SI Vježbe 5

Domain Model

1. Šta je to domain model? 1
2. Identifikacija konceptualnih klasa 2
3. Korištenje liste kategorija konceptualnih klasa 3
4. Identifikacija imenica 3
5. Kako kreirati domain model 4
6. Veze (Associations) 5
7. Koje veze prikazati? 6
8. Atributi 7

1. Šta je to domain model?

Nakon izrade "use-case" modela često se kreira "domain model", tj model objekata u sferi problema koji se rješava. Domain model se koristi kao izvor inspiracije za dizajniranje softverskih objekata u kasnijim iteracijama, i biće važan "ulaz" za izradu drugih dokumenata i modela u nastavku razvojnog procesa. Kada je riječ o UP razvojnom procesu, obično se izrađuje u fazi elaboracije ali se nadopunjuje (ako je to potrebno) i u fazi konstrukcije.

Domain model ilustruje smislene konceptualne klase iz domene problema. Domain model je najvažniji produkt objektno-orijentisane analize (use-case model je veoma bitan, ali ne spada u produkte objektno orijentisane analize).

Domain model je reprezentacija konceptualnih klasa iz stvarnog svijeta. Bitno je naglasiti da domain model **ne predstavlja** softverske komponente. To nije skup dijagrama koji opisuju softverske klase ili softverske objekte.

U UML notaciji domain model se ilustruje skupom dijagrama klasa na kojim se ne prikazuju operacije. Domain model može uključivati:

- objekte iz domene problema (ili konceptualne klase)
- veze između konceptualnih klasa
- atribute konceptualnih klasa

Najjednostavnije rečeno domain model je vizuelni riječnik projekta. Domain model prikazuje *apstrakcije* ključnih koncepata. To znači da model prikazuje samo djelomičan pogled (ili apstrakciju) bitnih elemenata (bitnih za projekat), a zanemaruje nebitne detalje. Sve informacije sa domain modela bi se mogle prikazati i u tekstualnom obliku, ali je vizualna prezentacija preglednija i lakša za razumijevanje.

Kao što je već rečeno domain model je vizualizacija stvari iz realnog svijeta, a ne softverskih komponenti (C#, C++ ili Java klasa). Zato slijedeće stvari nisu prikladne za domain model:

- Softverski pojmovi: window, database
- Zaduženja (metode ili operacije)

Neformalno, konceptualna klasa je ideja, stvar ili objekat.

Softverski problemi mogu biti kompleksni. Uobičajena strategija rješavanja kompleksnih problema je "divide-and-conquer" (doslovan prevod je "podijeli pa osvoji"). Primjenom te strategije problem se dijeli na manje dijelove koje je lakše razumjeti i njima ovladati. U strukturnoj analizi dekompozicija se radi po procesima ili po funkcijama. Nasuprot tome, u objektno orijentisanoj analizi dekompozicija se izvršava na osnovu entiteta ili "stvari" iz domene problema.

Na primjer, ako razmatramo sistem za studentske službe, koncepti iz stvarnog svijeta (ili konceptualne klase) bili bi slijedeći:

- student
- profesor
- predmet
- prijava
- ispit
- itd.

2. Identifikacija konceptualnih klasa

U iterativnom razvojnom procesu domain model se inkrementalno izgrađuje kroz nekoliko iteracija u fazi elaboracije. U svakoj iteraciji domain model uključuje samo koncepte identifikovane za "use-case"-ove koji se razmatraju u toj iteraciji (i koncepte iz ranijih iteracija). Centralni zadatak je, znači, identifikacija konceptualnih klasa vezano za scenario koji se trenutno analizira.

Slijedi koristan savjet pri identifikaciji konceptualnih klasa.

Uvijek je bolje detaljnije specifikovati više konceptualnih klasa od onog što je možda stvarno neophodno nego kreirati nedovoljno detaljan domain model.

Pogrešno je misliti da je domain model bolji ako ima manje klasa. Obično je istina suprotna tome. Nemojte isključivati neki koncept samo zato što zahtjevi (use-case) direktno ne upućuju da bi se trebale pamtiti informacije o njemu, ili zato što konceptualna klasa nema atributa (po tome se domain model razlikuje od dizajna relacione baze podataka gdje su upravo to kriteriji za eliminaciju koncepata).

Obično se ne otkriju svi koncepti u prvom koraku, nego se identifikuju kasnije pri identifikaciji atributa ili veza.

Za identifikaciju konceptualnih klasa mogu se koristiti dvije tehnike:

- Korištenje liste kategorija konceptualnih klasa
- Identifikacija imenica

Pored ove dvije tehnike moguće je koristiti "analysis patterns". "Analysis patterns" su postojeći djelomični domain modeli koji su kreirani od strane eksperata. Mogu se naći u slijedećim publikacijama:

- Fowler, M. 1986. Analysis Patterns: Reusable Object Models, Addison-Wesley
- Hay, D. 1996. Data Model Pattens: Conventions of Thought, Dorset House

3. Korištenje liste kategorija konceptualnih klasa

Kreiranje domain modela počnite formiranjem liste kandidatskih konceptualnih klasa. Pri tome, korisnom se može pokazati slijedeća lista kategorija konceptualnih klasa:

Kategorija koncept. klasa	Primjeri
fizičke ili opipljive stvari	Indeks, Prijava
opisi ili specifikacije stvari	
mjesta	
transakcije	Prijava ispita
uloge osoba	Referent, Nastavno osoblje, Student
stvari koji sadrže druge objekte ("containers")	NastavniPlan (sadrži predmete)
objekti unutar drugih objekata	Predmet (dio nastavnog plana)
drugi računari ili vanjski sistemi	
apstraktne imenice	
organizacije, dijelovi organizacije	Fakultet, Univerzitet, Odsjek, StudijskaGrupa
događaji	Ispit
pravila	UslovZaIspit, UslovZaUpis
katalozi	MaticnaKnjiga (kao katalog studenata)
zapisi o nekom događaju	IzvjestajSaIspita
finansijski instrumenti i usluge	IzvodIzBanke, Uplatnica
dokumenti, knjige...	Potvrda

4. Identifikacija imenica

Druga korisna tehnika za pronalazak je jezička analiza – identifikacija imenica iz tekstualnog opisa problema (tj. formalni "use-case" opisi). Međutim, pri korištenju ove tehnike potrebno je imati na umu da je ovakvo "mehaničko" mapiranje imenica u klase nije moguće, a riječi u prirodnom jeziku mogu biti dvosmislene. Ipak, i ova tehnika je koristan izvor inspiracije za pronalazak konceptualnih klasa.

Korištenjem ove tehnike često se dobija veći broj potencijalnih klasa od kojih će neke biti ignorisane u ovoj iteraciji, a neke predstavljaju atribute drugih klasa.

5. Kako kreirati domain model

Pri izradi domain modela mogu se primijeniti slijedeći koraci:

1. Napraviti listu kandidatskih konceptualnih klasa korištenjem kategorija ili identifikacijom imenica iz "use-case" opisa.
2. Ucrtati klase u domain model
3. Dodati veze koje koje je neophodno "pamtiti" u memoriji
4. Dodati atribute koji su neophodni za ispunjavanje zahtjeva iz "use-case" opisa

Domain model se kreira na način sličan kao što kartograf iscrtava karte:

- Koriste se samo postojeća imena na datom teritoriju (tj. iz domene problema)
- Isključuju se nerelevantni detalji (uključuju se samo relevantni detalji koji se obrađuju u trenutnoj iteraciji)
- Ne dodaju se objekti koji ne postoje u stvarnom svijetu

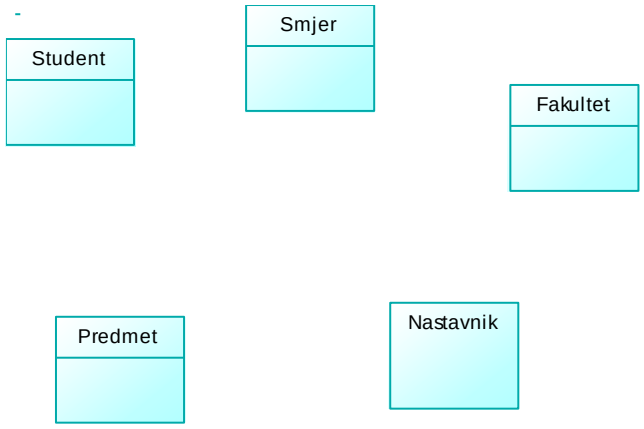
Slijedeći savjet je, također, jako bitan:

Izbjegavajte način razmišljanja koji je karakterističan za "waterflow" razvojni proces. Nemojte pokušavati napraviti detaljan ili "pravilan" model. Domain model nikad neće biti savršen. Takvo razmišljanje vodi stanju koje je poznato kao "analysis paralysis" (paraliza u analizi).

Najčešća pogreška pri kreiranju domain modela je kada se nešto predstavi kao atribut, a trebala je biti konceptualna klasa. Primjenom slijedećeg pravila može se izbjeći takva greška:

Ako o nekom objektu X ne razmišljamo kao o običnom broju ili tekstu onda je to najvjerovatnije konceptualna klasa X, a ne atribut.

Na UML dijagramima konceptualne klase se predstavljaju pravouganim figurama (dijagrama klasa) kao na slijedećoj slici:



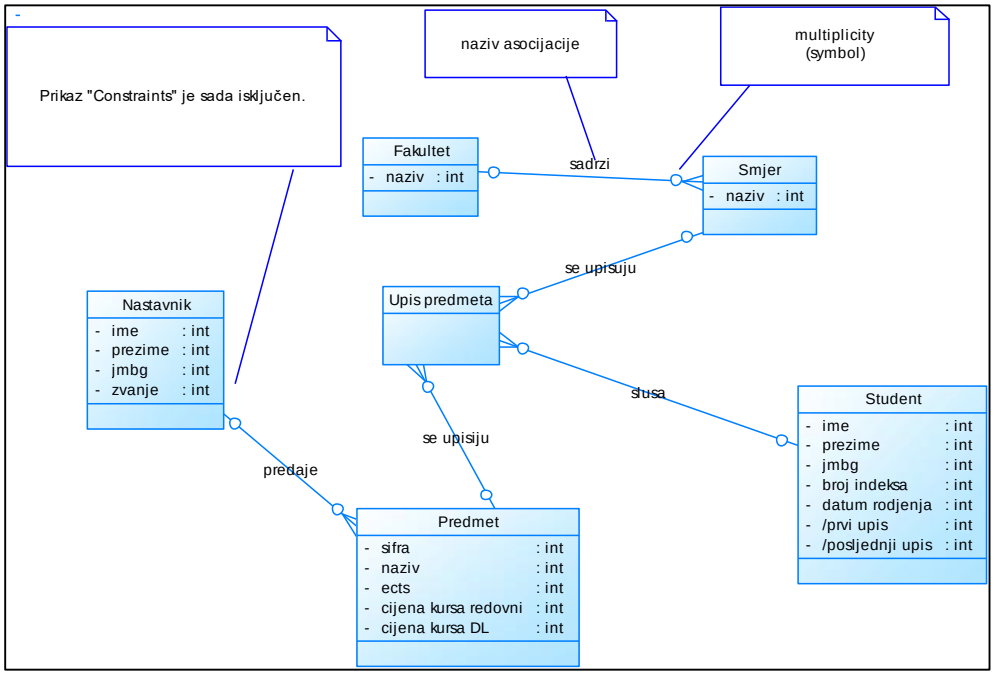
Slika 5.1. Entiteti domain modela

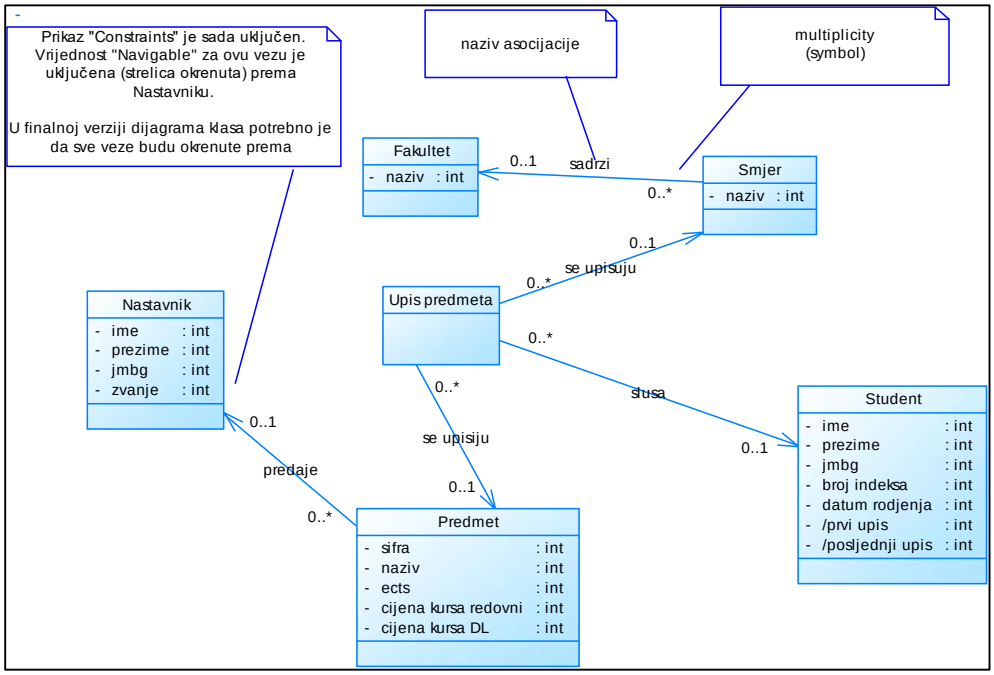
6. Veze (Associations)

Pri izradi domain modela korisno je pronaći veze između konceptualnih klasa koje su neophodne za zadovoljavanje zahtjeva za izvršavanje scenarija ("use-cases") koje trenutno obrađujemo.

"Association" je relacija između konceptualnih klasa (ili preciznije, između instanci tih klasa) koja upućuju na neku bitnu ili interesantnu vezu između tih klasa.

Na UML dijagramima veze se predstavljaju na jedna do dva načina:

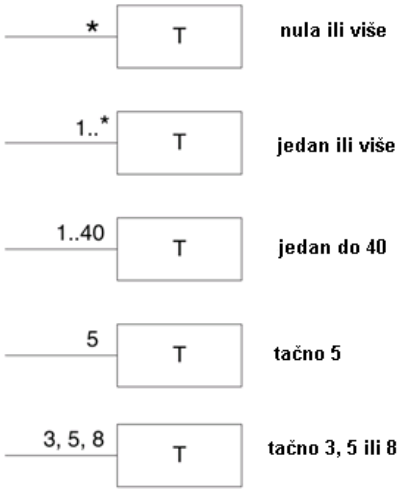




Na krajevima veze mogu biti ispisani "multiplicity" izrazi. Oni označavaju brojevnu vezu između instanci

konceptualnih klasa. Označavaju koliko instanci klase A može biti povezano sa jednom instancom klase B. Na domain modelu relacije su "bidirekcionalne" (dvosmjerne).

Na primjer: Jedan "Fakultet" može sadržati jedan ili više odsjeka "Odsjek" (to je naznačeno sa 1..* na strani klase Odsjek). Sa druge strane, jedan "Odsjek" može pripadati samo jednom "Fakultetu" (pa je to naznačeno sa "1" na strani "Fakulteta"). Slijedeća ilustracija prikazuje moguće "multiplicity" izraze.



Slika 6.1. Multiplicity izrazi

Veze se obično imenuju u stilu: "Klasa-glagol-Klasa" (npr. Fakultet-sadrzi-Odsjek). Ako pored naziva veze nije prikazana strelica podrazumijeva se (mada UML to ne propisuje) čitanje s lijeve strane prema desnoj i odozgo prema dole.

7. Koje veze prikazati?

Na domain modelu se prikazuju one veze koje je potrebno pamtiti određeno vrijeme (bilo da se radi o nekoliko milisekundi ili nekoliko godina, zavisno od konteksta).

U domain model se obično uključuju slijedeće asocijacije:

- Veze koje je potrebno "pamtiti" određeno vrijeme ("need-to-remember").
- Veze iz liste "uobičajenih" asocijacija.

Poželjno je da se dijagram ne "pretrpa" nepotrebnim vezama jer će to stvoriti "vizeulnu buku". Opšta ideja dijagrama je poboljšanje čitljivosti i preglednosti. Ako se unese previše nepotrebnih veza gubi se osnovna prednost vizuelnog predstavljanja domain modela. Potrebno je fokusirati se na "need-to-remember" tipove veza.

U ovoj fazi modeliranja prikazane veze ne predstavljaju obavezu da se implementira veza između softverskih objekata ili relacija u bazi podataka. Domain model nije model baze podataka. Domain model je samo konceptualna predstava objekata iz stvarnog svijeta.

Kao pomoć pri pronalasku veza može poslužiti slijedeća lista uobičajenih tipova veza.

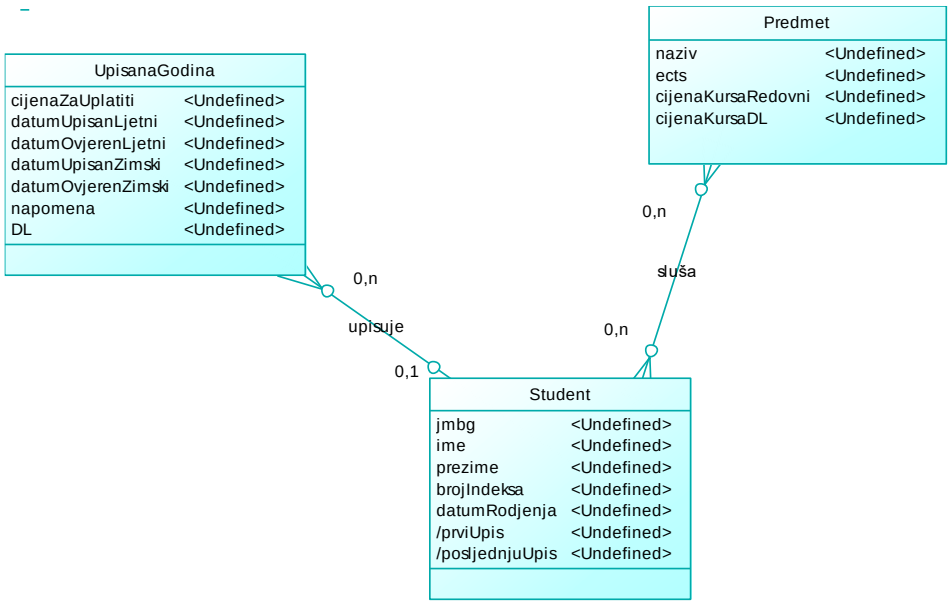
Kategorija veze
A je transakcija povezana sa drugom transakcijom B
A je artikal ili usluga koja pripada transakciji B
A je fizički ili logički dio od B
A se fizički ili logički nalazi unutar B
A je opis ili specifikacija za B
A se zapisuje u B
A je član od B
A je organizacijska jedinica od B
A koristi, upravlja ili posjeduje B

8. Atributi

Pored veza, na domain modelu se prikazuju i atributi koji su neophodni za ispunjavanje zahtjeva scenarija koji se trenutno obrađuju (u pojedinoj iteraciji).

Slično vezama, prikazuju se oni atributi koje je potrebno pamtiti (na osnovu "use-case" opisa gdje se ta potreba implicira ili direktno sugeriše). Na primjer, pri upisu studenta, pamte se njegovi lični podaci (ime, prezime), zatim broj dosijea, datum upisa i slično.

Na Domain modelu se ne prikazuju primarni ključevi koji ne postoje u stvarnom svijetu (npr. ID koji se automatski generiše). Ako je JMBG primarni ključ onda ćemo ga prikazati (jer on postoji u stvarnom svijetu).



Na prethodnoj ilustraciji prikazani su detalji koji se obično prikazuju na domain modelu. Atributi čije ime počinje sa

"/" su "izvedeni" atributi. Njihova vrijednost se izvodi na osnovu podataka vezanih za konceptualnu klasu. Na primjer, "/prviUpis" se izvodi tako što se pronađe instanca klase "Upis" sa najstarijim datumom. Isto tako "/posljednjiUpis" s izvodi tako što se pronađe instanca sa klase "Upis" sa najnovijim datumom.

Izvedeni atributi se prikazuju ako su iz nekog razloga bitni za klasu, a mogu se izvesti na bilo koji način. Time se izbjegava dupliranje informacija na domain modelu.

Na domain modelu obično nisu prikazani nivoi pristupa atributima (private, public, protected...).

Najčešća pogreška je stavljanje atributa koji predstavljaju "strani ključ" za vezu sa nekom drugom konceptualnom klasom. Takvi atributi ne bi trebali biti prikazani na domain modelu, nego prikazani asocijacijama. Domain model nije isto što i model baze podataka.

Ne postoji jedan "tačan" domain model. Svi modeli su aproksimacija problema koji nastojimo razumjeti. Domain model je prvenstveno alat za komunikaciju u okviru grupe.

Postoji ***koristan*** domain model, a takav je model koji opisuje ključne apstrakcije i prikazuje informacije neophodne za razumijevanje problema u kontekstu trenutnih zahtjeva.