1. Domain model

Domain model je najvažniji proizvod objektno-orijentirane analize (use-case model je veoma bitan, ali ne spada u proizvode objektno orijentirane analize).

Proces "objektno orijentisane analize" fokusiran je na pronalaženje i opis objekata ili koncepta iz domene problema. Također, podrazumijeva i specifikaciju najvažnijih atributa objekata i interakcije između objekata.

Ovaj proces se realizira u sljedećim koracima:

- izrada „domain modela“,
- izrada dijagrama sekvenci i
- definisanje ugovora o izvršenju operacije.

1.1. ŠTA JE TO DOMAIN MODEL?

Nakon izrade "use-case" modela često se kreira "domain model", tj. model objekata u domeni problema koji rješavamo. Domain model se kasnije koristi kao izvor inspiracije za dizajn softverskih objekata i bit će važan "ulaz" za izradu drugih dokumenata i modela u nastavku razvojnog procesa. Obično se izrađuje u toku objektno orijentisane analize.

Domain model ilustrira smislene konceptualne klase iz domene problema.

Domain model predstavlja konceptualne klase 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-u se za izradu domain modela koristi notacija dijagrama klasa na kojem se ne prikazuju operacije.

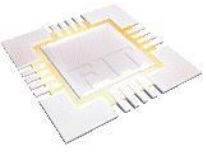
Domain model može sadržavati:

- objekte ili konceptualne klase iz domene problema,
- veze između konceptualnih klasa i
- atribute konceptualnih klasa.

Najjednostavnije rečeno domain model je grafički predstavljen ***riječnik projekta***. Domain model prikazuje *apstrakcije* ključnih koncepata. To znači da model prikazuje samo djelimičan pogled (ili apstrakciju) važnih elemenata za projekt, a zanemaruje nevažne detalje. Sve informacije sa domain modela bi se mogle prikazati i u tekstualnom obliku, ali je grafička prezentacija preglednija i lakše ju je razumjeti.

Kao što je već rečeno, domain model predstavlja pojave iz realnog svijeta, a ne softverske komponente (C#, C++ ili Java klasa). Zato sljedeće stvari nisu prikladni elementi domain modela:

- softverski pojmovi: window, database, itd;
- zaduženja (metode ili operacije).



Neformalno, *konceptualna klasa je ideja, stvar ili objekt.*

Softverski problemi mogu biti kompleksni. Uobičajena strategija rješavanja kompleksnih problema je "divide-and-conquer" (doslovan prijevod je "podijeli pa osvoji"). Primjenom te strategije problem se dijeli na manje dijelove koje je lakše razumjeti i njima ovladati. U strukturalnoj 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 primjer internet prodaje proizvoda, koncepti iz stvarnog svijeta (ili konceptualne klase) bili bi sljedeći:

- klijenti,
- narudžba,
- faktura,
- itd.

1.2. IDENTIFIKACIJA KONCEPTUALNIH KLASA

Domain model se radi na početku sekvencijalnog razvojnog procesa ili se radi inkrementalno u iterativnom razvojnom procesu. Ukoliko se radi o iterativnom razvojnom procesu u svakoj iteraciji domain model uključuje samo koncepte identificirane za "use-case"-ove koji se razmatraju u toj iteraciji (i koncepte iz ranijih iteracija). Centralni zadatak je, znači, identifikacija konceptualnih klasa vezano za scenarij koji se trenutno analizira.

Pogrešno je misliti da je domain model bolji ako ima manje klasa. Obično je istina suprotna tome. Nemojte isključiti 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 ne prepoznamo sve koncepte u prvom koraku, već se identificiraju kasnije pri identifikaciji atributa ili veza.

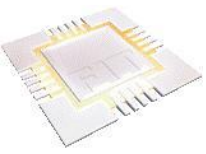
Za identifikaciju konceptualnih klasa mogu se koristiti dvije tehnike:

1. korištenje liste kategorija konceptualnih klasa i/ili
2. identifikacija imenica.

1.3. KORIŠTENJE LISTE KATEGORIJA KONCEPTUALNIH KLASA

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

Kategorija koncept. klasa	Primjeri za sistem studentskih službi
Fizičke ili opipljive stvari	Indeks, Prijava
Opisi ili specifikacije stvari	Broj indeksa
Mjesta	Studentska služba, dekanat
Transakcije	Prijava ispita
Uloge osoba	Referent studentske službe, Nastavno osoblje, Student
Stvari koji sadrže druge objekte ("containers")	Nastavni plan (sadrži predmete)



Objekti unutar drugih objekata	Predmet (dio nastavnog plana)
Drugi računari ili eksterni sistemi	
Apstraktne imenice	
Organizacije, dijelovi organizacije	Fakultet, Univerzitet, Odsjek, Studijska grupa
Događaji	Ispit
Pravila	Uslov za ispit, Uslov za upis
Katalozi	Matična knjiga (kao katalog studenata)
Zapisi o nekom događaju	Izvještaj sa ispita
Finansijski instrumenti i usluge	Izvod iz banke, Uplatnica
Dokumenti, knjige...	Potvrda

1.4. IDENTIFIKACIJA IMENICA

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

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

1.5. KAKO KREIRATI DOMAIN MODEL?

Pri izradi domain modela mogu se primijeniti sljedeći 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 je neophodno "pamtiti" u memoriji;
4. Dodati atribute koji su neophodni za ispunjavanje zahtjeva iz "use-case" dokumenata.

Sljedeći savjet je također jako bitan:

Izbjegavajte način razmišljanja koji je karakterističan za sekvencijalni 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 jedna konceptualna klasa predstavi kao atribut. Primjenom sljedećeg pravila može se izbjeći takva greška:

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

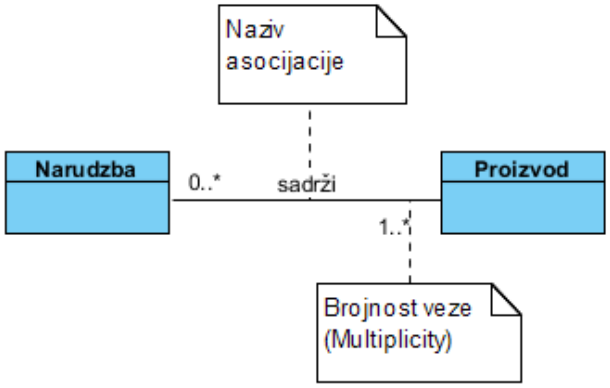
Na UML dijagramima konceptualne klase se predstavljaju pravouglim figurama kao na slici ispod:

2. 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.

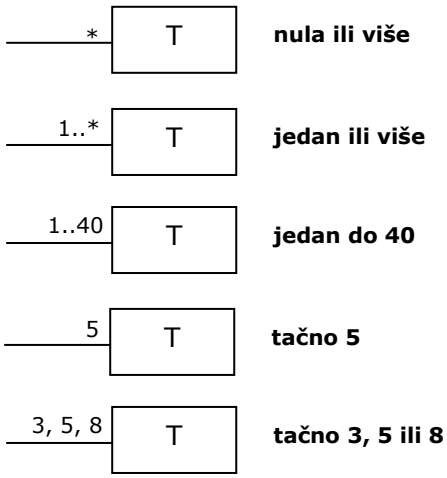
"Association" je relacija između konceptualnih klasa (ili preciznije, između instanci tih klasa) koja upućuje na neku bitnu vezu između tih klasa.

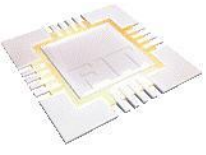
Na UML dijagramima veze se predstavljaju na sljedeći način:



Na krajevima veze mogu biti ispisani "multiplicity" izrazi. Oni predstavljaju brojnost veze (kardinalitet) između instanci konceptualnih klasa. Označavaju koliko instanci klase A može biti povezano sa jednom instancom klase B.

Naredna slika prikazuje moguće "multiplicity" izraze.





Za imenovanje veza obično se koriste glagolski oblici ("Klasa-glagol-Klasa"). Ako pored naziva veze nije prikazana strelica podrazumijeva se (mada UML to ne propisuje) čitanje s lijeve strane prema desnoj i odozgo prema dole.

U domain modelu relacije su dvosmjerne.

Svaka konceptualna klasa ima ulogu u vezi.

2.1. KOJE VEZE PRIKAZATI?

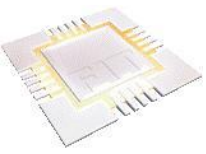
Na domain modelu se prikazuju one veze koje je potrebno pamtiti određeno vrijeme i uobičajene asocijacije (bilo da se radi o nekoliko milisekundi ili nekoliko godina, zavisno od konteksta).

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

U ovoj fazi modeliranja prikazane veze nisu obavezne za implementaciju 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 za identifikaciju veza može poslužiti sljedeća lista uobičajenih tipova veza.

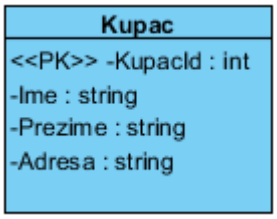
Kategorija veze
A je transakcija povezana sa drugom transakcijom B
A je artikl ili usluga koja pripada transakciji B
A je fizički ili logički dio od B
A se fizički ili logički nalazi unutar B
A je opis ili specifikacija za B
A se zapisuje u B
A je član od B
A je organizacijska jedinica od B
A koristi,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



3. Atributi

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

Prikazuju se oni atributi koje je potrebno pamtiti (na osnovu "use-case" opisa gdje je ta potreba implicitna ili direktno sugerisana). Na primjer, pri evidentiranju kupca pamti se njegovo ime i prezime, adresa, itd.



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 vrijednosti drugih atributa konceptualne klase. Na primjer, "/prvaNarudzba" se izvodi tako što se pronađe instanca klase "Narudzba" sa najstarijim datumom.

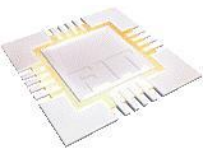
Izvedeni atributi se prikazuju ako su bitni za klasu, a mogu se izvesti na bilo koji način. Time se izbjegava redundansa informacija u domain modelu.

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

Najčešća pogreška je stavljanje atributa koji predstavljaju "eksterni 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.



4. Zadaci

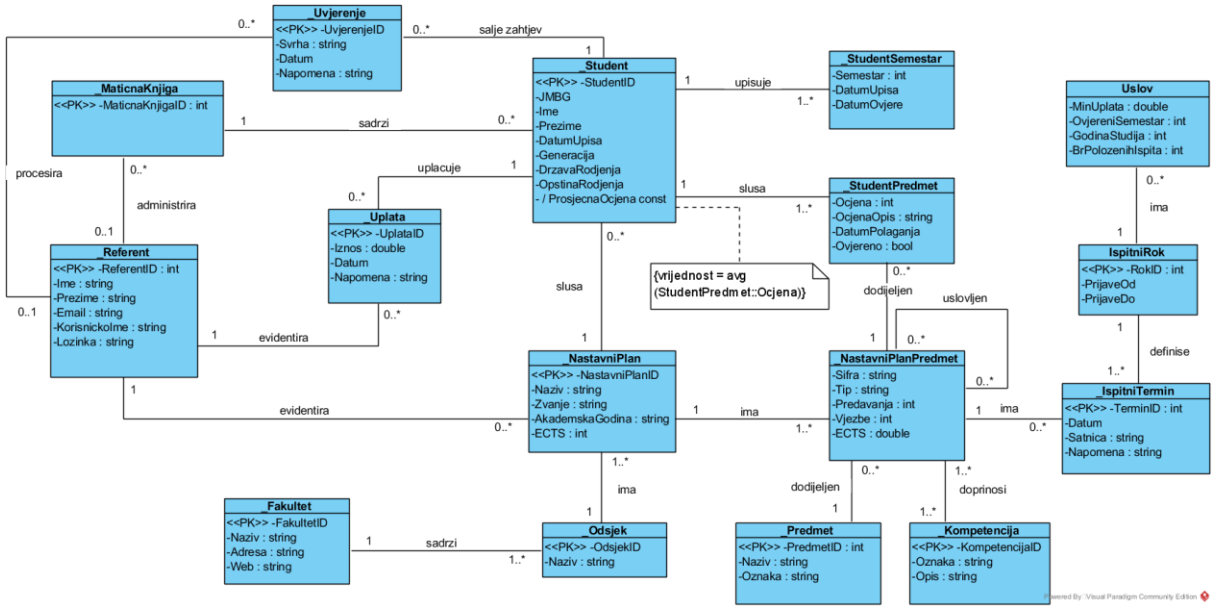
U nastavku je dat prijedlog domain modela za ranije predstavljeni modul *eSluzba*. U ovom slučaju je fokus na drugim elementima opisa zahtjeva, preciznije na onima koji sačinjavaju domain model (klase, atributi, asocijacije).

Zadatak 1: eSluzba

Osnovni korisnici modula eSluzba su referenti studentske službe, koji se za pristup sistemu prvobitno moraju prijaviti sa dodijeljenim korisničkim imenom i lozinkom. Poslovanje studentske službe jeste primarno administracija matične knjige studenata. Međutim, prije detaljnog opisa mogućnosti unutar elektronske matične knjige, potrebno je da budu zadovoljeni određeni preduslovi koji također spadaju u opis posla referenta studentske službe.
Kako bi upis studenata mogao biti u potpunosti podržan, neophodno je da referent evidentira sve predmete nastavnog plana i programa pojedinih odsjeka fakulteta, te da obavi detaljnu evidenciju nastavnog plana koja uključuje unos sljedećih podataka: semestar, naziv, tip i šifru predmeta, te ECTS bodove. Za svaki predmet nastavnog plana se opcionalno može evidentirati uslovljenost, odnosno odabrati lista uslovljenih predmeta. Također, svaki nastavni plan definiše tzv. matricu kompetencija gdje se svakom predmetu pridružuje jedna ili više kompetencija koju studenti stiču po završetku studija.
Po završetku unosa podataka o nastavnom planu, referent može da obavlja upis studenata kroz unos ličnih i kontakt podataka, podataka o prethodnom obrazovanju studenta, statusnih podataka upisanog studija, te opcionalno i podataka o uplatama. Sistem po završetku upisa studentu automatski dodjeljuje predmete odabranog nastavnog plana na upisanoj godini studija, te vrši automatski upis semestra i dodjelu pristupnih podataka. Administracija matične knjige se naknadno vrši putem pretrage po većem broju kriterija (odsjek, smjer, nastavni plan, broj indeksa, ime i prezime studenta), uz mogućnost pregleda i izmjene podataka o studentu, upisa i ovjere semestara, upisa i ovjere ocjena, te evidencije uplata. Svakim upisom ocjene, sistem studentu šalje obavještenje putem e-mail-a.

Izradom domain modela prepoznati konceptualne klase, njihove atribute i međusobne veze opisanog dijela modula *eSluzba*.

Prijedlog rješenja:



Zadatak 2 (za vježbu): eNastava

Osnovni korisnici modula softvera eNastava su članovi nastavnog osoblja, odnosno nastavnici, asistenti i eventualno demonstratori. Svako od osoblja dobiva fakultetsku mail adresu i generisanu lozinku za pristup samom softveru. Da bi nastavno osoblje moglo obavljati svoje svakodnevne poslovne zadatke neophodno im je dodijeliti aktuelne predmete na svim nastavnim planovima koji se trenutno izvode. Dakle, za svakog od njih se definiše na kojem predmetu je angažovan, u kojoj akademskoj godini i na kojoj poziciji. Za unos podataka o osoblju, kao i podataka o njihovom angažmanu brine se administrator sistema.

Ključna funkcionalnost modula eNastava je zapravo administracija pojedinačnih predmeta i ona je podijeljena u sljedeće sekcije: nastavni materijali, obavještenja, studenti na predmetu i ispiti.

- U sklopu sekcije „Nastavni materijali", osoblju je omogućeno postavljanje nastavnih sadržaja studentima koji slušaju predmet uključujući mogućnost pretrage i izmjene objavljenog sadržaja. Nastavni materijali, pored osnovnih podataka, mogu da sadrže više priloženih dokumenata. U osnovne podatke o materijalima spadaju: Naziv, Oblast, Tip, Datum objave, Autori, Ključne riječi i Napomena.
- Analogno se postavljaju i obavještenja na predmetima uz dodatne podatke o tekstualnom sadržaju tih obavještenja. Također je podržana mogućnost dodavanja priloga uz obavještenja. Dodavanje priloga uz nastavne materijale je obavezno, dok je ista opcija za obavještenja opcionalna.
- Sekcija „Studenti na predmetu" pruža uvid u studente koji slušaju odabrani predmet u određenoj akademskoj godini. Pored podataka o studentima, tu su podaci i o njihovom uspjehu, odnosno ocjenama (Datum polaganja, Ocjena, Priznata, Ovjerena). Svako iz reda nastavnog osoblja može da zaključuje ocjene studentima na njemu dodijeljenim predmetima. Ovjeru ocjena obavljaju referenti studentske službe.

- Pregled aktuelnih ispitnih rokova i termina na predmetu, kao i spiska prijavljenih studenata je osnovna mogućnost sekcije „Ispiti". Uz svaki ispitni termin moguće je definisati tzv. evaluciju koja preciznije određuje šta se konkretno polaže u datom terminu (parcijalni ispit, integralni ispit, odbrana seminarskog rada itd.). Nakon što se ispit održi, svakoj od prijava se može promijeniti status, npr. Polagao, Prijavljen i Odjavljen, te za sve one prijave koje su u statusu „Polagao" unijeti rezultati ispita (u skladu sa evalucijom koja je prethodno definisana).

Dopuniti prethodno kreirani domain model elementima koji čine osnov modula *eNastava*.

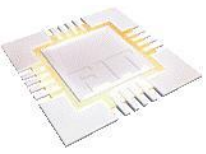
Ispitni zadatak:

U toku je organizacija konferencije na Fakultetu informacijskih tehnologija, te je potrebno kreirati web aplikaciju putem koje će se postavljati osnovne informacije o programu, omogućiti registracija učesnika i prijava radova.

Svi zainteresovani učesnici (*Ime, Prezime, Inicijali, Spol, E-mail, Drzava, Kontakt*) imaju mogućnost registracije na konferenciju, po čemu dobivaju konfirmacijski mail sa pristupnim podacima. Nakon registracije, učesnici mogu izmijeniti svoj profil (lične i pristupne podatke) i prijaviti jedan ili više radova. Prilikom prijave rada (*Sifra, Naslov, Sažetak, DatumPrijave*) moguće je odmah priložiti i dokument ili to uraditi naknadno putem pregleda svih aktivnih prijava. Uz prijavu se obavezno definiše i oblast kojoj rad pripada, te autori i njihovi kontakt podaci (*Zvanje, Ime, Prezime, E-mail*).

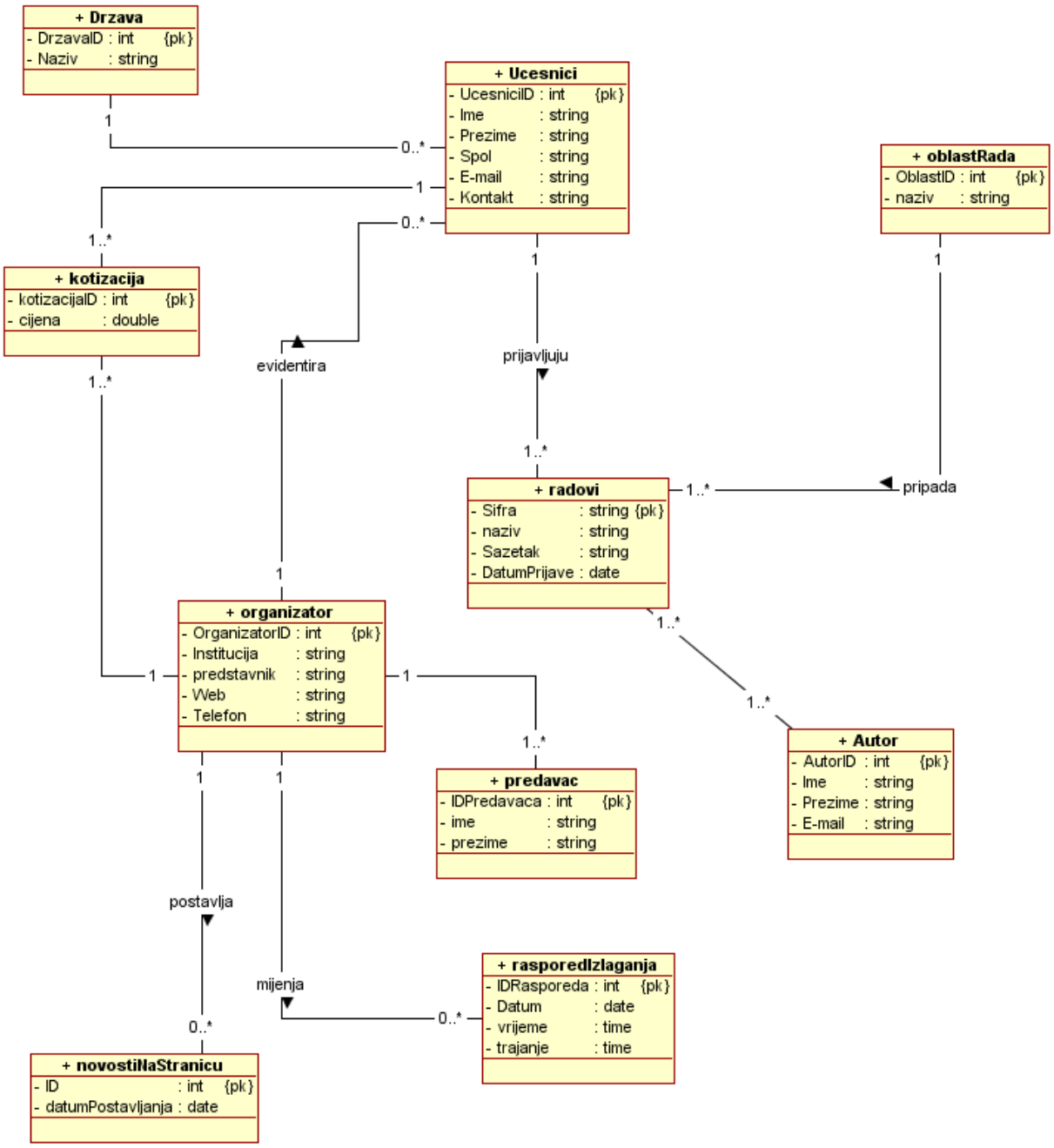
Organizator (*Institucija, Predstavnik, Web, Telefon*) se brine za postavljanje novosti na stranicu i definisanje osnovnih informacija o konferenciji (oblasti za prijavu radova, rokovi za predaju, kotizacije za učesnike i sl.). Također, on se brine i za ažuriranje programa konferencije, te evidenciju svih predavača. Svaka izmjena konferencijskog programa, odnosno rasporeda izlaganja (*Datum, Vrijeme, Trajanje*), zahtijeva slanje obavještenja registrovanim učesnicima putem mail-a.

Nakon završetka konferencije, učesnici mogu pretraživati i preuzimati sve radove koje je odobrio organizator.



Studentsko rješenje:

Ispod je prijedlog rješenja studentice koja je na ispitu osvojila **20** od mogućih 25 bodova. Dijagram je kreiran u alatu Open ModelSphere.



5. Ukratko

Aktivnost izrade domain modela treba da „definira dijagram klasa bez operacija unutar domene promatranja". Ovim modelom predstavljaju se koncepti, odgovarajući atributi i potrebne asocijacije između koncepata. Domain model odgovara klasičnom modelu objekti-veze. U okviru ove aktivnosti definira se lista kandidata korištenjem liste kategorije koncepta i identifikacijom imenica iz use case dokumenata, dodaju se atributi i odgovarajuće asocijacije. Da zaključimo, u ovoj aktivnosti se identificiraju uloge ljudi i stvari od interesa koji će biti angažirani u procesima.

Prateću video lekciju možete pogledati u sklopu YouTube kanala na sljedećem linku:

- Domain model (Open ModelSphere 3.2)

6. Literatura

1. Martin Fowler, UML Distilled Third Edition: A brief guide to the standard object modeling language, 2004. Download (Dostupno: 06.03.2017.).
2. Craig Larman, Applyng UML and paterns, Prentice Hall, 2004.
3. https://www.visual-paradigm.com/
4. https://www.youtube.com/user/VisualParadigm/
5. http://www.uml.org/