UDŽBENICI SVEUČILIŠTA U ZAGREBU
MANUALIA UNIVERSITATIS STUDIORUM ZAGRABIENSIS

Nakladnik
Graphis d.o.o.

Recenzenti
prof. dr. sc. Nikola Bogunović
izv. prof. dr. sc. Ninoslav Slavek

Priprema i dizajn
Graphis d.o.o.

Za nakladnika
Elizabeta Šunde, dir.

Objavljivanje ovog sveučilišnog priručnika odobrio je Senat Sveučilišta u Zagrebu, rješenjem klasa: 32-01/12-01/30; ur. br.: 380-061-117-12-2 od 21.11.2012.

ISBN 978-953-279-035-1

Cip zapis dostupan u računalnome katalogu Nacionalne i sveučilišne knjižnice u Zagrebu pod brojem 887718.

Tiskano u Hrvatskoj

dr. sc. Alan Jović, dipl. ing.
dr. sc. Marko Horvat, dipl. ing.
dr. sc. Igor Grudenić, dipl. ing.

UML-dijagrami

Zbirka primjera i riješenih zadataka

GRAPHIS
ZAGREB

Sadržaj

Predgovor

U literaturi postoji razmjerno velik broj knjiga o raznim praktičnim aspektima izrade programske potpore (engl. *software*) u okviru područja koje se naziva programsko inženjerstvo (engl. *software engineering*). Pritom je fokus stručne literature najčešće na raznim tehnikama programiranja, programskim jezicima, programskim knjižnicama i programskim alatima. Vrlo je malo priručnika koji se bave inženjerskim, sustavnim pristupom modeliranju programske potpore. Pogotovo nedostaje bilo kakve slične literature na hrvatskom jeziku.

Nedostatak literature navodi na zaključak da su današnji procesi programskog inženjerstva neorganizirani ili još uvijek pod sveobuhvatnom kontrolom mentalnog sklopa jednog ili nekolicine programera. To je, međutim, daleko od istine. Bilo koja ozbiljna tvrtka nužno radi na projektima koji zahtjevaju opsežnu tehničku dokumentaciju i koji si ne mogu dozvoliti da ovise o mentalnim procesima nekolicine programera. Upravo suprotno, programska potpora mora biti pravilno dokumentirana i poduprta modelima kako bi bilo koji inženjer mogao nastaviti njezin razvoj.

Modelno-usmjereni pristup oblikovanju programske potpore (engl. *Model-Based Design*) danas je u srcu procesa programskog inženjerstva. U praksi se modelno-usmjereni pristup često kombinira s drugim pristupima kao što su komponentni i agilni, tvoreći zajednički vrlo snažan i uspješan hibridni proces, na zadovoljstvo razvojnog tima, klijenata i krajnjih korisnika. Srž modelno-usmjerenog pristupa čini jezik za izgradnju modela. Iako je u prošlosti bilo raznih pokušaja tekstualnog opisa modela programskih sustava, tek je grafički prikaz doveo do stvarnog poboljšanja učinkovitosti cjelokupnog razvojnog procesa. Grafički prikaz olakšao je razumijevanje programkog sustava svima. Zasigurno najuspješniji jezik za modeliranje programske potpore, koji se tijekom godina postepeno razvijao i postao *de facto* industrijskom normom je UML.

U ovom priručniku, autori kroz primjere i riješene zadatke pokazuju ispravan način primjene UML-a. Neizbježna je činjenica da će se ova norma mijenjati te je namjera autora da s vremena na vrijeme revidiraju neke detalje priručnika. Ipak, trenutačna inačica priručnika predstavlja cjeloviti pogled u svijet modeliranja programske potpore korištenjem jezika UML. Autori se nadaju da će ovaj priručnik pomoći svim inženjerima kojima nedostaje formalno obrazovanje o ovom grafičkom jeziku i koji ga žele naučiti kako bi ga uspješno primijenili u praksi.

U Zagrebu, 30. 9. 2014.

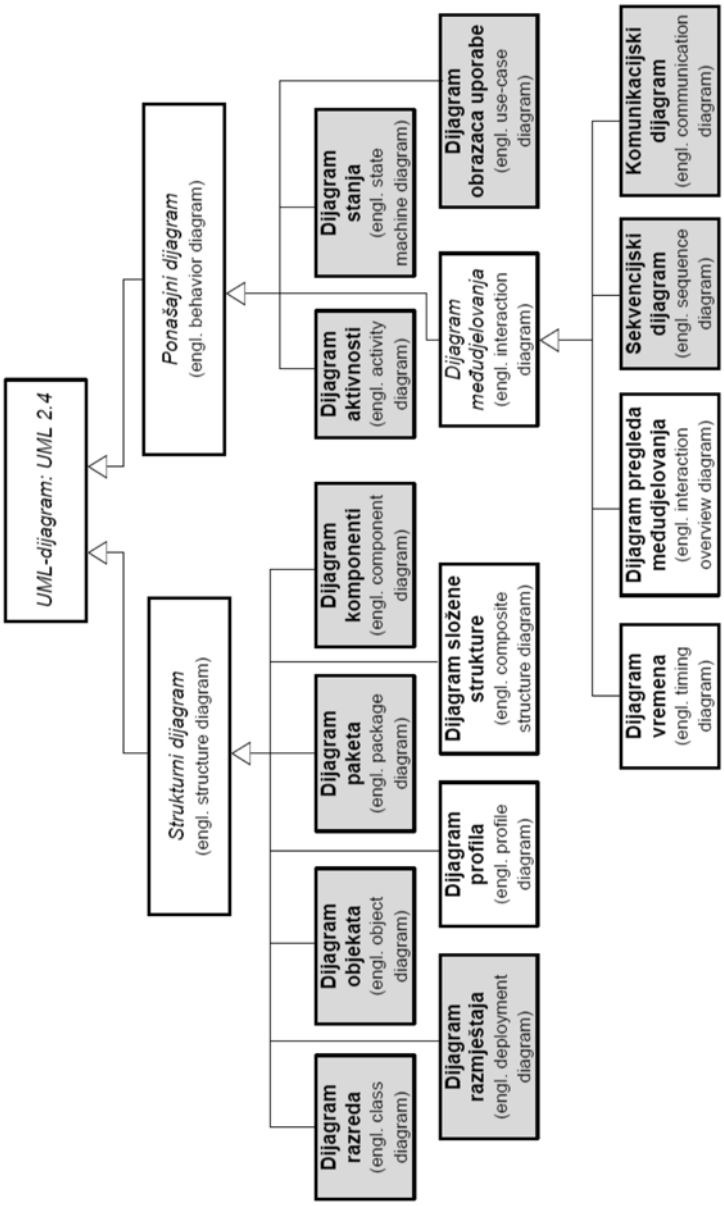
Autori

1. Uvod

Ujedinjeni jezik za modeliranje (engl. *Unified Modelling Language*, kraće: UML) je normirani jezik opće namjene koji se koristi za modeliranje računalnih sustava temeljenih na objektno-orijentiranoj paradigmi. Jezik je normirala Grupa za upravljanje objektima (engl. *Object Management Group*, kraće: OMG) s prvom normom iz 1997., a s trenutačnom inačicom UML 2.4.1 iz 2011. godine i normiran od strane ISO-a u 2012. UML uključuje skup tehnika koje ostvaruju grafički prikaz objektno-orijentiranih računalnih sustava. Računalni sustav modelira se raznovrsnim dijagramima, od kojih se svaki koristi za prikaz sustava iz ponešto drugačije perspektive.

Dijagrami unutar UML-a mogu se podijeliti s obzirom na dinamičnost na statičke i dinamičke dijagrame. Statički dijagrami ne razmatraju vremensku komponentu sustava, već daju sliku dijelova ili cijelog sustava kakav postoji u nekom trenutku. Težnja dinamičkih dijagrama je da uključe međudjelovanje sudionika i vremensku komponentu u opis sustava kako bi se modelirali slijedovi događaja unutar sustava. Nešto suvremenija podjela dijagrama je ona koja dijeli UML-dijagrame s obzirom na to da li modeliraju strukturu sustava (engl. *structure diagram*) ili ponašanje sustava (engl. *behavior diagram*). Ovakva podjela se okvirno podudara s podjelom na statičke (strukturni) i dinamičke (ponašajni) dijagrami, s iznimkom dijagrama obrazaca uporabe koji, iako modelira ponašanje, ne modelira vremensku komponentu sustava. Podjela UML-dijagrama prema normi UML 2.4 prikazana je na slici 1.1. U okviru ove podjele postoji veći broj pojedinačnih dijagrama, od čega se u okviru ove zbirke obrađuju oni dijagrami koji su označeni sivo na slici 1.1.

Cilj ove zbirke je predočiti što više normiranih UML-dijagrama kako bi se omogućila kvalitetna komunikacija na grafičkoj razini između inženjera koji su uključeni u razvoj računalnog sustava. Pritom je potrebno ovladati znanjem o komponentama svakog opisanog dijagrama i razumjeti za što se taj dijagram koristi. U zbirci je dan naglasak na tri dijagrama koji se najčešće koriste u praksi: dijagramu obrazaca uporabe, dijagramu razreda i sekvencijskom dijagramu. Čitatelji se upoznaju s dijagramima kroz nekoliko primjera i riješenih zadataka za svaku vrstu dijagrama. Rješavanjem zadanih zadataka či-



Slika 1.1. Pregled UML-dijagrama, norma UML 2.4 [OMG 2011]

tatelj može usporediti svoja rješenja s onima koja su dana u ovoj zbirci i na taj način naučiti ispravan način crtanja UML-dijagrama.

Zbirka je namijenjena poglavito studentima 3. godine preddiplomskog studija računarstva, na kolegiju Oblikovanje programske potpore (OPP), u sklopu Zavoda za elektroniku, mikroelektroniku, računalne i inteligentne sustave, Fakulteta elektrotehnike i

računarstva, Sveučilišta u Zagrebu (FER). Namjena zbirke je da se koristi kao dodatni nastavni materijal koji će olakšati studentima savladavanje gradiva iz modeliranja računalnog sustava korištenjem jezika UML u okviru projekta iz kolegija OPP, što pokriva približno trećinu gradiva kolegija. Također, vježbanjem zadataka iz ove zbirke studenti se mogu uspješno pripremiti za ispite iz ovog kolegija. Zbirka je također namijenjena i svim ostalim zainteresiranim čitateljima koji bi željeli naučiti modelirati sustave korištenjem jezika UML.

Primjeri i zadaci unutar ove zbirke crtani su korištenjem dvaju alata dostupnih studentima FER-a, a to su Microsoft Visio 2007 i 2010 (putem licence MSDNAA) i Astah Community v.6.8.0 (slobodan programski proizvod [Change 2014]). Izbor alata za crtanje UML-dijagrama ostavljen je na volju čitatelja budući da kod samih alata postoji velika raznolikost po pitanju licenciranja, podržanih dijagrama i mogućnosti crtanja njihovih komponenata.

Zbirka je ustrojena na sljedeći način. U poglavlju 2 opisuju se dijagrami obrazaca uporabe: njihovo značenje iz perspektive krajnjeg korisnika sustava i opis bitnih komponenti dijagrama. Poglavlje 3 posvećeno je dijagramima međudjelovanja, što znači sekvencijskim i komunikacijskim dijagramima. Opisuju se karakteristike tih dijagrama, njihove razlike i daju se razrađeni primjeri da bi se što lakše razumjelo kako modelirati međudjelovanje dijelova sustava. Dijagrami razreda detaljno su razrađeni u opsežnom poglavlju 4, zajedno sa svim entitetima koji se na tim dijagramima prikazuju. Poglavlje 5 opisuje dijagrame paketa i objekata, koji služe kao nadopuna dijagramima razreda. Poglavlje 6 posvećeno je dijagramima stanja i dijagramima aktivnosti koji modeliraju dinamičko ponašanje dijelova sustava. Poglavlje 7 uključuje dijagrame komponenti, a poglavlje 8 dijagrame razmještaja. Obje vrste dijagrama na sličan način modeliraju statičku sliku sustava iz perspektive povezanosti dijelova sustava, samo to čine na različitoj razini. U poglavlju 9 dana su rješenja zadataka iz svih ostalih poglavlja. Studentima se savjetuje da konzultiraju rješenja tek kad su sami izradili svoje rješenje zadatka na temelju primjera i uputa danih u ovoj zbirci. Literatura je navedena na kraju zbirke.

2. Dijagrami obrazaca uporabe

2.1. Karakteristike dijagrama

Dijagrami obrazaca uporabe (engl. *use-case diagrams*) koriste se da bi se prikazalo ponašanje sustava, dijelova sustava ili konkretnog razreda na način vidljiv korisniku sustava. Obrasci uporabe također služe tome da se razvojni tim i korisnici usaglase po pitanju ponašanja korisnika pri komunikaciji sa sustavom [Rational 2001]. Ponašanje sustava opisano je pomoću scenarija (engl. *use-cases* – obrasci (slučajevi) uporabe) i aktora (engl. *actors*) koji predstavljaju apstrakciju korisnika sustava, odnosno općenitije nekog od dionika (engl. *stakeholder*) sustava. Aktori mogu predstavljati i druge vanjske sustave koji sudjeluju u radu sustava koji se modelira. Jedan obrazac uporabe uključuje komunikaciju aktora s modeliranim sustavom koji se ostvaruje kao niz poruka među sudionicima obrasca uporabe, a koji doprinosi ostvarenju nekog jedinstvenog cilja. Na dijagramima obrazaca uporabe dodatno se definiraju odnosi između pojedinih obrazaca uporabe kao i odnosi između aktora.

Dijagrami obrazaca uporabe su statički UML-dijagrami (kao i dijagrami razreda, objekata, paketa i razmještaja), a također pripadaju i skupini ponašajnih dijagrama budući da modeliraju moguće ponašanje korisnika sustava.

Primjer 2.1.1. Blagajna u trgovačkom centru

Prikažite u dijagramu obrazaca uporabe funkcionalnost blagajne u trgovačkom centru.

U primjeru navedenom na slici 2.1 aktori u sustavu su Blagajnik i Kupac, označeni pojednostavljenim prikazom čovjeka (engl. *stickman*). Blagajnik može uključiti blagajnu, napraviti prijavu u sustav te napraviti transakciju. Te temeljne aktivnosti čine obrasce uporabe (engl. *use-case*) koji se prikazuju elipsom. Obrazac uporabe *Napravi transakciju* odnosi se na očitavanje svih artikala, ispis računa, odabir načina plaćanja i plaćanje robe.

3. Sekvencijski i komunikacijski dijagrami

3.1. Karakteristike dijagrama

Sekvencijski dijagrami (engl. *sequence diagram*) i komunikacijski dijagrami (engl. *communication diagram*) pripadaju široj skupini UML-dijagrama međudjelovanja (engl. *interaction diagram*). Obje vrste dijagrama spadaju u nadskupinu ponašajnih dijagrama (engl. *behavioral diagram*), zajedno s dijagramima stanja, aktivnosti i obrazaca uporabe. Stari naziv za komunikacijski dijagram je kolaboracijski dijagram.

Za razliku od dijagrama obrazaca uporabe, kod kojih vremenska komponenta nije važna, sekvencijski i komunikacijski dijagrami daju naglasak na vremenskom redoslijedu kojim se odvija međudjelovanje sudionika u sustavu, tako da ih se svrstava i u dinamičke UML-dijagrame.

Sudionici koji se modeliraju na sekvencijskim i komunikacijskim dijagramima mogu biti aktori (predočavanjem komunikacije aktora s dijagrama obrazaca uporabe) ili objekti, pri čemu se prikazuje komunikacija instanci razreda prikazanih na dijagramu

Tablica 3.1. Razlike između sekvencijskog i komunikacijskog dijagrama

Sekvencijski	**Komunikacijski**
Veći naglasak na vremenskoj uređenosti scenarija	Veći naglasak na pregledu scenarija, odnosno na međusobnoj komunikaciji
Redoslijed poruka sudionika odozgora lijevo prema dolje desno	Redoslijed poruka sudionika određenje brojčanim oznakama koje se stavljaju na poruke
Koristi se u ranim fazama specifikacije i analize projekta	Koriste se u fazi dizajniranja implementacije odnosa
Raširenији i čitljiviji	Sažetiji i teži za čitati
Teže izmjene	Lakše izmjene

4. Dijagrami razreda

4.1. Karakteristike razreda

Dijagrami razreda (engl. *class diagrams*) opisuju razrede i njihove međusobne veze. Jednako tako, dijagrami razreda opisuju vrste objekata unutar nekog sustava i njihove međusobne statične odnose.

Dijagrami razreda pripadaju strukturnoj skupini UML-dijagrama (engl. *structure diagram*). Oni ne opisuju događaje, stanja, aktivnosti ili bilo kakvu vremenski promjenjivu karakteristiku sustava koji se modelira. Naprotiv, dijagrami razreda su statični s obzirom na vremensku komponentu.

Razred ili klasa (engl. *class*) je osnovni tvorbeni element UML-dijagrama razreda. Stoga su drugi nazivi za dijagrame razreda dijagram klasa, ili *class dijagram*. Za ispravno razumijevanje definicije razreda važno je prvo odrediti značenje objekta. Objekt predstavlja entitet iz stvarnog svijeta ili neki koncept, odnosno apstrakciju nečega što ima dobro definirane granice i smisao u sustavu. Stoga je razred opis grupe objekata sa sličnim svojstvima, a svaki objekt je obvezno pojedinac (instanca) jedne klase.

Za svaki razred nužno je definirati naziv, a moguće je odrediti popis atributa i operacija. Makar atribute i operacije nije nužno definirati, bez njih razred nema implementacijsku svrhu. Za atribute nužno je odrediti njihov naziv i tip podataka, a za operacije njihovu definiciju koja uključuje naziv operacije, te sve ulazne i izlazne parametre.

Primjer 4.1.1. Definirati razred Student

Svaki student ima svoj identifikacijski broj (ID, podatkovnog tipa Long), ime (String), prezime (String) i prosjek ocjena (Double). Student može prijaviti ispit, odjaviti ispit i pristupiti ispitu. Za prijavu i odjavu ispita potrebna je šifra predmeta (Integer), a operacije vraćaju logičke (boolean) vrijednosti da li je izvršenje uspjelo ili ne. Pristupanje ispitu je definirano drugačije: ulazni argument operacije je šifra predmeta (Integer), a

5. Dijagrami objekata i paketa

5.1. Dijagrami objekata

5.1.1. Karakteristike objekata

Dijagrami objekata (engl. *object diagrams*) kao i dijagrami razreda pripadaju strukturnim UML-dijagramima, ali njihova uloga je različita: dijagrami objekata prikazuju strukturu sustava u nekom trenutku. Prikaz može biti djelomičan ili cjelovit, odnosno nije nužno prikazati objekte svih razreda sustava. U proizvoljno odabranim trenucima objekti sustava imaju različite vrijednosti atributa i dijagrami objekata prikazuju upravo te vrijednosti, zajedno s objektima i njihovim međusobnim vezama. Najčešće je dovoljno prikazati samo jedan trenutak u radu sustava, ali ponekad ako je stanje sustava izrazito dinamično potrebno je izraditi više dijagrama koji opisuju rad sustava u nekoliko trenutaka s karakterističnim stanjima objekata.

5.1.2. Definiranje objekata

Simbol objekta je pravokutnik s dva pretinca – jedan za naziv objekta i drugi za vrijednosti atributa. Objekti se razlikuju od razreda jer nemaju definicije atributa i procedura, ali imaju jasno označeno stanje važnih atributa. Prikazuju se atributi po kojima se pojedinci unutar dijagrama međusobno razlikuju, odnosno po kojima su oni specifični.

Primjer 5.1.2.1. Objekti razreda Osoba

Razred Osoba ima sljedeće atribute: jedinstvena šifra (Integer), ime (String), prezime (String) i OIB (String). Šifra je zaštićene vidljivosti, a ime, prezime i OIB javno dostupni podaci. Izradite tri pojedinca navedenog razreda s proizvoljnim vrijednostima atributa.

6. Dijagrami stanja i aktivnosti

6.1. Karakteristike dijagrama

Dijagrami stanja (engl. *statechart*, *UML state machine diagram*) i aktivnosti (engl. *activity diagram*) pripadaju ponašajnim dijagramima (engl. *behavioral diagram*) u UML-u, zajedno s dijagramom obrazaca uporabe. Za razliku od dijagrama obrazaca uporabe, dijagrami stanja i aktivnosti prikazuju funkcionalnost softverskog sustava iz perspektive unutrašnjosti sustava. Pritom ovi dijagrami ne prikazuju niti aktore niti vanjsko sučelje prema krajnjim korisnicima. Budući da razrađuju ponašanje sustava u smislu aktivnosti i prijelaza između stanja, svrstavaju se u dinamičke dijagrame.

Iako slični, ove dvije vrste dijagrama imaju i određene razlike koje se mogu uočiti. Prva razlika odnosi se na njihovu svrhu. Dijagram stanja služi za opis diskretnih stanja sustava i prijelaza između tih stanja. Težište mu je na unutarnjem djelovanju dijelova sustava i često prikazuje prijelaze između stanja u sustavu koji su poticani događajima. Dijagram aktivnosti prikazuje radni tok (ili kontrolni tok) aktivnosti koje se obavljaju u sustavu korak po korak. Stoga je kod dijagrama aktivnosti naglasak na jednostavnosti i poslovnim operacijama koje se uvijek odvijaju slijedno, jedna za drugom.

Druga razlika je u tome što dijagrami stanja na prijelazima između stanja sadrže naziv događaja koji je aktivirao prijelaz i često poziv izlazne metode ili operacije koja se događa, dok kod dijagrama aktivnosti to nije slučaj.

6.2. Dijagram stanja

6.2.1. Elementi dijagrama

Dijagram stanja sastoji se od sljedećih (najčešćih) gradivnih elemenata:

1. Početno stanje i konačno stanje. Početno stanje označeno je crnim krugom, a konačno stanje označeno je zaokruženim crnim krugom, slika 6.1. Dijagram stanja može imati jedno početno i više konačnih stanja.

7. Dijagrami komponenti

7.1. Karakteristike dijagrama

Dijagrami komponenti prikazuju komponente (strukturne cjeline) sustava i njihove međusobne odnose [Fowler 2000]. Komponenta je zasebna cjelina programske potpore s vlastitim sučeljem. Komponenta predstavlja fizičku i stvarnu implementaciju logičkih elemenata sustava kao što su razredi, sučelja i pridruživanja [Booch 1999]. Komponentni dijagrami pomažu u modeliranju fizičkih cjelina sustava kao što su izvršne datoteke, programske biblioteke, tablice, datoteke i svi drugi dokumenti. Često se kaže da su u UML-u sve fizičke „stvari“ modelirane kao komponente [Booch 1999].

U izradi programske podrške mnogi operacijski sustavi i računalni jezici podržavaju koncept komponente: objektne biblioteke, izvršne datoteke, DCOM i COM+ komponente i Enterprise Java Beans su primjeri komponenata koje se mogu direktno predstaviti u UML-u korištenjem komponenti [Booch 1999]. Jedan razred može biti predstavljen s više komponenata, ali samo s jednim paketom. Na primjer, razred Java *String* je smješten u paket *java.lang*, ali istodobno sastoji se od mnogo komponenata [Fowler 2000].

Dijagrami komponenti su strukturni UML-dijagrami. Prikazuju vremenski nepromjenjiva (statička) svojstva sustava s fizičkog aspekta implementacije. Stoga se uz dijagrame razmještaja još nazivaju fizički dijagrami te se često prikazuju zajedno u jednom UML-dijagramu komponenti i razmještaja.

Primjer 7.1.1. Sistemska DLL datoteka

Prikazati sistemsku datoteku 'gdi32.dll' operacijskog sustava Windows. Datoteka eksportira *Graphics Device Interface* (GDI) programsko sučelje. Zanemariti ostala sučelja komponente.

Rješenje primjera 7.1.1 prikazano jer na slici 7.1.

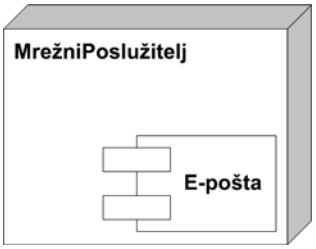
8. Dijagrami razmještaja

8.1. Karakteristike dijagrama

Dijagrami razmještaja (engl. *deployment diagrams*) opisuju topologiju sklopovlja i programsku potporu koja se koristi u implementaciji sustava u njegovom radnom i produkcijskom okruženju. Drugim riječima, dijagrami razmještaja prikazuju računalne resurse koji su neophodni za ispravno funkcioniranje sustava i njihove međusobne odnose: stvarne uređaje (poslužitelje, radne stanice, korisnička računala, itd.), komponente programske podrške koje se na njima izvršavaju i veze između prikazanih resursa. Dijagrami razmještaja, kao i dijagrami paketa i razreda, su statički i strukturni UML-dijagrami.

Primjer 8.1.1. Mrežni poslužitelj s komponentom

Prikažite u dijagramu razmještaja mrežni poslužitelj s komponentom za slanje elektroničke pošte. Rješenje primjera 8.1.1 prikazano je na slici 8.1.

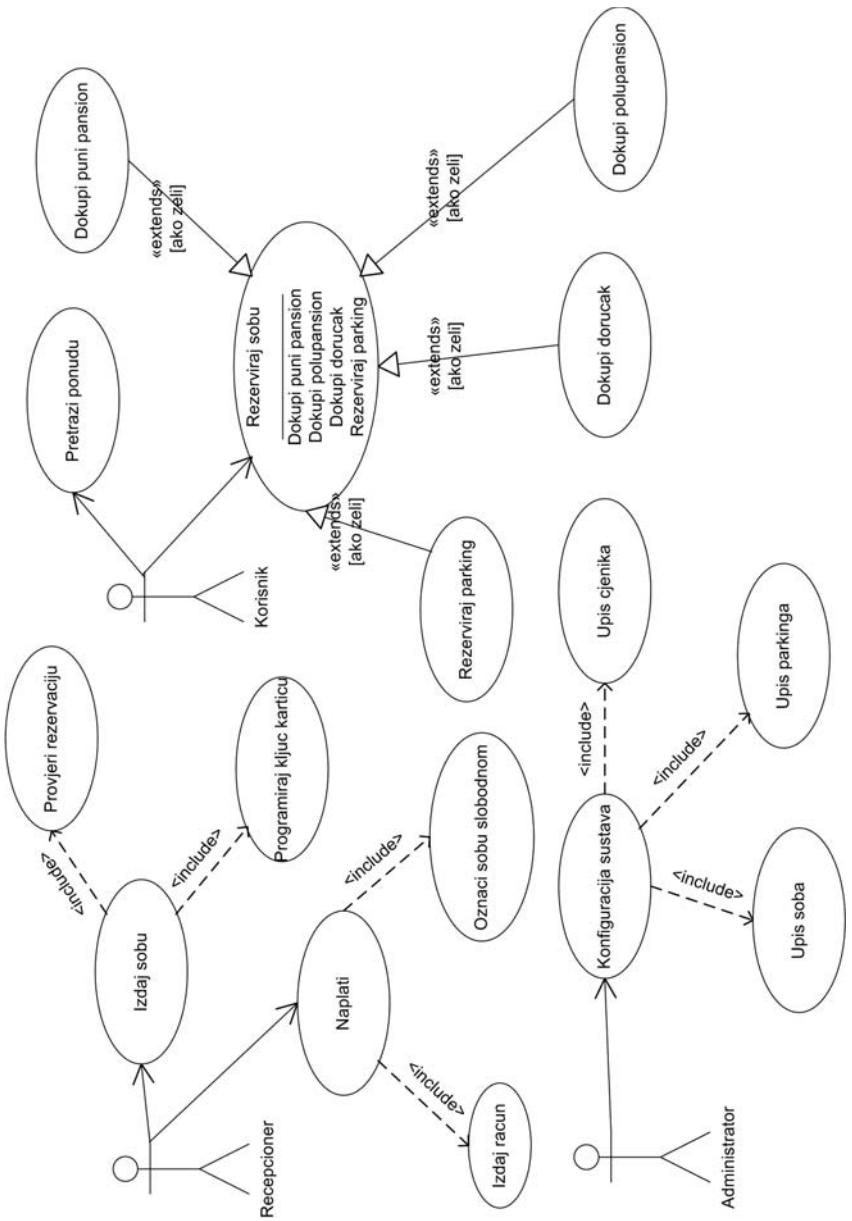


Slika 8.1. Rješenje primjera 8.1.1.

9. Rješenja zadataka

9.1. Dijagrami obrazaca uporabe

Zadatak 2.1.



Literatura

[Bell 2004] D. Bell, *UML basics: The sequence diagram*, 2004, dostupno na: http://www.ibm.com/developerworks/rational/library/3101.html, [pristupljeno dana 29. 8. 2014.]

[Booch 1999] G. Booch, J. Rumbaugh, I. Jacobson, *The Unified Modeling Language User Guide*, Addison-Wesley, 1999.

[Change 2014] Change Vision, *Astah*, 2014, dostupno na: http://www.astah.net/ [pristupljeno dana 22. 8. 2014.]

[Fowler 2000] M. Fowler, K. Scott, *UML Distilled*, 2nd ed., Addison-Wesley, 2000.

[Fowler 2004] M. Fowler, *UML Distilled: a brief guide to the Standard object modeling language*, 3rd ed., Addison-Wesley, 2004.

[Holub 2012] A. I. Holub, *Allen Holub's UML Quick Reference* (UML 2.0), version 2.1.3, dostupno na: http://www.holub.com/goodies/uml/, [pristupljeno dana 29. 8. 2014.]

[Lethbridge 2005] T. C. Lethbridge, R. Laganière, *Object-Oriented Software Engineering*, McGraw-Hill Education, 2005.

[OMG 2011] Object Management Group, *OMG Unified Modeling Language* (OMG UML), Infrastructure, Version 2.4.1, 2011, dostupno na: http://www.omg.org/spec/UML/2.4.1/Infrastructure/PDF/, [pristupljeno dana 29. 8. 2014.]

[Quatrani 2002] T. Quatrani, *Visual Modeling with Rational Rose 2002 and UML*, 3rd ed., Addison-Wesley, 2002.

[Rational 2001] Rational Software Corporation, *Artifact: Use-Case Model*, dostupno na: http://www.ts.mah.se/RUP/RationalUnifiedProcess/process/artifact/ar_ucmod.htm, [pristupljeno dana 29. 8. 2014.]