaze od jednog objekta, a završavaju na drugom. Poruka (Message) definisana nazivom i parametrima. Parametrima se definiše jedan ili više argumenata. Poruka je opisana na sledeći način:

*Naziv poruke* (lista parametara)

Uz naziv poruke može se navesti i broj kojim se definiše redosled izvođenja poruka. Lista parametara je opciona i male zagrade se obavezno navode bez obzira na listu parametara.

## Definisanje indikatora sinhronizacije

Indikatorom sinhronizacije definiše se tip poruke. Na slici 2.19. su prikazani različiti oblici indikatora sinhronizacije.

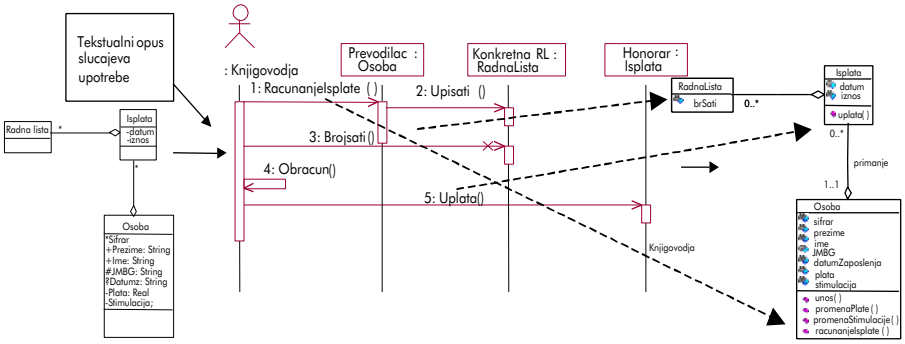
| Oznaka | Opis |
|---|---|
| Obična (Simple) | Kod običnih poruka nije bitan tip poruke te se i ne specificira. |
| Sinhrone poruke (Synchronous) | Sinhrone poruke zahtevaju odgovor, koji se šalje sa odredišta po prijemu i obradi, blokiraju izvršenje dok odgovor ne stigne. |
| Asinhrona poruka (Asynchronous) | Asinhrone poruke ne blokiraju izvršenje kada se šalju i koriste se kod konkurentnog programiranja |
| Vremenska poruka (Timeout) | Ovim tipom veze definiše se vremensko čekanje na izlaz. |
| Prepreka(Balking) | Ovim tipom poruke se definiše prepreka ili greška. |

**Slika 2.19.**
*Prikaz različitih indikatora sinhronizacije*

## Primer definisanja dijagrama sekvenci

Na slici 2.20. se pokazuje način dolaženja do dijagrama sekvenci, gde se rezultati koriste. Polazi se od tekstualnog opisa slučajeva upotrebe, kao i definisanog dijagrama koncepta. Dijagramom koncepta se definišu koncepti koji su ovde objekti i predstavljaju crne kutije. Tekstulanim opisom slučajeva upotrebe definišu se događaji i operacije koje kao izlaz definišu operacije za objekt Osoba.

Dakle, dijagram sekvenci može da posluži da se operacije pridruže klasama objekata u dijagramu klasa, kako je to na slici 2.20. prikazano. Svi objekti moraju da poseduju operacije kojima se "odazivaju" na ulazne tranzicije (događaje, odnosno poruke).




**Slika 2.20.**
*Prikaz dolaženja do dijagrama sekvenci*

Ako posmatramo postupak dolaženja do dijagrama sekvenci prikazan na slici 2.20., vidi se da dijagram koncepta prikazuje veze između učesnika Knjigovođa i klasa Osoba, Isplata i Radna lista. Na dijagramu sekvenci je opisan niz operacija radi evidentiranja isplata radnika.

Redosled poruka je sledeći:

- Prvo, Knjigovođa zahteva od objekta Radnik klase Osoba da izvrši računanje isplate korišćenjem poruke (operacije) *racunanje Isplata ()*;
- Druga poruka *upisati()* je deo tekuće operacije računanje isplate;
- Nakon toga ide treća poruka *brojsati()*, kojom Knjigovođa očitava broj sati sa radne liste;
- Četvrtom porukom *Obracun()* izvodi se obračun isplata;
- Na kraju, Knjigovođa šalje petu poruku *Uplata()*, kojom poziva istoimenu operaciju klase Isplata.

# Definisanje ugovora o izvršenju operacije

Definisanjem dijagrama koncepta opisani su osnovni koncepti i atributi, a dijagramom sekvenci buduće operacije za buduće klase. Potrebno je definisati i odgovarajuće ugovore o izvršenju operacija. Dijagramom sekvenci se opisuju događaji koje spoljašnji učesnik generiše, ali se ne razrađuju detalji vezani za funkcionalnost asocirane pozvane operacije. Ugovorom se opisuje šta operacija postiže izvršenjem, tj. deklariše se šta će se dogoditi i kako se to postiže. Dakle, ugovor o izvršenju operacije opisuje ponašanja sistema, tj. izmenu stanja sistema.

Ugovor o izvršenju operacije je pomoć u definisanju ponašanja sistema, tj. njime se opisuju efekti operacije na sistem. U UML-u su sadržani elementi potrebni za definisanje ugovora o izvršenju operacije. Pri tome se definišu prethodni i postuslovi za operaciju. U UML formalnim definicijama ili metamodelima operacije postoji set predefinisanih osobina kojima se uključuju prethodni i postuslovi.

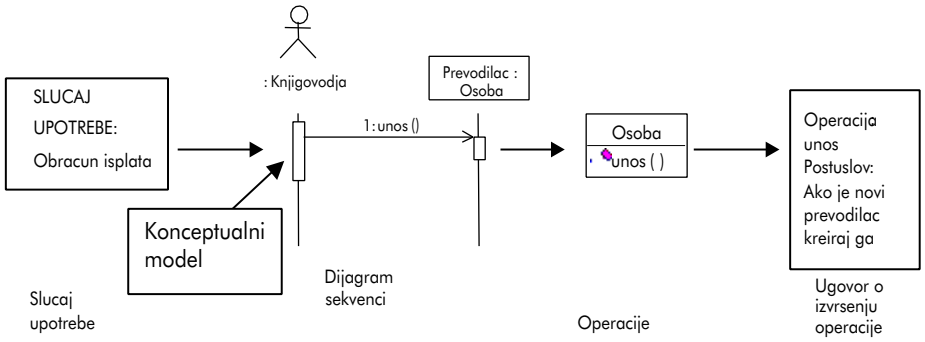
Redosled koraka za definisanje ugovora o izvršenju operacija je sledeći:

- identifikacija operacija u dijagramu sekvenci, za koje se definiše ugovor o izvršenju operacije;
- izrada ugovora o izvršenju operacije za izabranu operaciju;
- opisivanje svrhe izvođenja operacije;
- kompletiranje postuslovne sekcije deklarativnim opisom izmena stanja koje će se dogoditi na objektu u konceptualnom modelu.

  Za opisivanje postuslova koriste se sledeće kategorije:
- instance kreiranja i brisanja,
- atributi modifikacije i
- asocijacije, formirane i prekinute.

Redoseled dolaženja do ugovora o izvršenju operacije je sledeći: slučaj upotrebe sugeriše događaje, a dijagram sekvenci identifikuje operacije koje se u ugovorima o izvršenju operacija opisuju.

Dakle, imajući u vidu prethodno rečeno, prikaz dat na slici 2.21, slučaj upotrebe daje odgovor na pitanje koji je domen procesa. Konceptualnim modelom definiše se koncept, tj. opisuje se realni svet. Dijagramom sekvenci definišu se događaji i operacije, a definisanjem ugovora definiše se šta operacija radi.




**Slika 2.21.**
*Prikaz dolaženja do ugovora*