Tehničko veleučilište u Zagrebu

Informatičko-računarski odjel (INRO)

Dokumentiranje projekta i modeliranje programske potpore jezikom UML

Primjer uporabe CASE alata

Pripremili: dr. sc. Danko Ivošević, pred.

doc. dr. sc. Marko Horvat, viši pred.

Zagreb, 11. rujna 2023.

DOKUMENTIRANJE PROJEKTA I MODELIRANJE PROGRAMSKE POTPORE JEZIKOM UML

PRIMJER UPORABE CASE ALATA



TEHNIČKO VELEUČILIŠTE U ZAGREBU ZAGREB UNIVERSITY OF APPLIED SCIENCES

izdavač	Tehničko veleučilište u Zagrebu Vrbik 8, Zagreb
za izdavača	prof. dr. sc. Jana Žiljak Gršić
autori	dr. sc. Danko Ivošević, pred. doc. dr. sc. Marko Horvat, viši pred.
recenzenti	izv. prof. dr. sc. Alan Jović, dipl. ing. dr. sc. Nikolina Frid, mag. ing. comp.
vrsta djela	priručnik
	Objavljivanje je odobrilo Vijeće Tehničkog veleučilišta u Zagrebu odlukom broj: 5959-10/23 od 19.09.2023.
ISBN	978-953-8444-09-8
CIP zapis	Dostupan u računalnom katalogu Nacionalne i sveučilišne knjižnice u Zagrebu pod brojem _____

Autori zadržavaju sva prava nad sadržajem priručnika.

Sadržaj

Uvodne napomene 14
1. Programsko inženjerstvo i modeliranje procesa 15
2. Primjer modeliranja uporabom jezika UML 17
2.1. Osnove UML dijagrama 17
2.2. Opis projekta 18
2.3. Dijagram slučajeva uporabe 20
Aktori 20
Slučajevi uporabe 21
Odnos pridruživanja 25
Ostale vrste odnosa 27
Odnos poopćenja 27
Odnos obuhvaćanja 32
Odnos proširenja 34
Pojednostavljenja prikaza 35
Razlikovanje vrste odnosa 37
Česte pogreške 38
Nepostojanje granice sustava 38
Sustav kao poseban aktor 39
Krivi smjerovi veza 39
Zamjena semantike 40
Međusobno povezivanje ili ulančavanje slučajeva uporabe 41
Obuhvaćanje slučajeva uporabe različitih inicijatora 43
2.4. Dijagram aktivnosti 45
Kupnja proizvoda 46
Narudžba novih proizvoda 53
Redovita pohrana podataka 56
Česte pogreške 57
Miješanje semantike račvanja i skupljanja sa semantikom uvjetnog grananja 57
Neispravna notacija uvjeta kod uvjetnog grananja 58
2.5. Dijagram slijeda 60
Slučaj uporabe 'Prijava u sustav' (UC4, UC5) 60
Slučaj uporabe 'Pregled proizvoda' (UC1, UC2) 64
Slučaj uporabe 'Kupnja proizvoda' (UC6, UC7, UC8) 69
Slučaj uporabe 'Narudžba novih proizvoda' (UC12, UC13, UC14) 70
Slučaj uporabe 'Odgovaranje na upite korisnika' (UC3, UC10) 70

Ostali slučajevi uporabe 71
2.6. Dijagram komunikacije 73
Slučaj uporabe 'Pregled proizvoda' (UC1, UC2) 73
Slučaj uporabe 'Prijava u sustav' (UC4, UC5) 74
Slučaj uporabe 'Odgovaranje na upite korisnika' (UC3, UC10)“ 75
2.7. Dijagram razreda 77
Razina domene (konceptualna) 77
Dijagram razreda podatkovnog modela 78
Mogućnosti drugačijih rješenja zadatka 82
Izrada izvornog koda 83
Dijagram razreda prodajnog modela 86
Dijagram razreda modela nabave 88
Razina ostvarenja (implementacijska) 89
2.8. Dijagram objekata 93
2.9. Dijagram stanja 94
Kupnja proizvoda 94
Baza podataka 95
Česte pogreške 96
2.10. Dijagram komponenti 99
Komponenta kao logička cjelina 99
Hijerarhija komponenti 103
Proširena notacija sučelja 104
Komponenta kao fizički element 104
2.11. Dijagram razmještaja 106
Grafička notacija čvorova 108
3. Modelno usmjereno programsko inženjerstvo 109
3.1. Uvod i definicija 109
3.2. Različiti pogledi i vrste modela 110
3.3. Standardi i alati za modeliranje 113
4. Uporaba alata *Enterprise Architect* 115
4.1. Stvaranje novog projekta 115
Hijerarhijska organizacija sadržaja 116
Umetanje novog dijagrama 118
4.2. Alatni okviri i primjeri uporabe 120
Dijagram slučajeva uporabe 120
Umetanje elemenata 121

Umetanje odnosa između elemenata 124
Umetanje bilješke ili komentara 126
Dijagram aktivnosti....127
Postavljanje particija....127
Umetanje čvorova i prijelaza....128
Različite vrste završnih čvorova....132
Akcije vođene događajima....132
Dijagram slijeda 132
Umetanje aktora i objekata 133
Umetanje fragmenata....136
Umetanje poruke 138
Načini pobrojavanja....140
Referenciranje drugog dijagrama....141
Dijagram komunikacije 141
Umetanje poruka 142
Automatska pretvorba iz dijagrama slijeda 143
Nova skupina pobrojavanja poruka....144
Dijagram razreda 145
Definicija razreda....146
Definicija svojstava....150
Umetanje postojećeg razreda....150
Povezivanje razreda....151
Generiranje izvornog koda....154
Definicija vrste podataka 157
Uvoz izvornog koda....159
Kombiniranje različitih paketa 160
Dijagram objekata....161
Opis stanja objekta....162
Dijagram stanja....164
Opis stanja 164
Opis prijelaza....166
Dijagram komponenti 166
Definicija sučelja....167
Umetanje složenijih komponenti 169
Definicija stereotipa 172
Uporaba proširene notacije sučelja....174

Umetanje artefakata 175
Dijagram razmještaja 177
Umetanje i hijerarhija čvorova 177
Umetanje veza između čvorova 179
Definicije stereotipova 181
Uvoz biblioteke simbola 182
5. Zaključak 186
6. Literatura 187
7. Dodaci 190
7.1. Generirani programski kodovi 190
Dijagram razreda korisničkog modela 190
Pobrojavanje 'RazinaPristupa' (datoteka *RazinaPristupa.java*) 190
Razred 'Korisnik' (datoteka *Korisnik.java*) 190
7.2. Primjeri ispitnih zadataka 192
Dijagram slučajeva uporabe 192
Dijagram razreda 194

Popis slika

Slika 2.1.1 Pregled glavnih UML dijagrama unutar standarda 2.5.1. 18
Slika 2.3.1 Primjer jednostavnog dijagrama slučajeva uporabe s vezama pridruživanja 26
Slika 2.3.2 Primjer dijagrama slučajeva uporabe s vezama pridruživanja 27
Slika 2.3.3 Moguća rješenja povezanosti aktora-korisnika i slučajeva uporabe 28
Slika 2.3.4 Veze pridruživanja s primjerom bilješke dijagrama slučajeva uporabe 29
Slika 2.3.5 Odnosi između aktora – korisnika sustava 30
Slika 2.3.6 Detaljniji prikaz slučaja uporabe ‘Osvježavanje baze podataka’ 31
Slika 2.3.7 Bilješka o postojanju ‘Baze podataka’ kao podrazumijevanog aktora 32
Slika 2.3.8 Veza obuhvaćanja na dijagramu slučajeva uporabe 33
Slika 2.3.9 Dijagram slučajeva uporabe ‘Prikaz prodaje’ s umetnutim vezama obuhvaćanja 33
Slika 2.3.10 Izgled točke proširenja na dijagramu slučajeva uporabe 34
Slika 2.3.11 Konačni izgled dijagrama slučajeva uporabe ‘Prikaz prodaje auto-dijelova’ 35
Slika 2.3.12 Dijagram slučajeva uporabe prodaje auto-dijelova sa strane vanjskih korisnika 36
Slika 2.3.13 Dijagrama slučajeva uporabe prodaje auto-dijelova sa strane poslovnog osoblja 37
Slika 2.3.14 Česte pogreške: primjer nepostojanja granice sustava 38
Slika 2.3.15 Česte pogreške: umetanje sustava kao posebnog aktora 39
Slika 2.3.16 Česte pogreške: primjer krivih usmjerenja pojedinih veza 40
Slika 2.3.17 Primjer specijalizacija slučaja uporabe 'Plaćanje računa' 40
Slika 2.3.18 Česte pogreške: zamjena semantike obuhvaćanja i poopćenja 41
Slika 2.3.19 Česte pogreške: zamjena semantike proširenja i poopćenja 41
Slika 2.3.20 Česte pogreške: umetanje odnosa pridruživanja između slučajeva uporabe 42
Slika 2.3.21 Česte pogreške: umetanje odnosa obuhvaćanja 42
Slika 2.3.22 Česte pogreške: „ulančavanje“ slučajeva uporabe 43
Slika 2.3.23 Česte pogreške: povezivanje slučajeva uporaba 43
Slika 2.3.24 Česte pogreške: povezivanje slučajeva uporabe različitih inicijatora 44
Slika 2.3.25 Česte pogreške: povezivanje slučajeva uporabe različitih inicijatora 44
Slika 2.4.1 Postavljanje particija dijelova sustava koji sudjeluju na dijagramu 46
Slika 2.4.2 Postavljanje slijeda aktivnosti na dijagramu aktivnosti 47
Slika 2.4.3 Primjer račvanja i skupljanja na dijagramu aktivnosti 47
Slika 2.4.4 Konačni oblik dijagrama aktivnosti 48
Slika 2.4.5 Prikaz situacija da proizvod nije nađen 49
Slika 2.4.6 Prikaz situacije da kupac ne želi nađeni proizvod 49
Slika 2.4.7 Prikaz opcija neuspješne kupnje proizvoda 50
Slika 2.4.8 Detaljniji oblik dijagrama aktivnosti 51
Slika 2.4.9 Isječak dijagrama slučajeva uporabe vezan uz provedbu kupnje proizvoda 52
Slika 2.4.10 Aktivnosti trgovca prilikom narudžbe novih proizvoda 53
Slika 2.4.11 Primjer primjene signala u dijagramu aktivnosti 54
Slika 2.4.12 Dijagram aktivnosti narudžbe novih proizvoda 55
Slika 2.4.13 Primjer primjene vremenskog signala u dijagramu aktivnosti 56
Slika 2.4.14 Dijagram aktivnosti sigurnosne pohrane podataka 57
Slika 2.4.15 Primjer miješanja semantike račvanja sa semantikom uvjetnog grananja 58
Slika 2.4.16 Primjer neispravne notacije kod uvjetnog grananja 59
Slika 2.5.1 Kontekst slučajeva uporabe registracije i prijave korisnika u sustav 61
Slika 2.5.2 Registracija korisnika u sustav – početak oblikovanja 61
Slika 2.5.3 Dijagram slijeda registracije korisnika 62

Slika 2.5.4 Dijagram slijeda prijave korisnika u sustav 63
Slika 2.5.5 Isječak dijagrama slučajeva uporabe koji se odnosi na pretragu i pregled proizvoda 64
Slika 2.5.6 Primjer fragmenta opt na dijagramu slijeda 65
Slika 2.5.7 Dijagram slijeda s pretragom i pregledom detalja....66
Slika 2.5.8 Primjer dijagrama slijeda slučajeva uporabe “Pretraga proizvoda” i “Pregled proizvoda” 67
Slika 2.5.9 Primjer detaljnijeg dijagrama slijeda za slučajeve uporabe “Pretraga proizvoda” i “Pregled proizvoda” 68
Slika 2.5.10 Isječak dijagrama slučajeva uporabe vezan uz narudžbu novih proizvoda 70
Slika 2.5.11 Isječak dijagrama slučajeva uporabe vezan uz korisničku podršku korisniku 70
Slika 2.5.12 Dijagram slijeda kod odgovaranja na upite korisnika....71
Slika 2.6.1 Primjer dijagrama komunikacije slučaja uporabe “Pretraga proizvoda” 73
Slika 2.6.2 Automatski generirani dijagram komunikacije za slučajeve uporabe 'Pretraga proizvoda' i 'Pregled proizvoda' 74
Slika 2.6.3 Dorađeni dijagram komunikacije slučajeva uporabe 'Pretraga proizvoda'' i 'Pregled proizvoda'....74
Slika 2.6.4 Dijagram komunikacije za registraciju korisnika 75
Slika 2.6.5 Dijagram komunikacije za prijavu korisnika u sustav....75
Slika 2.6.6 Dijagram komunikacije kod odgovaranja na upite korisnika....76
Slika 2.7.1 Prikaz usmjerene veze pridruživanja (strelica) u dijagramu razreda 78
Slika 2.7.2 Dio dijagrama razreda vezan uz model automobila....80
Slika 2.7.3 Dio dijagrama razreda s proizvođačima i dobavljačima dijelova. 81
Slika 2.7.4 Dijagram razreda vezan za podatkovni model razreda AutomobilskiDio....82
Slika 2.7.5 Početni dijagram razreda korisničkog modela 84
Slika 2.7.6 Konačni oblik dijagrama razreda korisničkog modela....85
Slika 2.7.7 Pregled stvorenih datoteka izvornog koda....85
Slika 2.7.8 Dijagram razreda podsustava prodaje auto-dijelova....87
Slika 2.7.9 Prikaz strukture datoteka izvornog koda....89
Slika 2.7.10 Razredi upravljačkog dijela programske potpore (paket 'controller') 90
Slika 2.7.11 Razred veze prema bazi podataka (paket 'factory') 90
Slika 2.7.12 Razredi dohvata i upravljanja podacima (paket 'mongo')....91
Slika 2.7.13 Dijagram razreda ostvarenja administratorskih ovlasti, kombiniranje razreda iz više paketa 92
Slika 2.8.1 Dijagram objekata dobavljača i proizvođača nekog auto-dijela....93
Slika 2.9.1 Dijagram stanja košarice pri kupnji proizvoda....95
Slika 2.9.2 Dijagram stanja baze podataka tijekom radnog tjedna 96
Slika 2.9.3 Primjer miješanja semantike dijagrama aktivnosti i dijagrama stanja....98
Slika 2.10.1 Prikaz komponenti unutar dijagrama komponenti....99
Slika 2.10.2 Primjer ponuđenog i zahtijevanog sučelja komponenti 100
Slika 2.10.3 Ponuđena i zahtijevana sučelja komponenti....100
Slika 2.10.4 Ponuđeno javno sučelje komponente....101
Slika 2.10.5 Prikaz različitih vrsta sučelja komponenti 101
Slika 2.10.6 Dijagram komponenti sustava prodaje auto-dijelova 102
Slika 2.10.7 Prikaz odnosa komponenti uporabom sučelja 103
Slika 2.10.8 Dijagram komponenti s prikazom podsustava 'Web Prodaja' 103
Slika 2.10.9 Primjer prikaza s proširenom notacijom sučelja....104
Slika 2.10.10 Dijagram komponenti administratorskih poslova....105
Slika 2.11.1 Primjer dijagrama razmještaja za sustav prodaje auto-dijelova 107

Slika 2.11.2 Primjer jednostavnog dijagrama razmještaja 108
Slika 2.11.3 Primjer grafički uljepšanog dijagrama razmještaja 108
Slika 3.1.1 Osnovni koncepti modelno usmjerenog razvoja (prema [15]) 110
Slika 3.2.1 Uloga modelno usmjerene arhitekture (prema [17]) 112
Slika 4.1.1 Otvaranje novog projekta alatom Enterprise Architect 115
Slika 4.1.2 Početni izgled sučelja 116
Slika 4.1.3 Postavljanje naziva modela u projektnom pregledniku 117
Slika 4.1.4 Upravljanje elementima hijerarhije modela projekta uporabom alatne trake 117
Slika 4.1.5 Definicija novog paketa 118
Slika 4.1.6 Početno sučelje alata Enterprise Architect s otvorenim novim paketom 118
Slika 4.1.7 Izbor ponašajnih vrsta UML dijagrama unutar alata Enterprise Architect. 119
Slika 4.1.8 Izbor strukturnih vrsta UML dijagrama unutar alata Enterprise Architect. 120
Slika 4.2.1 Alatni okviri modela slučajeva uporabe: (a) elementi, (b) odnosi i (c) opći dodaci. 121
Slika 4.2.2 Definicija novog slučaja uporabe 122
Slika 4.2.3 Izgled novog slučaja uporabe 122
Slika 4.2.4 Definicija granice sustava modeliranog dijagramom slučajeva uporabe 123
Slika 4.2.5 Definicija novog aktora 123
Slika 4.2.6 Projektni preglednik 124
Slika 4.2.7 Definicija veze pridruživanja 125
Slika 4.2.8 Definiranje točke proširenja za vezu proširenja na dijagramu slučajeva uporabe 125
Slika 4.2.9 Umetanje bilješke na dijagram slučajeva uporabe 126
Slika 4.2.10 Alatni okviri dijagrama aktivnosti 127
Slika 4.2.11 Postavljenja particije (plivačke staze) na dijagram aktivnosti 128
Slika 4.2.12 Postavljanje početnog i završnog čvora na dijagramu aktivnosti 129
Slika 4.2.13 Postavljanje aktivnosti na dijagramu aktivnosti 130
Slika 4.2.14 Postavljanje račvanja i skupljanja na dijagramu aktivnosti 131
Slika 4.2.15 Prikaz usporednog izvođenja na dijagramu aktivnosti 131
Slika 4.2.16 Postavljanje uvjeta grananja na dijagramu aktivnosti 132
Slika 4.2.17 Alatni izbornici za izradu dijagrama slijeda 133
Slika 4.2.18 Postavljanje postojećeg aktora na dijagram slijeda 134
Slika 4.2.19 Nova životna linija postojećeg aktora u dijagramu slijeda 134
Slika 4.2.20 Postavljanje životne linije novog objekta u dijagram slijeda 135
Slika 4.2.21 Aktori i objekti sa svojim životnim linijama 136
Slika 4.2.22 Postavljanje elementa vrste Fragment u dijagram slijeda 136
Slika 4.2.23 Izgled elementa vrste ‘Fragment’ na dijagramu slijeda 137
Slika 4.2.24 Definicija uvjetnog grananja (alt) na dijagramu slijeda 137
Slika 4.2.25 Prikaz uvjetnog grananja na dijagramu slijeda 138
Slika 4.2.26 Nova poruka u dijagramu slijeda 139
Slika 4.2.27 Upis detalja poruke dijagrama slijeda 140
Slika 4.2.28 Pokretanje nove skupine poruka na dijagramu slijeda 141
Slika 4.2.29 Alatni okviri dijagrama komunikacije 142
Slika 4.2.30 Umetanje poruka između povezanih sudionika 143
Slika 4.2.31 Postupak automatske pretvorbe dijagrama slijeda u dijagram komunikacije 144
Slika 4.2.32 Pokretanje automatske pretvorbe dijagrama slijeda u dijagram komunikacije 144
Slika 4.2.33 Definicija nove skupine pobrojavanja poruka 145
Slika 4.2.34 Alatni okviri dijagrama razreda 146
Slika 4.2.35 Definicija novog razreda 147

Slika 4.2.36 Pristup definiciji atributa i metoda razreda 147
Slika 4.2.37 Definicija novog atributa razreda 148
Slika 4.2.38 Definicija novih metoda razreda 149
Slika 4.2.39 Prikaz novog razreda na dijagramu 149
Slika 4.2.40 Postavljanje polja Property za pojedini atribut razreda 150
Slika 4.2.41 Postavljanje postojećeg razreda u dijagram 151
Slika 4.2.42 Definicija stavki pobrojavanja 152
Slika 4.2.43 Pobrojavanje u dijagramu razreda 152
Slika 4.2.44 Postavljanje usmjerenja veze pridruživanja 153
Slika 4.2.45 Postavljanje višestrukosti veze pridruživanja 153
Slika 4.2.46 Postavljanje podrazumijevanog ciljnog jezika za generiranje izvornog koda 154
Slika 4.2.47 Poziv funkcionalnosti generiranja koda 155
Slika 4.2.48 Opcije pri generiranju novog programskog koda 156
Slika 4.2.49 Ponovno postavljanje jezika generiranja 156
Slika 4.2.50 Izbor definicije nove vrste podataka 157
Slika 4.2.51 Definicija nove vrste podataka 158
Slika 4.2.52 Dodavanje vrste podataka String za programski jezik Java 158
Slika 4.2.53 Pokretanje uvoza izvornog koda 159
Slika 4.2.54 Postavljanje uvjeta uvoza programskog koda 159
Slika 4.2.55 Usporedba strukture datoteka izvornog koda i stvorenih razreda 160
Slika 4.2.56 Kombiniranje razreda različitih paketa 161
Slika 4.2.57 Alatni okviri dijagrama objekata 162
Slika 4.2.58 Poziv opisa stanja objekta 163
Slika 4.2.59 Opis stanja objekta (Run States) 163
Slika 4.2.60 Alatni okviri dijagrama stanja 164
Slika 4.2.61 Opis unutarnjeg ponašanja nekog stanja 165
Slika 4.2.62 Opis unutarnjih okidača nekog stanja 165
Slika 4.2.63 Opis prijelaza između stanja 166
Slika 4.2.64 Alatni okviri dijagrama komponenti 167
Slika 4.2.65 Definicija sučelja na dijagramu komponenti 168
Slika 4.2.66 Prikaz sučelja komponente 168
Slika 4.2.67 Prikaz povezivanja zahtijevanog i ponuđenog sučelja komponenti 169
Slika 4.2.68 Pojednostavljeni prikaz povezivanja zahtijevanog i ponuđenog sučelja komponenti 169
Slika 4.2.69 Stvaranje podsustava na dijagramu komponenti 170
Slika 4.2.70 Definicija pristupne točke podsustava na dijagramu komponenti 171
Slika 4.2.71 Prikaz pristupnih točaka s vanjskim sučeljima podsustava na dijagramu komponenti 171
Slika 4.2.72 Povezivanje sučelja podsustava na dijagramu komponenti 172
Slika 4.2.73 Definicija novog stereotipa na dijagramu komponenti 173
Slika 4.2.74 Upotpunjeni prikaz podsustava 'Web Prodaja' na dijagramu komponenti 173
Slika 4.2.75 Postavljanje stereotipa za prošireni prikaz sučelja na dijagramu komponenti 174
Slika 4.2.76 Prošireni prikaz sučelja na dijagramu komponenti 175
Slika 4.2.77 Definiranje stereotipa za komponentu vrste Component 176
Slika 4.2.78 Definiranje stereotipa za komponentu vrste Artifact 176
Slika 4.2.79 Alatni okviri dijagrama razmještaja 177
Slika 4.2.80 Umetanje čvorova u dijagram razmještaja 178
Slika 4.2.81 Prikaz hijerarhije čvorova i komponenti na dijagramu razmještaja 178
Slika 4.2.82 Definicija komunikacijske veze na dijagramu razmještaja 179

Slika 4.2.83 Smještaj i prikaz artefakta komponente dijagrama razmještaja 180
Slika 4.2.84 Primjer povezivanja aplikacijskog dijela i baze podataka na dijagrama razmještaja 180
Slika 4.2.85 Definicija stereotipa komponente na dijagramu razmještaja 181
Slika 4.2.86 Definicija stereotipa komponente baze podataka na dijagramu razmještaja 182
Slika 4.2.87 Uvoz datoteke simbola.....183
Slika 4.2.88 Jednostavni prikaz čvorova na dijagramu razmješaja.....183
Slika 4.2.89 Izbor simbola iz uvezene biblioteke.....184
Slika 4.2.90 Pregled dostupnih simbola biblioteke 185
Slika 4.2.91 Simbolički prikaz čvorova na dijagramu razmještaja.....185
Slika 7.2.1 Rješenje zadatka dijagram slučajeva uporabe 193
Slika 7.2.2 Rješenje zadatka dijagrama razreda 195

Popis tablica

Tablica 2.3.1 Pregled slučajeva uporabe s opisima 23
Tablica 2.3.2 Pregled aktora s pripadajućim slučajevima uporabe.....25
Tablica 2.7.1 Pregled razina vidljivosti atributa i metoda razreda.....79

Uvodne napomene

Priručnik u prvom redu opisuje modeliranje programske potpore notacijom UML na temelju primjera jednog projektnog zadatka i sastavljen je kao nastavno pomagalo predmeta „Objektno orijentirani razvoj programa“ na diplomskom studiju informatike smjera računarstvo na Tehničkom veleučilištu u Zagrebu. U sklopu toga predmeta je prvenstveno namijenjen kao pomagalo u izradi laboratorijskih vježbi koje se sastoje u dokumentiranju konkretnog projekta i modeliranju programske potpore kao rezultata projekta. Iako pretpostavlja da je čitatelj upoznat s teoretskim dijelom po pitanju svrhe i notacije UML standarda, u kratkim crtama uvodi temelje jezika UML. Priručnik se ne ograničava svrhom vezanom uz tu materiju jer u kratkim crtama uvodi ideju modelno usmjerenog programskog inženjerstva kao jednog smjera ili modela procesa programskog inženjerstva.

Zadatak koji se obrađuje u priručniku zadan je kao ogledni primjer zadatka s mogućim stvarnim temeljem, izveden iz teksta jednog nasumično odabranog studentskog projektnog naslova i opisa, s ciljem poticanja studenata na kritičko razmišljanje o mogućem rješenju koje najprije treba modelirati i opisati notacijom UML prije i tijekom samog ostvarenja u nekoj od računarskih tehnologija, programskih alata i jezika. S obzirom na to da se prikazana rješenja temelje na opisu projektnog zadatka bez obavljenog procesa detaljnog izlučivanja zahtjeva, takva rješenja nisu jedina ispravna već su prvenstveno namijenjena prikazu načina modeliranja i dokumentiranja, te mogućih pristupa razvoja i načela razmišljanja uobličenih u rješenja UML-dijagrama.

Iako se tekst priručnika nastoji ne vezati posebno niti uz jedan CASE (engl. *Computer-Aided Software Engineering*) alat za modeliranje, u posebnom poglavlju se izlažu neke od karakteristika alata *Enterprise Architect* jer su primjeri preneseni iz tog okruženja. Priručnik je u temeljnom dijelu zamišljen neovisno o alatu za modeliranje a u navedenom poglavlju je naglasak stavljen na uporabu konkretnog alata.

Osim alata *Enterprise Architect* postoji i niz drugih CASE alata, komercijalnih i otvorenog pristupa, kojima se može modelirati sustav programske potpore na manje ili više sveobuhvatan način. Neki od alata imaju mogućnost pokriti veći dio projektnih faza izrade programske potpore a neki njih od njih služe samo kao grafičko sučelje za izradu UML-dijagrama. S obzirom na to da su studenti u praksi koristili cijeli niz različitih alata za modeliranje po svom izboru to bi mogla biti posebna tema za izlaganje, koja se ovdje ne razrađuje.

Alat *Enterprise Architect* prethodno je predstavljen izdanjem priručnika *Enterprise Architect, Verzije 11 i 12* [23], te izdanjem priručnika *Programsko inženjerstvo i informacijski sustavi* autora Edmonda Krushe i Željka Kovačevića koji pretpostavlja uporabu izdanja *Enterprise Architect 15*, [24], kao i službena dokumentacija ovih alata dostupna na web sjedištu tvrtke *SparxSystems*, [27].

1. Programsko inženjerstvo i modeliranje procesa

Disciplina programskog inženjerstva se bavi metodama i alatima za profesionalno oblikovanje i proizvodnju programske potpore (engl. *software*) na uređen i mjerljiv način i dio je svakog studijskog programa računarstva ili informatike. Za razliku od računalnog inženjerstva koje se općenitije bavi sustavima zasnovanima na računalima, što uključuje i inženjerstvo sklopovlja (engl. *hardware*), programsko inženjerstvo usredotočeno je na razvoj programske infrastrukture odnosno primjenu inženjerstva kao sustavnog, uređenog i mjerljivog pristupa programskoj potpori. Osim poznavanja temelja računarske znanosti i prakse izrade proizvoda programske potpore to uključuje i inženjersku ekonomiku odnosno učinkovitost proizvodnje u smislu troškova izrade. U odnosu na računarsku znanost koja se bavi teorijskim osnovama računarstva, računarskih sustava i aplikacija, programsko inženjerstvo se bavi praktičnim problemima razvoja, oblikovanja i isporuke korisnog računalnog programa.

Različiti pristupi razvoja programske potpore kao jednoj takvoj potpuno zasebnoj disciplini su se oblikovali kao različiti modeli procesa programskog inženjerstva. Ti modeli su se razvijali kroz vrijeme usporedno s razvitkom mogućnosti računarske tehnologije uz napretke u praksi pristupa programskog inženjerstva. U početcima se najprije razvio vodopadni model (engl. *waterfall model*) koji je obilježen strogo odvojenim fazama razvoja koje se slijedno oslanjaju jedna na drugu. S druge strane, danas je aktualna agilna metodologija razvoja (engl. *agile development*) i njezinim različitim izvedbama, gdje su aktivnosti razvoja vremenski isprepleteni s iterativnim pristupom razvoju. Osim tih, kao temeljni modeli navode se evolucijski model (engl. *evolutionary model*) koji se zasniva na inkrementalnom pristupu čime uvodi više iteracija i čini osnovu agilnog razvoja te komponento usmjereni model (engl. *component-based model*) koji je motiviran mogućnošću ponovne uporabe postojećih prethodno razvijenih komponenti.

Modelno usmjereni razvoj (engl. *model-driven development*) predstavlja ideju temeljenu na oblikovanju uporabom modela. Uvođenje inženjerskog pristupa razvoju ogleda se u pokušajima definiranja odgovarajućih standarda kojima bi se postiglo uređenje pravilnosti u oblikovanju modela iz kojih će nastati gotovi proizvodi i u dokumentiranju rezultata na razumljiv način. Grafička notacija pri inženjerskom oblikovanju je neizbježna i ključna u razmjeni i prezentaciji ideja i rezultata te tome teži jezik za oblikovanje naziva *Unified Modeling Language* (UML) za koji su tijekom godina razvijeni brojni alati programske potpore za potporu uporabi notacije UML.

U drugom poglavlju ovog rada se daju uvodne napomene o jeziku UML nakon čega se predstavlja primjer opisa projektnih zahtjeva koji se elaborira u smislu zahtjeva i mogućnosti notacije UML. U većem dijelu poglavlja se daje model opisa specifikacije i implementacije sustava uporabom odabranog skupa UML-dijagrama.

U trećem poglavlju se daje kratak osvrt na modelno usmjereno programsko inženjerstvo kroz nekoliko definicija i pogleda na taj pristup razvoja programske potpore, te pregled jezika i alata za modeliranje.

U četvrtom poglavlju se daju osnovne informacije i upute za modeliranje uporabom alata *Enterprise Architect*, koji je namijenjen za modeliranje i potporu poslovnim procesima izrade programske potpore, kroz primjere UML-dijagrama koji pokrivaju bitne elemente prikaza modeliranog projektnog zadatka.

U završnih poglavljima se daju zaključak na temu poduzetog pristupa i rezultata modeliranja i dokumentiranja projektnog zadatka, popis literature i dodaci tekstu.

2. Primjer modeliranja uporabom jezika UML

U ovom poglavlju se daju kratak uvod u jezik UML, daje se opis projektnog zadatka programske potpore koji se potom kao ogledni primjer u nizu daljnjih poglavlja modelira UML notacijom.

2.1. Osnove UML dijagrama

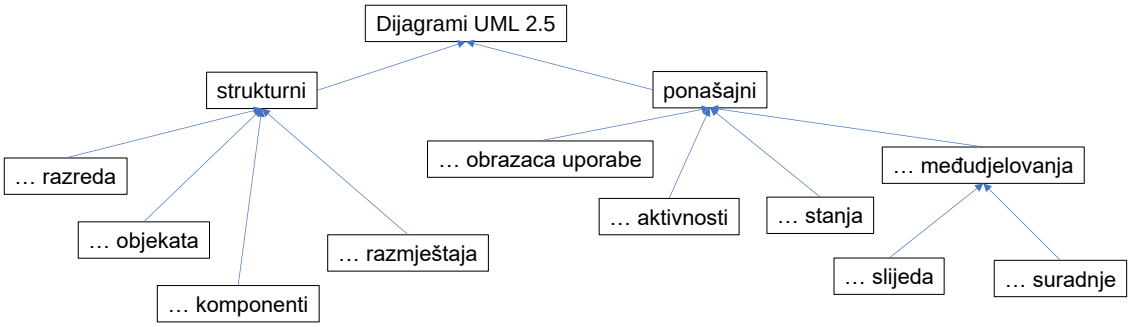
Za modeliranje i dokumentiranje koristi se niz dijagrama notacije UML. Važeći standard UML u trenutku pisanja ovog nastavnog materijala je inačica UML 2.5.1 koji uključuje niz dijagrama koji se u osnovi mogu podijeliti na ponašajne i strukturne.
Ponašajni dijagramu su sljedeći:

- dijagram slučajeva uporabe (engl. *use case diagram*),
- dijagram aktivnosti (engl. *acitivity diagram*),
- dijagram stanja (engl. *state machine diagram*),
- dijagrami međudjelovanja (engl. *interaction diagrams*):
    - dijagram slijeda (engl. *sequence diagram*),
    - dijagram komunikacije (engl. *communication diagram*),
    - dijagram vremena (engl. *timing diagram*),
    - dijagram pregleda interakcije (engl. *interaction overview diagram*).

Među strukturne dijagrame spadaju:

- dijagram razreda (engl. *class diagram*),
- dijagram objekata (engl. *object diagram*),
- dijagram paketa (engl. *package diagram*),
- dijagram složene strukture (engl. *composite structure diagram*),
- dijagram komponenti (engl. *component diagram*),
- dijagram razmještaja (engl. *deployment diagram*),
- dijagram profila (engl. *profile diagram*).

Između njih se može izdvojiti devet češće korištenih vrsta dijagrama o kojima će se govoriti i u ovom priručniku. To su ponašajni dijagrami: dijagram slučajeva uporabe, dijagram aktivnosti, dijagram stanja, dijagram slijeda i dijagram komunikacije te strukturni dijagrami: dijagram razreda, dijagram objekata, dijagram komponenti i dijagram razmještaja, Slika 2.1.1.



Slika 2.1.1 Pregled glavnih UML dijagrama unutar standarda 2.5.1.

Terminologija korištena u nazivima dijagrama i ključnih pojmova u literaturi na hrvatskom jeziku se ponešto razlikuje ovisno o izvoru, što će biti naznačeno navodom različitih naziva po poglavljima s opisima dijagrama.
Ogledni primjer zadatka koji je zadan opisan je UML notacijom. U prvom dijelu tog opisa više se radi o modeliranju problema i rezultatima izlučivanja zahtjeva a u drugom dijelu o dokumentiranju implementacije.

2.2. Opis projekta

Cilj projektnog zadatka je izrada web stranice za narudžbu auto-dijelova. Korisnici bi trebali imati na raspolaganju pregled ponude proizvoda auto-dijelova, a za mogućnost kupnje bi se najprije trebali registrirati u sustavu te nakon toga biti prijavljeni da bi mogli upravljati svojim narudžbama različitih proizvoda. Administratori i prodavači bi trebali imati na raspolaganju uvid u registrirane korisnike i njihove narudžbe te mogućnost analize korisničkih zahtjeva na temelju koje bi trgovci radili narudžbe i nabavu novih artikala proizvoda u skladište. Osim toga, trgovcima bi trebao biti omogućen uvid u najprodavanije proizvode.

Cilj koji se želi postići ovom aplikacijom je učinkovita pretraga auto-dijelova za kupce i odgovarajući uvid u stanje na skladištu za trgovce. Kupovina bi se trebala obavljati stavljanjem izabranih proizvoda u „košaricu“ i potvrdom narudžbe. Prodavači bi trebali moći proizvoljno definirati različite popuste pri kupnji proizvoda, npr. kod određenog iznosa računa. Podrazumijeva se da kupac račun za postavljenu narudžbu plaća kreditnom karticom.

Korisnici bi također trebali imati na raspolaganju sustav korisničke podrške s mogućnostima slanja upita na službeni kontakt e-pošte te slanja povratnih informacija o pruženoj ponudi i uslugama.

Administratori bi prilikom rada aplikacije trebali imati mogućnost ažuriranja odnosno osvježavanja stanja dostupnih proizvoda u bazi podataka što podrazumijeva dodavanje,

brisanje kao i promjene cijene pojedinih proizvoda te oblikovanje izvješća s povratnim informacijama dobivenim od strane korisnika.

Redovito bi se trebala pokretati sigurnosna pohrana podataka spremljenih u bazi podataka: dnevno u lokalnu pohranu i tjedno u vanjsko spremište podataka.

Tekst zadatka je napisan slobodnim, neformalnim opisom, na razini primjera korisničkih zahtjeva.

Na temelju iščitavanja teksta prepoznaju se potrebne funkcionalnosti koje je potrebno ostvariti. U smislu modeliranja UML-om i kasnijeg programskog ostvarenja nužno je prepoznati i definirati ključne entitete koji će činiti okosnicu buduće arhitekture programske potpore. Na taj način će se najprije oblikovati UML-dijagrami, a prema njima i programski kôd proizvoda.

2.3. Dijagram slučajeva uporabe

UML-dijagrami slučajeva uporabe ili dijagrami slučajeva korištenja (engl. *use case diagrams*) opisuju funkcionalne zahtjeve sustava i služe prikazu namjene sustava za vanjskog korisnika. To su dijagrami ponašajne (engl. *behavioral*) naravi, opisuju način međudjelovanja vanjskog svijeta sa zatvorenim sustavom programskog proizvoda. Sastoje se od **aktora** i **slučajeva uporabe**. Pri tome je važno navesti **granicu sustava** koja odvaja zatvoreni sustav programskog proizvoda od vanjskog svijeta. Aktori predstavljaju korisnike sustava a način korištenja sustava se oblikuje putem navoda pojedinih funkcionalnosti koje se u kontekstu dijagrama nazivaju slučajevi uporabe. To su tri temeljna elementa dijagrama koji imaju sljedeću notaciju:

- Aktor (engl. *actor*) se na dijagramu prikazuje kao „čovječuljak“ ili *stickman* s nazivom ispod a aktori koji nisu ljudski entiteti već dijelovi sustava mogu se prikazati običnom pravokutnom notacijom. Dobro ih je odmah pozicionirati uz slučajeve uporabe uz koje su vezani da bi dijagram u konačnici bio pregledniji.
- Slučajevi uporabe (engl. *use cases*) se prikazuju kao ovalni oblici s nazivom odnosno kratkim navodom kojim bi se trebalo jasno opisati funkcionalnost koju predstavljaju,
- Granica sustava (engl. *system boundary*) se prikazuje pravokutnikom koji unutar sebe obuhvaća sve slučajeve uporabe dok se aktori nalaze izvan nje.

Dijagrami bi trebali biti dovoljno jasni da čitatelj može shvatiti kako se sustav može upotrijebiti. Dijagram govori koji su načini, obrasci ili konkretni slučajevi uporabe sustava i koji su odgovori sustava prilikom pristupa izvana. U svrhu postizanja dovoljne jasnoće dijagram može sadržavati i **bilješke** (engl. *notes*) kao neke opće elementi koji su primjenjivi u većini UML dijagrama.

Aktori

Analizom opisa projektnog zadatka u prvom koraku odnosno prolasku kroz tekst projektnog zadatka može se prepoznati više vrsta korisnika sustava kao aktora ili sudionika. Potencijalni aktori su prilikom prolaska kroz tekst označeni zelenom bojom.

Cilj projektnog zadatka je izrada web stranice za narudžbu auto-dijelova. Korisnici bi trebali imati na raspolaganju pregled ponude proizvoda auto-dijelova, a za mogućnost kupnje bi se najprije trebali registrirati u sustavu te nakon toga biti prijavljeni da bi mogli upravljati svojim narudžbama različitih proizvoda. Administratori i trgovci bi trebali imati na raspolaganju uvid u registrirane korisnike i njihove narudžbe te mogućnost analize korisničkih zahtjeva na temelju koje bi trgovci radili narudžbe i nabavu novih artikala i proizvoda u skladište. Osim toga, trgovcima bi trebao biti omogućen uvid u najprodavanije proizvode.

Cilj koji se želi postići ovom aplikacijom je učinkovita pretraga auto-dijelova za kupce i odgovarajući uvid u stanje na skladištu za trgovce. Kupovina bi se trebala obavljati stavljanjem izabranih proizvoda u „košaricu“ i potvrdom narudžbe. Trgovci bi trebali moći proizvoljno definirati različite popuste pri kupnji proizvoda, npr. kod određenog iznosa računa. Podrazumijeva se da kupac račun za postavljenu narudžbu plaća kreditnom karticom.

Korisnici bi također trebali imati na raspolaganju sustav korisničke podrške s mogućnostima slanja upita na službeni kontakt e-pošte te slanja povratnih informacija o pruženoj ponudi i uslugama.

Administratori bi prilikom rada aplikacije trebali imati mogućnost ažuriranja odnosno osvježavanja stanja dostupnih proizvoda u bazi podataka što podrazumijeva dodavanje, brisanje kao i promjene cijene pojedinih proizvoda te oblikovanje izvješća s povratnim informacijama dobivenim od strane korisnika.

Redovito bi se trebala pokretati sigurnosna pohrana podataka spremljenih u bazi podataka: dnevno u lokalnu pohranu i tjedno u vanjsko spremište podataka.

Prepoznaje se nekoliko različitih vrsta korisnika ili aktora:

- Korisnici
- Administratori
- Trgovci
- Registrirani korisnici
- Vanjsko spremište podataka

U pravilu je prije dodavanja aktora u dijagram potrebno postaviti granicu sustava koja odvaja predviđeni ili ostvareni skup slučajeva uporabe od aktora, tj. naglašava fizičku odvojenost programskog proizvoda i vanjskog svijeta i predstavlja logičko sučelje između korisnika i sustava. Aktori se uvijek nalaze izvan granice sustava. Unutar prikaza granice sustava definira se i naziv sustava koji bi trebao biti programski ostvaren a koji je vanjskom svijetu vidljiv kroz skup ponuđenih slučajeva uporabe koji se nalaze unutar granice sustava. Osim toga, dobro je i pogledati podjelu i uređenje slučajeva uporabe prema vrstama korisnika s kojima su povezani te ih tako pozicionirati da bi dijagram bio što jednostavniji za razumijevanje, posebno s obzirom na to da neke od slučajeva uporabe koristi više korisnika te se mogu razdvojiti i posložiti po vrstama korisnika. Konkretnije će to biti prikazano u primjerima modeliranja koji slijede nakon raščlambe povezivanja aktora i slučajeva uporabe.

Slučajevi uporabe

Daljnjom analizom opisa projekta trebaju se prepoznati tražene funkcionalnosti što se najlakše postiže pregledom mogućih akcija aktora u sustavu. Te funkcionalnosti postaju slučajevi uporabe ciljanog programskog proizvoda. Prilikom prolaska kroz tekst prepoznate funkcionalnosti su označene žutom bojom.

Cilj projektnog zadatka je izrada web stranice za narudžbu auto-dijelova. Korisnici bi trebali imati na raspolaganju pregled ponude proizvoda auto-dijelova, a za mogućnost kupnje bi se najprije trebali registrirati u sustavu te nakon toga biti prijavljeni da bi mogli upravljati svojim narudžbama različitih proizvoda. Administratori i trgovci bi trebali imati na raspolaganju uvid u registrirane korisnike i njihove narudžbe te mogućnost analize korisničkih zahtjeva na temelju koje bi trgovci radili narudžbe i nabavu novih artikala i proizvoda u skladište. Osim toga, trgovcima bi trebao biti omogućen uvid u najprodavanije proizvode.

Cilj koji se želi postići ovom aplikacijom je učinkovita pretraga auto-dijelova za kupce i odgovarajući uvid u stanje na skladištu za trgovce. Kupovina bi se trebala obavljati stavljanjem izabranih proizvoda u „košaricu“ i potvrdom narudžbe. Trgovci bi trebali moći proizvoljno definirati različite popuste pri kupnji proizvoda, npr. kod određenog iznosa računa. Podrazumijeva se da kupac račun za postavljenu narudžbu plaća kreditnom karticom.

Korisnici bi također trebali imati na raspolaganju sustav korisničke podrške s mogućnostima slanja upita na službeni kontakt e-pošte te slanja povratnih informacija o pruženoj ponudi i uslugama.

Administratori bi prilikom rada aplikacije trebali imati mogućnost ažuriranja odnosno osvježavanja stanja dostupnih proizvoda u bazi podataka što podrazumijeva dodavanje, brisanje kao i promjene cijene pojedinih proizvoda te oblikovanje izvješća s povratnim informacijama dobivenim od strane korisnika.

Redovito bi se trebala pokretati sigurnosna pohrana podataka spremljenih u bazi podataka: dnevno u lokalnu pohranu i tjedno u vanjsko spremište podataka.

Temeljem analize teksta opisa projekta u dijagram je stoga potrebno dodati odgovarajući skup slučajeva uporabe, Tablica 2.3.1. Slučajevi uporabe su ujedno prikazanim redoslijedom i pobrojani te im je dodijeljen i skraćeni naziv oblika *UC*<n> gdje je <*n*> dodijeljeni redni broj slučaja uporabe radi lakšeg daljnjeg referenciranja u tekstu i slikama.

Tablica 2.3.1 Pregled slučajeva uporabe s opisima

Oznaka	***Naziv***	***Opis***
UC1	Pretraga proizvoda	Gost ili kupac može pretraživati ponuđene proizvode prema ponuđenom sadržaju izbornika ili po ključnim riječima.
UC2	Pregled proizvoda	Gost ili kupac može gledati detalje proizvoda koji su navedeni kao opis proizvoda.
UC3	Slanje upita	Gost ili kupac mogu slati upit radi dobivanje dodatnih informacija o proizvodima.
UC4	Registracija korisnika	Gost se može registrirati i time steći pravo kupovine u web trgovini.
UC5	Prijava u sustav	Jednom registrirani korisnik se može prijaviti u aplikaciju i stavljati proizvode u vlastitu košaricu.
UC6	Izbor proizvoda	Proizvod se može staviti u košaricu radi daljnje narudžbe, odnosno kupnje.
UC7	Kupnja proizvoda	Kupnja proizvoda je moguća svih registriranim korisnicima.
UC8	Ostvarivanje popusta	Popust se pri kupnji ostvaruje ako su ispunjeni prethodno definirani i objavljeni uvjeti od strane trgovca.
UC9	Slanje povratne informacije	Kupac može poslati povratnu informaciju trgovcu i izraziti (ne)zadovoljstvu pruženom uslugom, dati komentar ili prijedlog za poboljšanje usluge.
UC10	Korisnička podrška	Trgovac bi u pravilu trebao dati odgovor na upite i komentare korisnika aplikacije.
UC11	Uvid u korisničke narudžbe	Trgovac bi trebao imati uvid u pojedine korisničke narudžbe i daljnje potrebe.
UC12	Uvid u stanje na skladištu	Trgovac treba imati uvid u stanje pojedinih artikala koji su dostupni na skladištu radi zadovoljenja korisničkih potreba.
UC13	Narudžba novih proizvoda	Trgovac treba imati mogućnost narudžbe novih proizvoda i artikala prema uvidu o stvarnim potrebama kupaca i stanja na skladištu.
UC14	Analiza korisničkih zahtjeva	Sustav bi trgovcima trebao dati i napredniju analizu potreba te prepoznate uzorke i statistiku naručivanja
UC15	Definiranje popusta	Trgovac bi trebao moći definirati uvjete ostvarivanja popusta na vrstu i količinu proizvoda te iznose računa za naplatu.
UC16	Uvid u statistiku prodaje	Administrator i trgovac bi trebali imati uvid u statistiku prodaje pojedinih proizvoda i zarade u određenim vremenskim razdobljima ovisno o načinu ponude i omogućenih popusta.
UC17	Oblikovanje izvješća	Administrator bi trebao biti u mogućnosti oblikovati izvješća sa sadržajima prema zahtjevima trgovca na temelju podataka pohranjenih u bazi podataka.
UC18	Osvježavanje baze podataka	Administrator treba redovito ažurirati podatke o proizvodima i njihovim cijenama u bazi podataka: dopunjavati, mijenjati i brisati.
UC19	Pohrana podataka	Podaci se redovito, na dnevnoj i tjednoj bazi, trebaju pohranjivati u vanjsko spremište podataka.

Svaki slučaj uporabe, osim rednog broja, ima naziv koji će se upotrijebiti na dijagramu kao sadržaj notacije ovalnog oblika. U kratkom opisu svakog slučaja uporabe je iz teksta opisa projekta izlučen kontekst odgovarajuće funkcionalnosti: na kojeg se korisnika i na koji način se odnosi.

Iz toga je vidljivo koje vrste aktora su vezani uz pojedine funkcionalnosti definirane kao slučajevi uporabe, te njihov način uporabe u budućem programskom proizvodu. To su vanjski korisnici koji mogu biti registrirani u sustav i koji započinju neke akcije s aplikacijom, trgovci kojima aplikacija olakšava posao i administratori koji podržavaju temeljne funkcionalnosti aplikacije i koji također započinju određene akcije koje su im omogućene aplikacijom. Svi oni se mogu svrstati pod skupinu **aktivnih aktora** ili inicijatora koji započinju određene akcije.

Vidljivo je iz opisa projektnog zadatka da se između više vrsta predviđenih korisnika sustava razlikuju oni koji se nalaze na strani kupca i oni koji se nalaze na strani trgovca. Tako na strani kupaca postoji korisnik 'Gost' koji je neregistriran i može samo pregledavati ponudu proizvoda i korisnik 'Kupac' koji se registrirao i može kupovati proizvode, tj. upravljati svojim narudžbama. Isto tako, na strani prodavača postoji korisnik 'Administrator' koji upravlja informacijskim sustavom, ali i korisnik 'Trgovac' koji je stručna osoba u domeni prodaje auto dijelova i koja zaprima i odgovara na upite kupaca te naručuje nove auto-dijelove.

U svakom slučaju, mogu se povezati pojedini slučajeve uporabe s pojedinim vrstama korisnika ili aktora, Tablica 2.3.2.

Uz element aktora se često u dokumentaciji koristi i pojam **uloge** (engl. *role*) koju pojedini aktor u sustavu ima i koja je dana u dodatnom opisu te se svaki aktor veže uz niz prethodno definiranih slučajeva uporabe.

Nakon postavljanja slučajeva uporabe i aktora u dijagram potrebno ih je povezati logikom koja je zadana specifikacijom projektnog zadatka kako bi vjerodostojno pokazivali njihove odnose.

Četiri su vrste veze ili odnosa koji će se koristiti u ovom primjeru su one koje se i najčešće koriste u oblikovanju dijagrama slučajeva uporabe:

- odnos **pridruživanja** ili asocijacije (engl. *association*),
- odnos **poopćenja** ili generalizacije (engl. *generalize*),
- odnos **obuhvaćanja** ili uključivanja, (engl. *include*),
- odnos **proširenja** (engl. *extend*).

Tablica 2.3.2 Pregled aktora s pripadajućim slučajevima uporabe

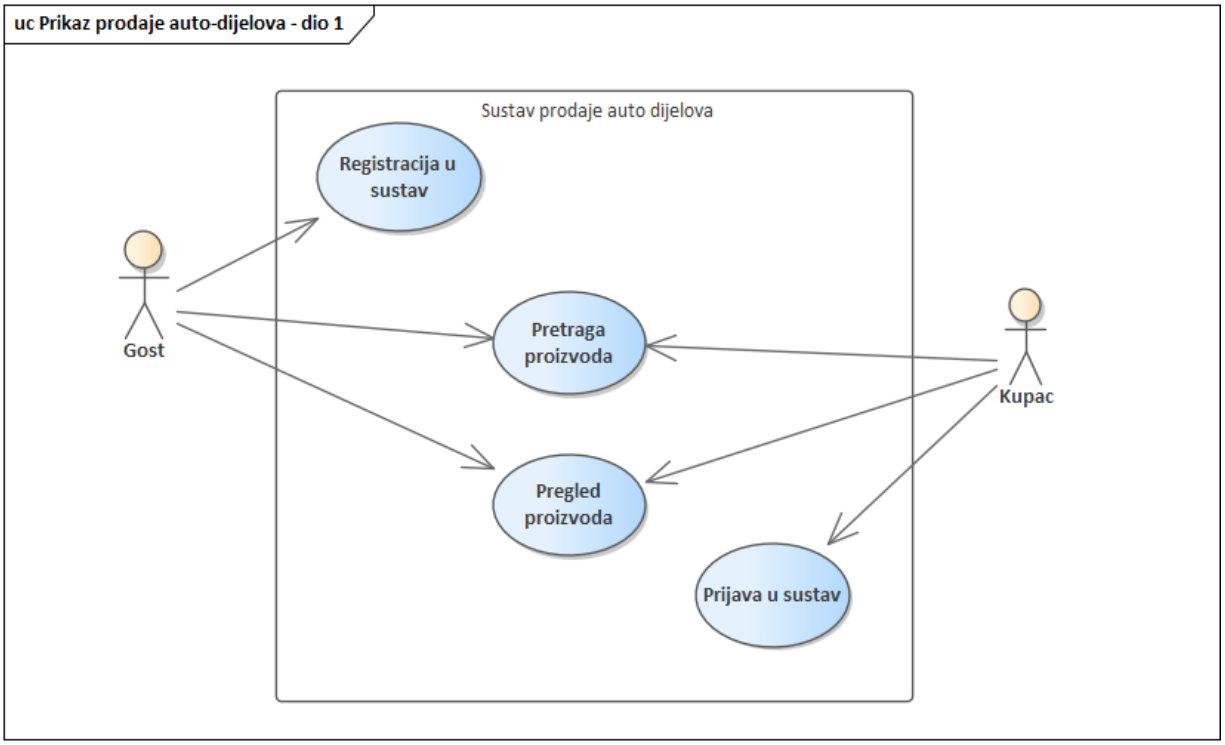
R. br.	*Aktor*	*Opis uloge*	*UC br.*
1	**Gost/Neregistrirani korisnik**	Gost može pretraživati i pregledavati ponudu proizvoda, postavljati upite o njima te se registrirati nakon čega postaje registrirani korisnik.	UC1 UC2 UC3 UC4
2	**Kupac/Registrirani korisnik**	Registrirani korisnik može raditi s proizvodima sve što može i neregistrirani korisnik te, nakon što se prijavi, raditi i što je vezano uz narudžbe proizvoda kao i slati povratne informacije o zadovoljstvu pruženim uslugama.	UC1 UC2 UC3 UC5 UC6 UC7 UC8 UC9
3	**Trgovac/Prodavač**	Trgovac je osoba koja je osposobljena i ovlaštena za prodaju auto-dijelova te specifično može obavljati poslove vezane uz pružanje korisničke podrške, narudžbe novih proizvoda, evidenciju stanja i statistike kao i definiranje popusta.	UC10 UC11 UC12 UC13 UC14 UC15 UC16
4	**Administrator**	Administrator prvenstvo obavlja poslove vezane uz održavanje informacijskog sustava i web-aplikacije što uključuje i upravljanje sustavom baze podataka.	UC11 UC16 UC17 UC18
5	**Spremište podataka**	Mjesto vanjske pohrane podataka na dnevnoj i tjednoj bazi.	UC19

Najprije će se nešto reći o odnosu pridruživanja kao o temeljnom odnosu na dijagramu odnosno povezivanju aktora i slučajeva uporabe nakon čega će se predstaviti i ostale vrste odnosa koje mogu postojati između aktora i između slučajeva uporabe.

Odnos pridruživanja

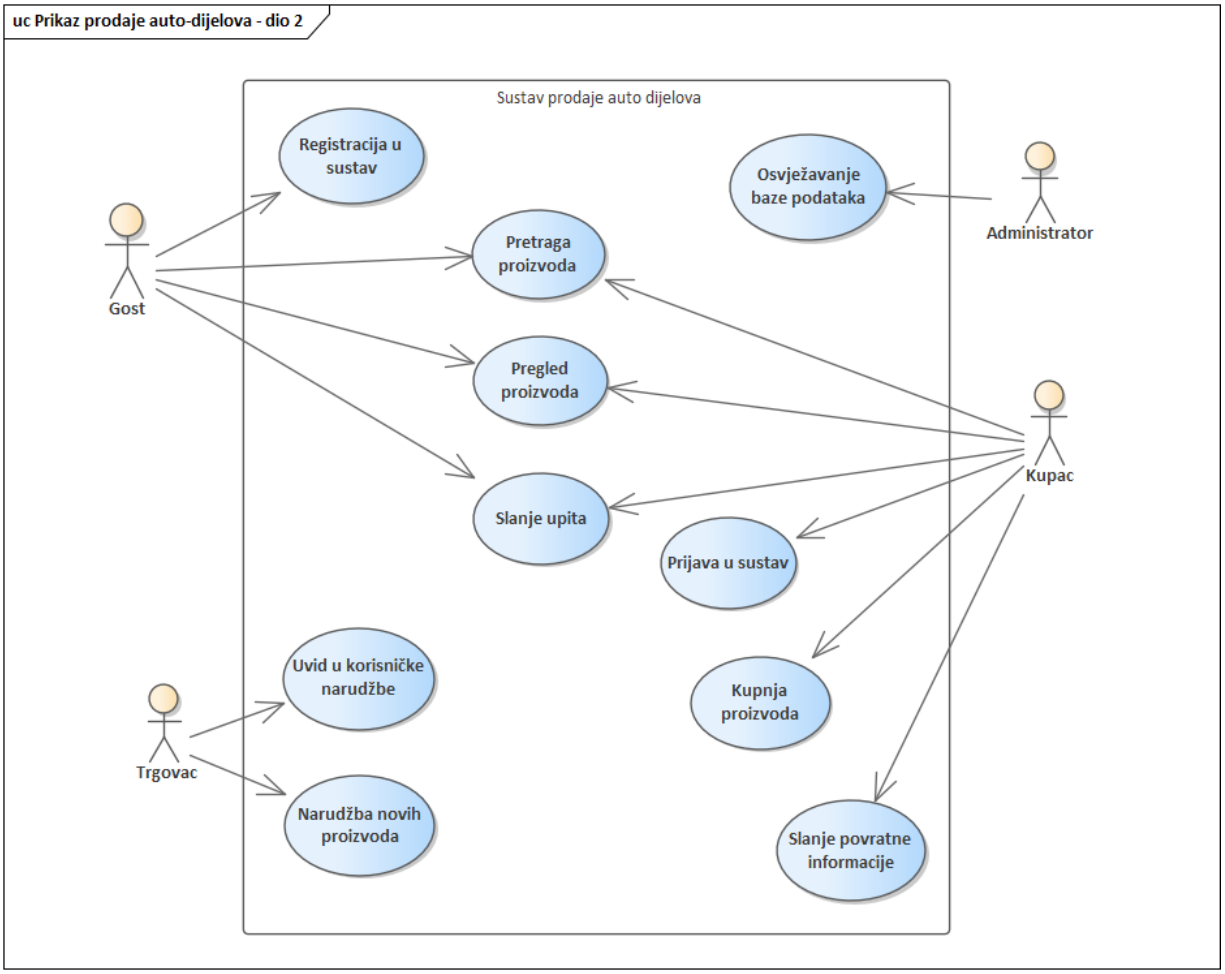
Vezom pridruživanja ili asocijacijom povezuju se aktori i slučajevi uporabe, te se označava da neki aktor pokreće ili sudjeluje u pojedinom slučaju uporabe. Ova veza može imati naglašeno usmjerenje: od onog tko započinje odnos ili komunikaciju prema onome tko je prihvaća. Na dijagramu je to prikazano usmjerenjem linije od strane inicijatora neke akcije prema onom drugom i označena je punom jednostavnom linijom s mogućom strelicom koja pokazuje usmjerenje.
Slika 2.3.1 prikazuje primjer jednostavnog dijagrama slučaja uporabe koji prikazuje povezivanje aktora Gost i Kupac sa slučajevima uporabe UC1, UC2, UC4 i UC5. Navedeni slučajevi

uporabe smješteni su unutar naznačene granice sustava zajedno sa zadanim nazivom dijagrama dok se aktori nalaze izvan granica sustava.



Slika 2.3.1 Primjer jednostavnog dijagrama slučajeva uporabe s vezama pridruživanja

I gost i kupac su inicijatori pojedinih akcija pa veza pridruživanja ima odgovarajuće usmjerenje. Vidljivo je da i Gost i Kupac mogu raditi pretragu i pregled proizvoda. Gost se može registrirati u sustavu čime zadobiva status Kupca koji se potom može prijaviti u sustav. Osnovna notacija i semantika dijagrama je u osnovi jednostavna, nema mogućnosti prikaza napredne semantike kao što bi bila mogućnost izričitog prikaza načina redoslijeda akcija niti načina kako Gost postaje Kupac već se to eventualno može staviti u bilješku dijagrama.
Na isti način se postavljaju i ostali slučajevi uporabe, i za aktore 'Trgovac' i 'Administrator', zajedno s vezama pridruživanja prema aktorima. Svi slučajevi uporabe se nalaze unutar, a aktori izvan granica sustava, Slika 2.3.2.



Slika 2.3.2 Primjer dijagrama slučajeva uporabe s vezama pridruživanja

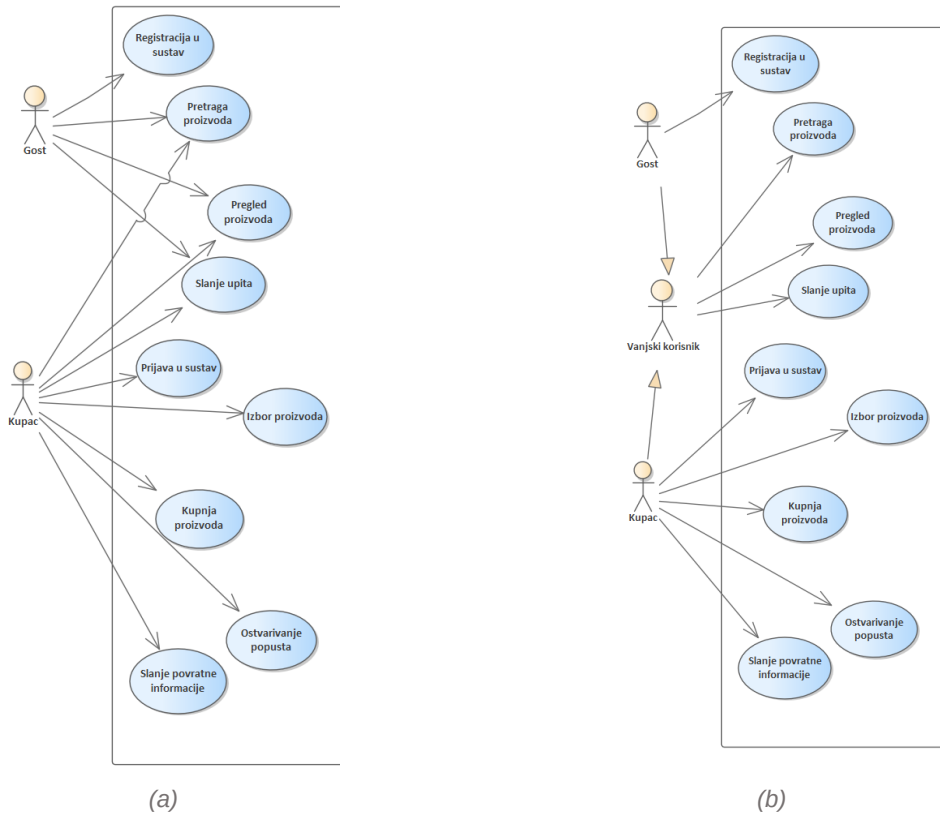
Slika 2.3.2 prikazuje povezanost aktora 'Gost' i 'Kupac' s njihovim slučajevima uporabe te je moguće vidjeti na koji način ove dvije vrste korisnika mogu koristiti predviđeni sustav prodaje auto-dijelova.
Može se primijetiti da postoje slučajevi uporabe koje koriste obje vrste aktora, i 'Gost' i 'Kupac'. To su slučajevi uporabe 'Pretraga proizvoda', 'Pregled proizvoda' i 'Slanje upita' te su prikazane odgovarajuće veze pridruživanja za svakog od tih aktora posebno. U slučaju dodavanja većeg broja elemenata u dijagram može nastati problem čitljivosti i razumljivosti dijagrama ako se brojne veze isprepliću. Kako je već spomenuto, dobro je smjestiti bliže jedne drugima aktore koji dijele isti skup funkcionalnosti te je moguće je u tom smislu unaprijediti preglednost prikaz primjenom odnosa poopćenja između tako srodnih aktora.

Ostale vrste odnosa

Odnos poopćenja

Veza poopćenja (engl. *generalization*) se može definirati između aktora međusobno te između slučajeva uporabe međusobno. Time se prikazuje da je neki aktor općenitijeg tipa i da se iz njega izvode specijalni slučajevi aktora odnosno njegove **specijalizacije**. Isto tako, iz nekog općenitijeg slučaja uporabe mogu se izvesti njegove specijalizacije slučajeva uporabe. Odnos se označava punom crtom s trokutastom strelicom na kraju u smjeru od specijalnog prema općenitijem entitetu, bilo da se radi o aktorima ili slučajevima uporabe.

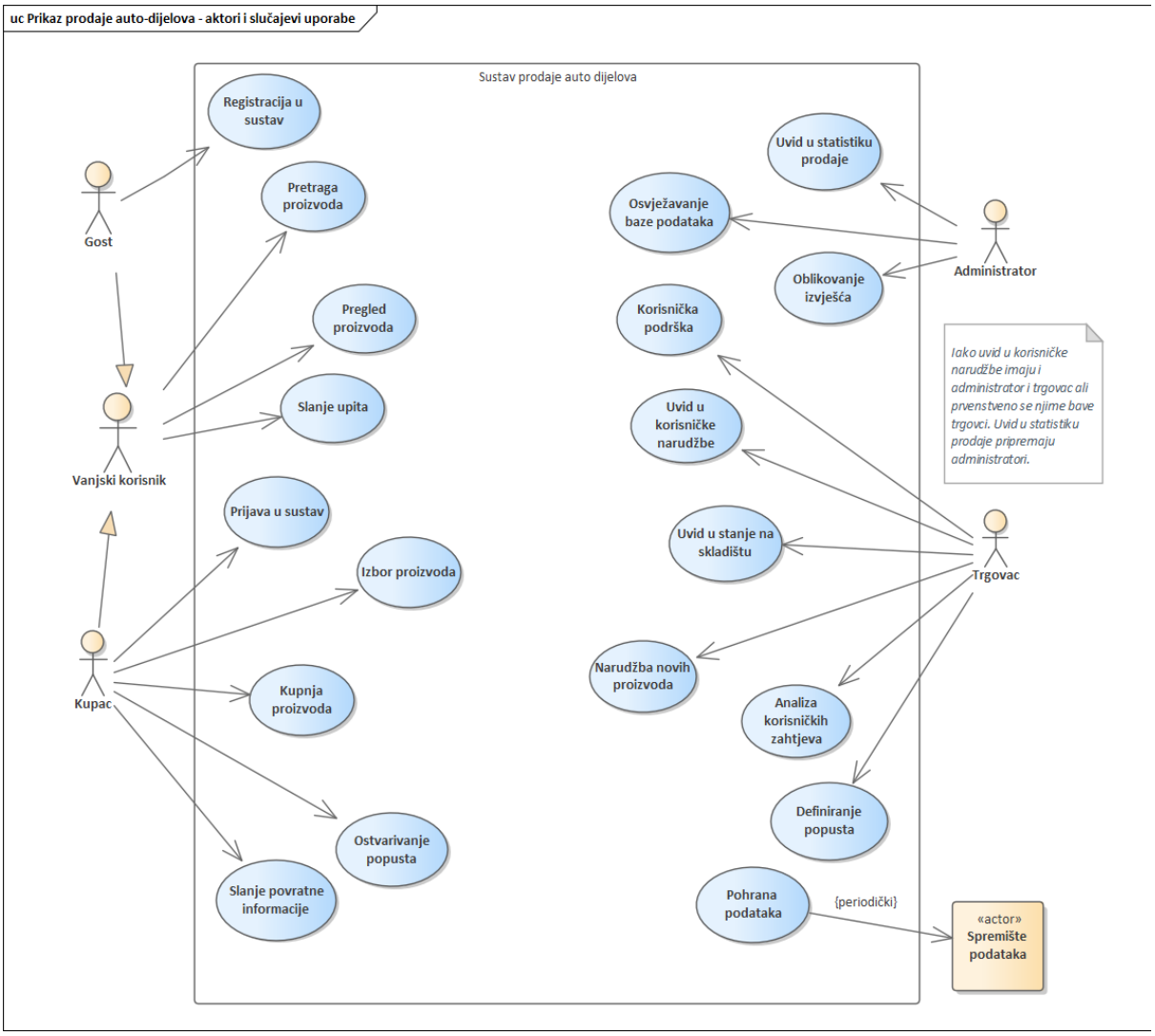
U primjeru dijagrama iz prethodnog poglavlja aktori 'Gost' i 'Kupac' imaju neki zajednički skup slučajeva uporabe s kojima su povezani pa je moguće iskoristiti prikaz odnosa poopćenja nad ovim dvjema vrstama korisnika. I gost i kupac su vrste korisnika koje u konačnom ostvarenju mogu biti izvedene iz općenitijeg entiteta, aktora vrste 'Vanjski korisnik' kojemu 'Gost' i 'Kupac' predstavljaju posebne slučajeve (specijalizaciju). Prema tome aktor 'Korisnik' je poopćenje aktora 'Gost' i 'Kupac' pa se to može naznačiti na dijagramu vezom poopćenja. Ako su zato oba aktora smješteni blisko na istoj strani dijagrama sa svim svojim pripadajućim slučajevima uporabe može se dobiti pojednostavljenje prikazanih veza pridruživanja na način da su za zajednička tri slučaja uporabe aktora 'Gost' i 'Kupac' one stavljene između općenitijeg aktora 'Korisnik' i zajedničkih slučajeva uporabe. Preostale veze pridruživanja ostaju prikazane podijeljeno između aktora: one koje postoje između aktora 'Gost' i pojedinih slučajeva uporabe naznačuju da ih samo 'Gost' može koristiti, a isto tako vrijedi i za aktora 'Kupac'. Na taj način i sâm prikaz dijagrama postaje čitljiviji i sadržajniji. Slika 2.3.3 prikazuje tu razliku, gdje je pod stavkom (a) prikazano početno, a pod stavkom (b) poboljšano rješenje.



Slika 2.3.3 Moguća rješenja povezanosti aktora-korisnika i slučajeva uporabe

Na istovjetan način vezama pridruživanja moguće je povezati i aktore 'Trgovac' i 'Administrator' s njihovim slučajevima uporabe. Pod pretpostavkom jasnijeg odvajanja odgovornosti trgovca i administratora u ovom slučaju će se prikaz zajedničkih slučajeva uporabe učiniti na drugi način odnosno s drugačijim naglascima.

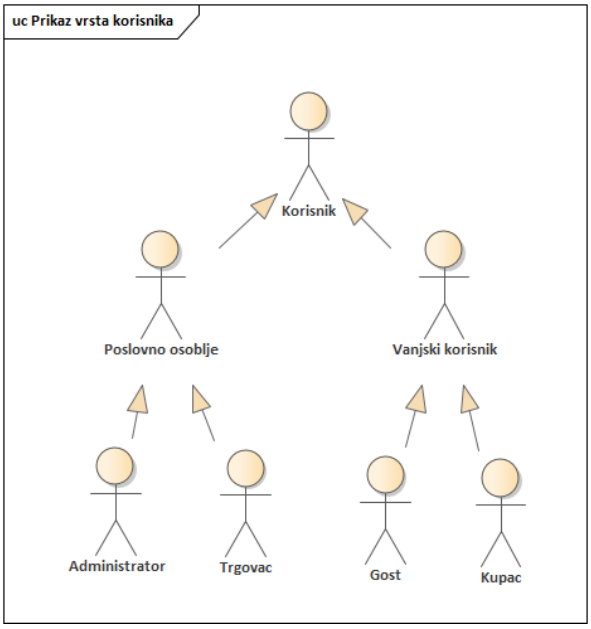
Slučajevi uporabe 'Uvid u korisničke narudžbe' i 'Uvid u statistiku prodaje' su načelno zajednički aktorima 'Trgovac' i 'Administrator', ali će se uzeti da temeljnu odgovornost za uvid u korisničke narudžbe ima trgovac kojem je to primarni posao. Isto tako, za pretpostaviti je da statistiku prodaje i najčešće odabranih proizvoda može generirati prije svih administrator kao stručna osoba u pitanjima dohvata informacija iz baze podataka proizvoda i prodaje. Zato će se na dijagramu slučajeva uporabe te veze između aktora i slučajeva uporabe jasno razdvojiti uz postavljanje dodatne bilješke ili komentara u dijagram. Slika 2.3.4 tako prikazuje sve aktore i slučajeve uporabe povezane odnosima pridruživanja. Spremište podataka kao aktor koji nije ljudski entitet prikazan je pravokutnom notacijom kao **pasivan aktor**, tj. onaj koji ne započinje komunikaciju već odgovara na akcije započete od strane sustava.



Slika 2.3.4 Veze pridruživanja s primjerom bilješke dijagrama slučajeva uporabe

S druge strane, hijerarhija korisnika po vrstama može se posebno prikazati. Može se reći da se sve postojeće vrste korisnika u ovom slučaju i u završnom programskom ostvarenju mogu izvesti iz iste općenite vrste korisnika 'Korisnik' pa je potrebno dodati veze poopćenja i između aktora 'Trgovac' i 'Administrator' prema aktoru 'Korisnik'. Međutim, to bi za posljedicu imalo postojanje četiri veze poopćenja od četiri vrste aktora koji se nalaze na različitim stranama dijagrama prema istom zajedničkom općem aktoru 'Korisnik' što bi sa strane preglednosti dijagrama bilo nepovoljno.

Zato je možda bolje napraviti zaseban dijagram slučajeva uporabe 'Prikaz vrsta korisnika' koji bi bio namijenjen prikazu takve postojeće hijerarhije aktora gdje se jasno pokazuju odnosi između postojećih vrsta korisnika sustava, Slika 2.3.5.



Slika 2.3.5 Odnosi između aktora – korisnika sustava

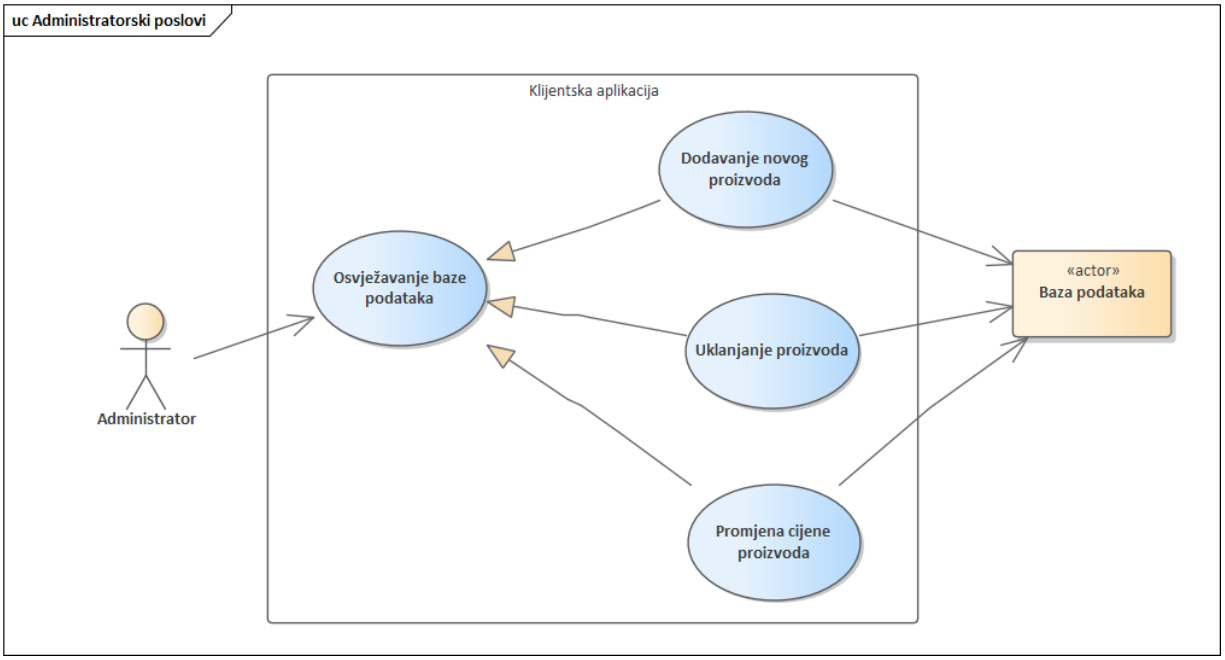
Kao što je već napomenuto, odnos poopćenja može postojati i međusobno između dva različita slučaja uporabe. U tom slučaju to govori da postoji slučaj uporabe koji je općenitiji u odnosu na drugoga, a da je taj drugi posebnije naravi u odnosu na prvoga.

Primjer za mogućnost poopćenja slučajeva uporabe se može naći u opisu projektnog zadatka gdje stoji:

Administratori bi prilikom rada aplikacije trebali imati mogućnost ažuriranja odnosno osvježavanja stanja dostupnih proizvoda u bazi podataka što podrazumijeva dodavanje, brisanje kao i promjene cijene pojedinih proizvoda te oblikovanje izvješća s povratnim informacijama dobivenim od strane korisnika.

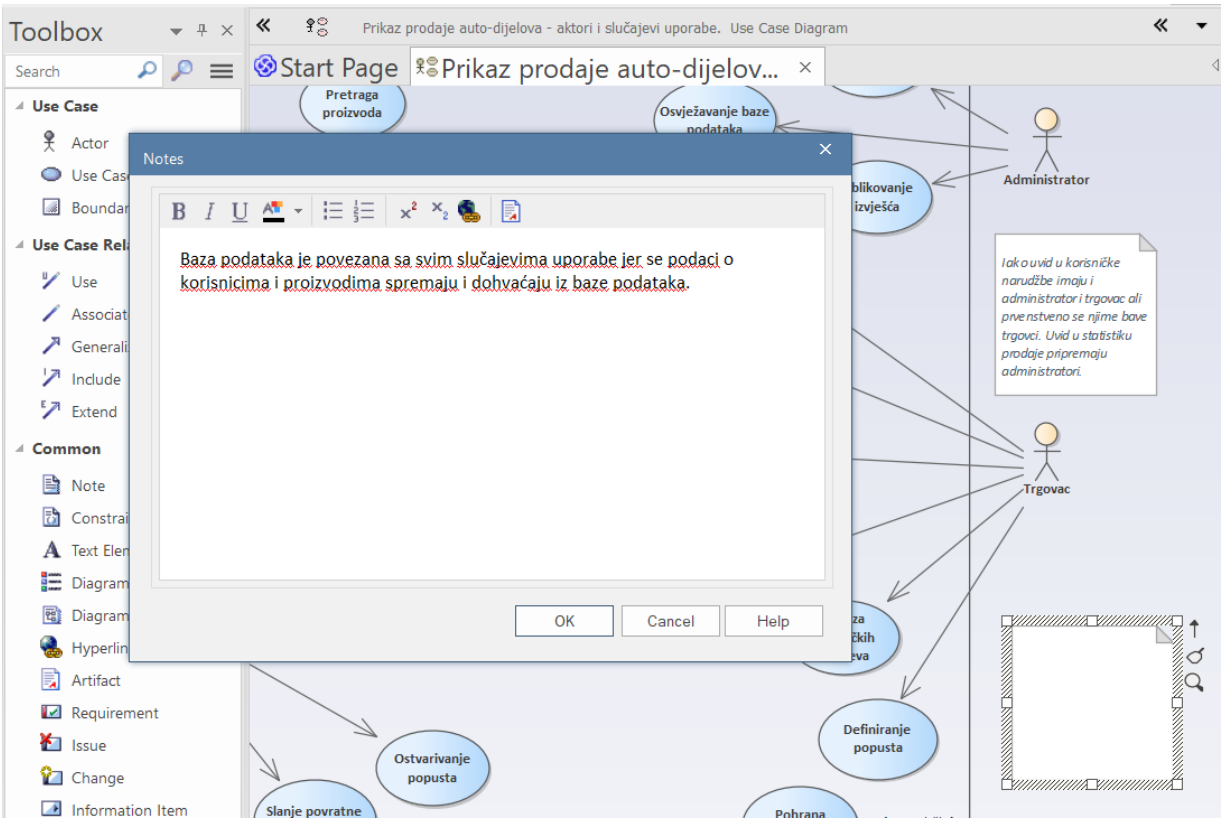
Prema tome, potrebna je funkcionalnost ažuriranja proizvoda u bazi podataka koja se u samoj sebi može sastojati od tri posebne vrste akcija: (1) dodavanja proizvoda u bazi, (2) brisanja proizvoda iz baze i (3) promjene cijena proizvoda u bazi podataka, Slika 2.3.6. Baza podataka

izričito je prikazana kao aktor u grafičkoj notaciji pravokutnika, a ne „čovječuljka“. Glavni aktor je 'Administrator' koji putem klijentske aplikacije upravlja osvježavanjem baze podataka auto-dijelova s tri različite vrste akcija koje su specijalizacija slučaja uporabe 'Osvježavanje baze podataka', tj. postoje veze poopćenja od posebnijih slučajeva uporabe 'Dodavanje novog proizvoda', 'Uklanjanje proizvoda' i 'Promjena cijene proizvoda' prema općem slučaju uporabe 'Osvježavanje baze podataka'.



Slika 2.3.6 Detaljniji prikaz slučaja uporabe ‘Osvježavanje baze podataka’

Važno je uočiti da na glavnom dijagramu slučajeva uporaba baze podataka nije prikazana kao zaseban aktor niti je slučaj uporabe koji pokriva funkcionalnost ažuriranja proizvoda u bazi podataka prikazan u tom dijagramu. Razlog za to je da je baza podataka integralni dio sustava aplikacije kojoj korisnici pristupaju te je usto povezana praktično sa svim slučajevima uporabe jer se skoro svaka transakcija u sustavu temelji na pristupu bazi podataka, bilo da je riječ o slanju upita u bazu ili upis nove prodaje ili proizvoda u bazu. Zato bi zbog preglednosti dijagrama bilo uputno staviti posebnu bilješku o tome, Slika 2.3.7. Što se tiče dijagrama administracije baze podataka prikaz je ograničen na administratorsku klijentsku aplikaciju pa je baza podataka izričito prikazana kao dio pozadinskog dijela aplikacije.



Slika 2.3.7 Bilješka o postojanju 'Baze podataka' kao podrazumijevanog aktora

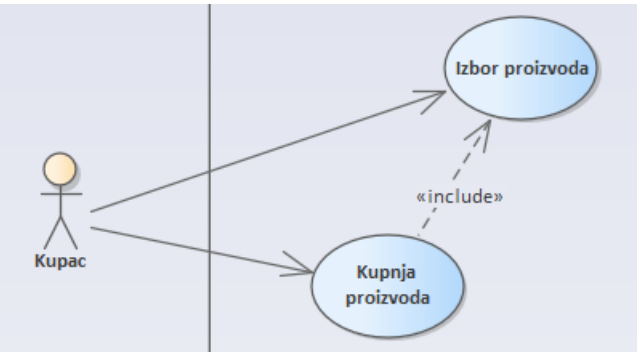
Baza podataka je u oba slučaja aktor pasivne vrste, jer u pravilu samo odgovara na akcije korisnika inicijatora što je naznačeno usmjerenjem veza pridruživanja od slučajeva uporabe prema aktoru 'Baza podataka'.

Odnos obuhvaćanja

Veza obuhvaćanja (engl. *include*) definira se između dva slučaja uporabe i označava da provedba osnovnog slučaja uporabe obuhvaća ili uključuje u sebi provedbu uključenog slučaja uporabe, tj. da se uključeni slučaj uporabe izvršava u sklopu izvršavanja osnovnog slučaja uporabe. Koristi se za pojednostavljenje složenih slučajeva uporabe podjelom u više manjih i izdvajanje manjih slučajeva uporabe koji su zajednički onim složenijim, tj. radi se o funkcijskoj dekompoziciji slučajeva uporabe. Osnovni slučaj uporabe je nepotpun bez uključenog slučaja uporabe. Odnos se označava notacijom isprekidane crte s usmjerenjem od osnovnog prema uključenom slučaju uporabe.

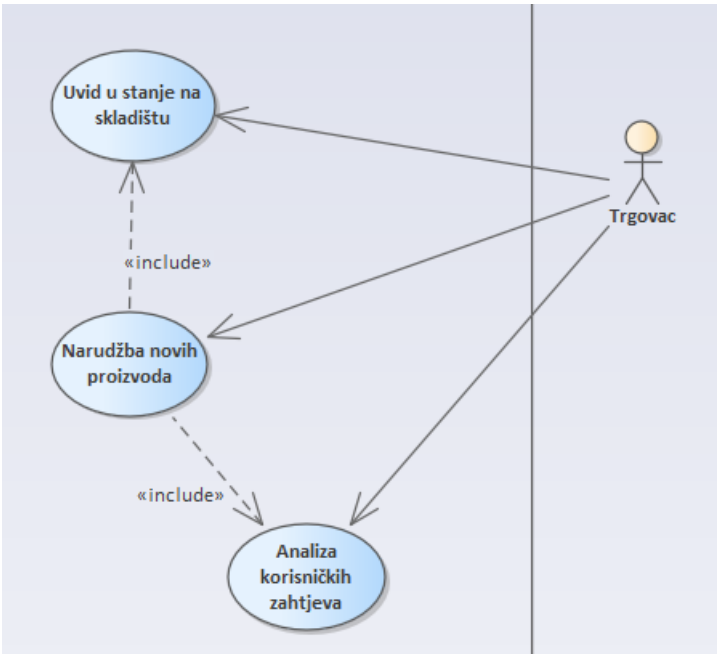
Primjerice, u dijagramu slučajeva uporabe 'Prikaz prodaje' slučaj uporabe 'Kupnja proizvoda' podrazumijeva slučaj uporabe 'Izbor proizvoda', tj. ne može se provesti kupnja nekog proizvoda bez da se napravi izbor istog proizvoda. Ako je slučaj uporabe 'Kupnja proizvoda' nepotpun u provedbi bez slučaja uporabe 'Izbor proizvoda' stavlja se odnos obuhvaćanja, Slika 2.3.8. Pri tome bi se veza pridruživanja između aktora Kupac i slučaja uporabe 'Izbor proizvoda' mogla i

izostaviti unutar dijagrama jer se podrazumijeva posredno preko slučaja uporabe 'Kupnja proizvoda'.



Slika 2.3.8 Veza obuhvaćanja na dijagramu slučajeva uporabe

Isto tako, slučaj uporabe 'Narudžba novih proizvoda' u svojem provođenju podrazumijeva provođenje slučajeva uporabe 'Uvid u stanje na skladištu' i 'Analiza korisničkih zahtjeva'. Uključeni slučajevi se mogu provesti nezavisno od osnovnog pa postoje i veze pridruživanja od aktora 'Trgovac' prema njima, ali bi se te veze mogle i izostaviti, Slika 2.3.9.



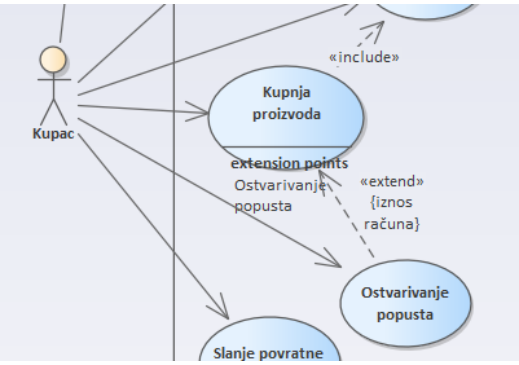
Slika 2.3.9 Dijagram slučajeva uporabe 'Prikaz prodaje' s umetnutim vezama obuhvaćanja

Može se reći da je slučaj uporabe 'Prijava u sustav' je preduvjet za veliku većinu ostalih slučajeva uporabe ali to nije isto što i odnos obuhvaćanja. Preduvjeti se pišu zasebno u raspisu slučajeva uporabe, a mogu se navesti i u komentarima na dijagramu, ali se nikako ne modeliraju vezom obuhvaćanja. Osim toga, korisnik ne mora svaki put napraviti prijavu prilikom uporabe sustava jer može ostati prijavljen otprije što ima značenje preduvjeta a ne odnosa obuhvaćanja koji bi svaki put podrazumijevao izvođenje uključenog slučaja uporabe.

Odnos proširenja

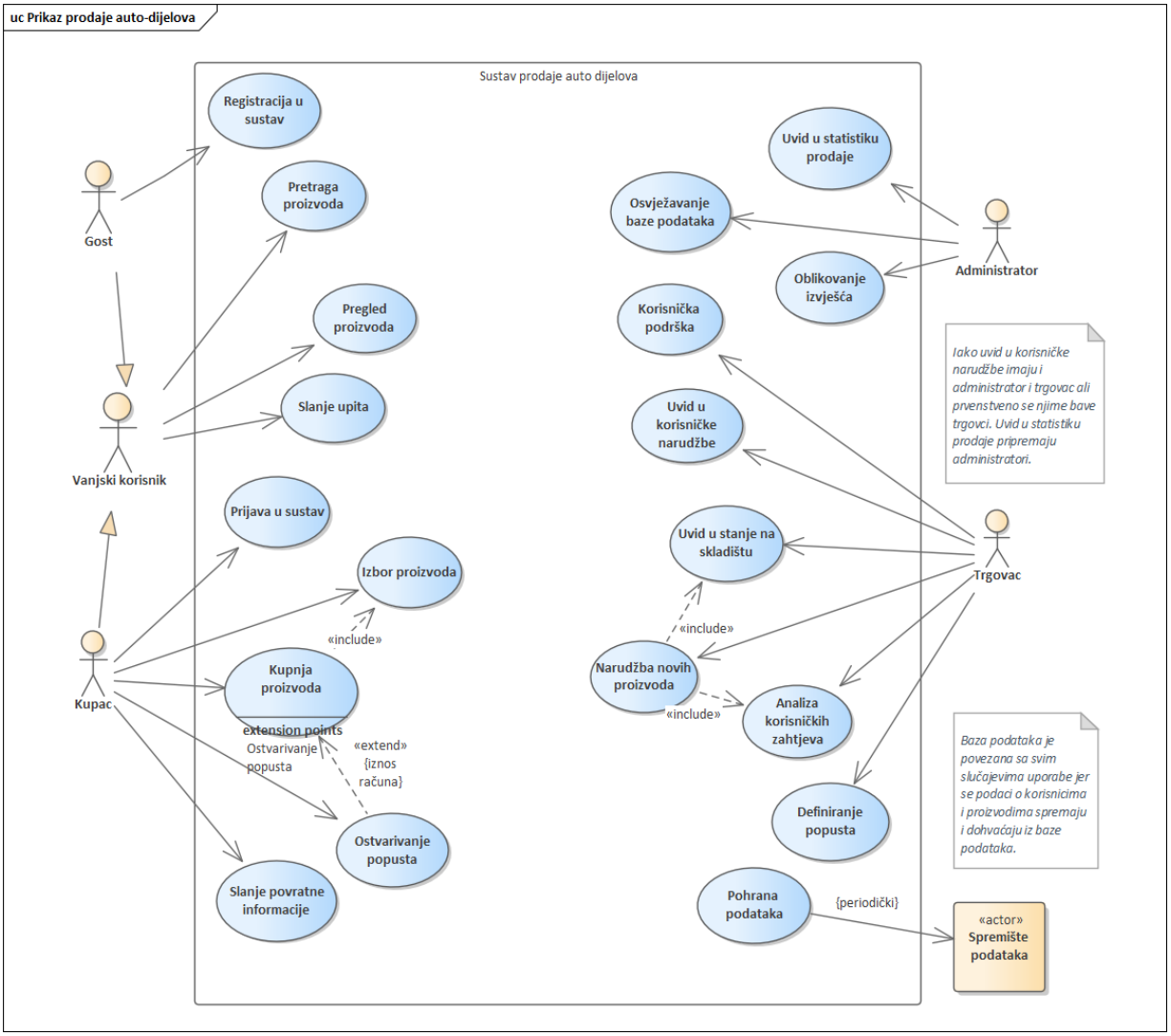
Veza proširenja (engl. *extend*) modelira odnos slučajeva uporabe kada provedba osnovnog slučaja uporabe pod određenim uvjetima može biti proširena s provedbom nekog drugog slučaja uporabe. Veza je usmjerena od proširenog prema osnovnom slučaju uporabe.

Unutar osnovnog slučaja uporabe može se definirati tzv. točka proširenja (engl. *extension point*) na koju se spaja prošireni slučaj uporabe i uz koju se navodi uvjet proširenja. Npr. u dijagramu slučajeva uporabe 'Prikaz prodaje' slučaj uporabe kupca 'Kupnja proizvoda' se proširuje s slučajem uporabe 'Ostvarivanje popusta' pod uvjetom da se prijeđe neki iznos računa. Pri tome se navodi uvjet proširenja te se prikaz osnovnog slučaja uporabe proširuje točkom proširenja i ispisom uvjeta proširenja (Slika 2.3.10).



Slika 2.3.10 Izgled točke proširenja na dijagramu slučajeva uporabe

Slika 2.3.11 prikazuje konačni izgled dijagrama slučajeva uporabe 'Prikaz prodaje' nakon umetanja svih aktora, slučajeva uporabe i njihovih odnosa.

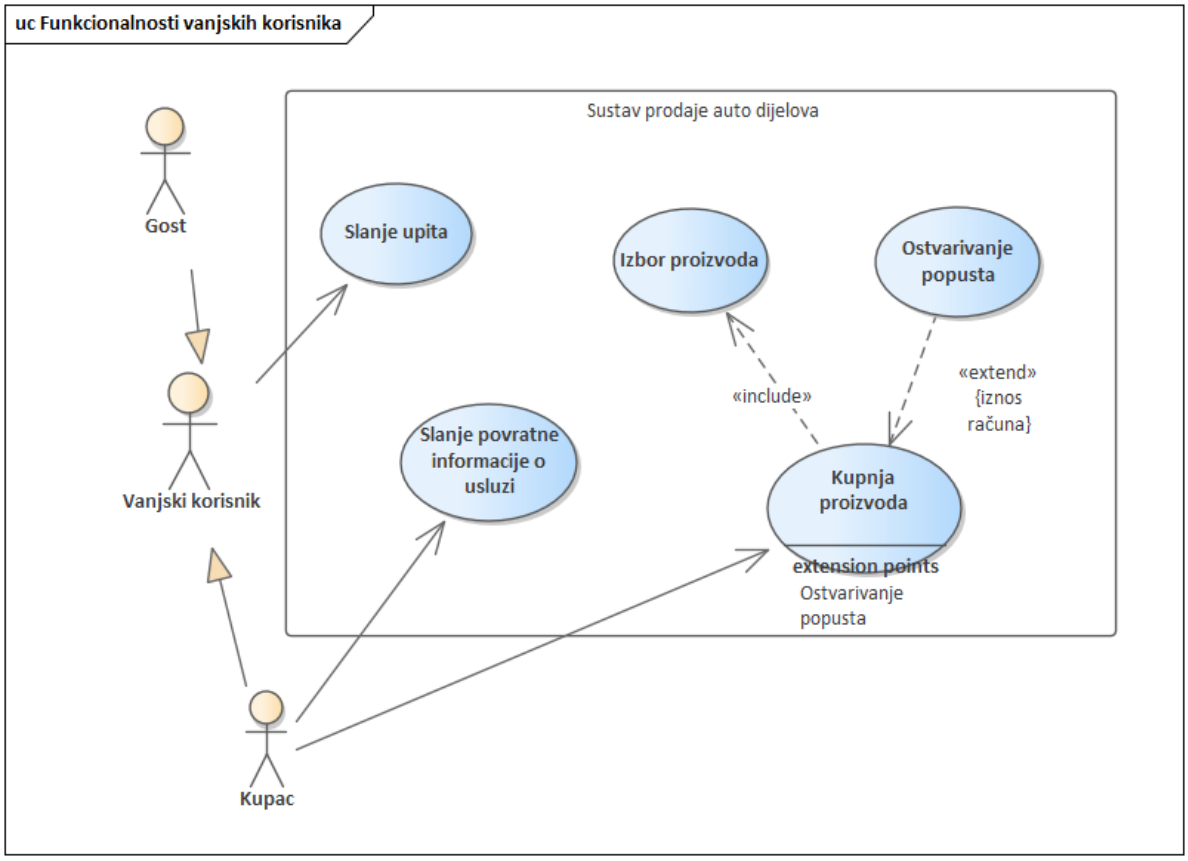


Slika 2.3.11 Konačni izgled dijagrama slučajeva uporabe 'Prikaz prodaje auto-dijelova'

Svi aktori su na odgovarajući način povezani sa svojim slučajevima uporabe uz popratne komentare.

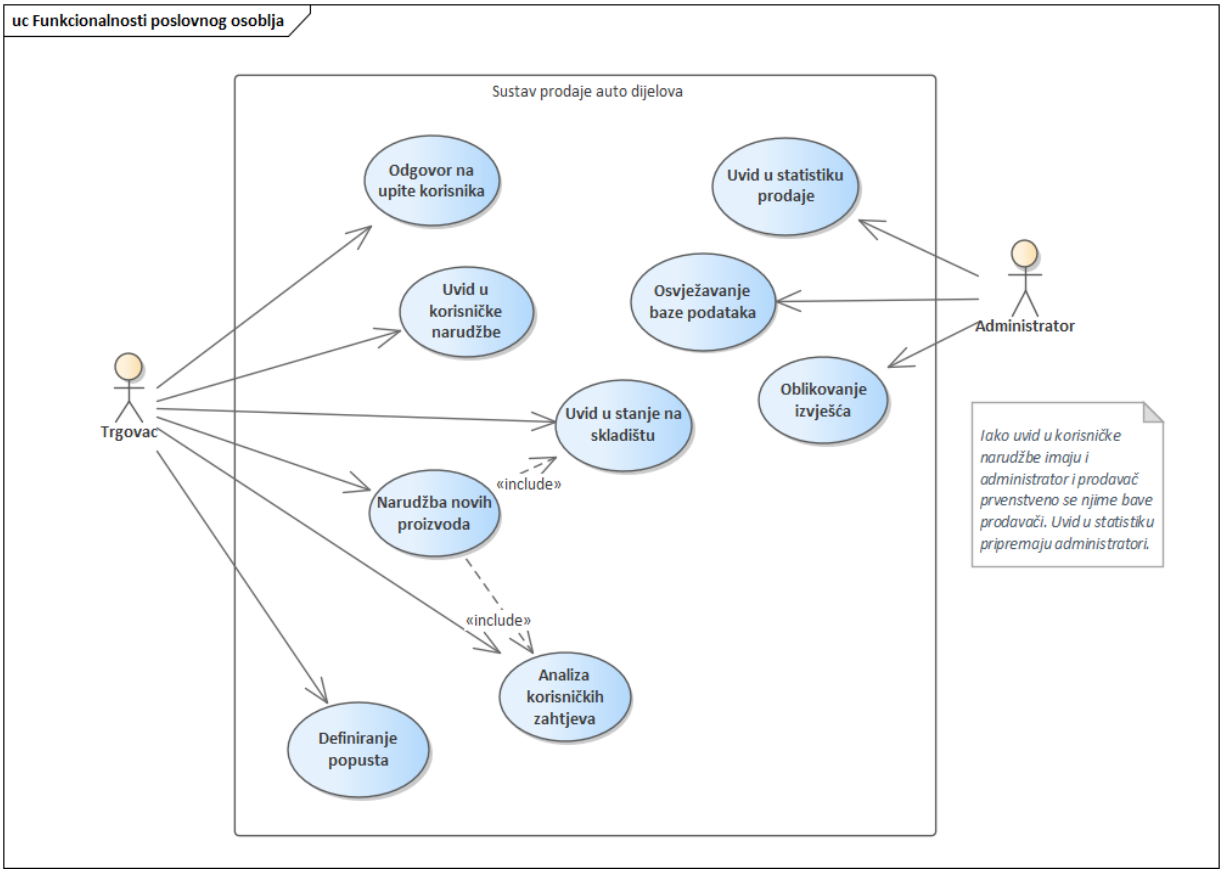
Pojednostavljenja prikaza

Prikaz jednog složenijeg dijagrama slučaja uporabe cijelog sustava se može pojednostavniti razdvajanjem istoga u više jednostavnijih dijagrama i pojednostavljenjima u prikazu.
Primjer pojednostavljenja već je prikazan u razvijenom dijagramu cijelog sustava u smislu izostavljanju prikaza baze podataka kao posebnog aktora i dodavanju odgovarajućeg komentara.
Ako se dijagram čini presložen može ga se podijeliti u više jednostavnijih dijagrama s podskupovima funkcionalnosti definiranih prema nekoj logici. Npr. zasebno se može prikazati dio s pregledom izabranih slučajeva uporabe vanjskih korisnika, Slika 2.3.12.



Slika 2.3.12 Dijagram slučajeva uporabe prodaje auto-dijelova sa strane vanjskih korisnika

Isto tako, zasebno se može prikazati i dio sa slučajevima uporabe poslovnog osoblja, Slika 2.3.13.



Slika 2.3.13 Dijagrama slučajeva uporabe prodaje auto-dijelova sa strane poslovnog osoblja

Ovisno o dojmu i potrebama razumljivosti dijagrama razdvajanje složenih dijagrama može biti više ili manje opravdano. Neke elemente nekad nije potrebno navoditi ni u jednostavnijim dijagramima kao što ni na ovom dijagramu nije prikazano poopćenje dvije vrste korisnika kao osoblja web prodaje.

Razlikovanje vrste odnosa

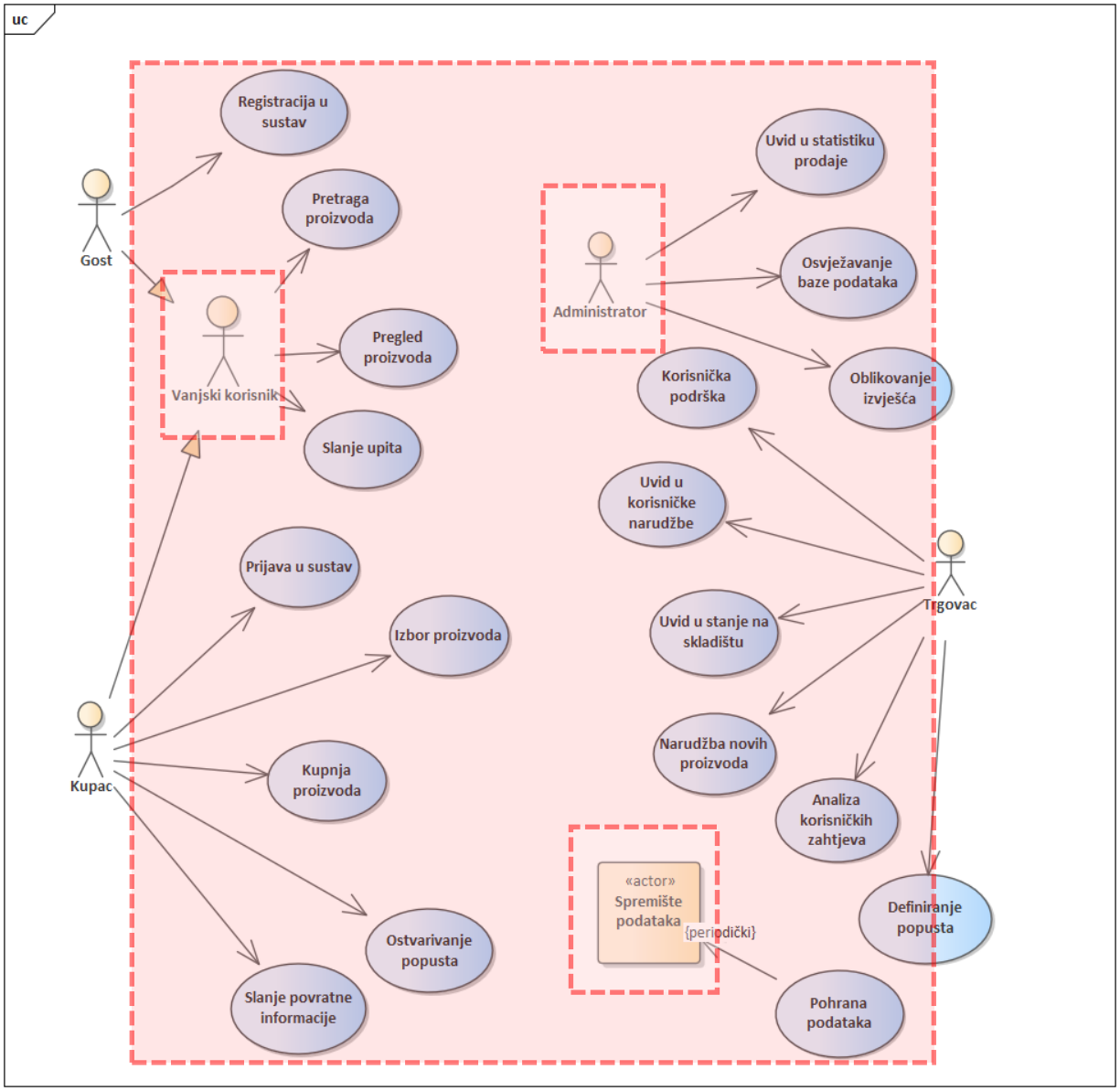
Vrste odnosa koji se mogu definirati između slučajeva uporabe međusobno, razlikuju se po notaciji i semantici. Potrebno je razlikovati slučajeve uporabe uključene u pojedini odnos po njihovoj potpunosti ili nepotpunosti odnosno nužnosti ili opcionalnosti.
Npr. kod odnosa poopćenja osnovni slučaj uporabe može biti i apstraktan, tj. nepotpun, i konkretan, odnosno potpun. Za razliku od njega, kod odnosa obuhvaćanja osnovni slučaj uporabe je uvijek nepotpun jer se ne može do kraja izvršiti bez sebi obuhvaćenog slučaja uporabe, dok je kod odnosa proširenja osnovni slučaj uporabe uvijek potpun jer se proširujući slučaj uporabe ne mora izvršiti svaki put kod izvođenja osnovnog slučaja uporabe.
Kod odnosa obuhvaćanja uključeni slučaj uporabe nužan je za izvođenje osnovnog slučaja uporabe, a kod odnosa proširenja dodatni proširujući slučaj uporabe nije nužan već opcionalan.
Kod odnosa poopćenja specijalni slučaj uporabe je nužan samo ako je osnovni slučaj uporabe apstraktan odnosno nepotpun, a inače nije nužan.

Česte pogreške

U praksi modeliranja primijećene su neke ponavljane pogreške u izradi dijagrama slučajeva uporabe koje se ukratko komentiraju u odlomcima koji slijede gdje je za sve prikazane dijagrame koji predstavljaju kriva ili nepoželjna rješenja to posebno i naznačeno na njima.

Nepostojanje granice sustava

Jedan od uobičajenih zahtjeva je da se naznači granica sustava. U slučaju nepostojanja odnosno nepovlačenja granice sustava koja odvaja aktore od slučajeva uporabe moguće je da se slučajevi uporabe i aktori prikažu ispremiješano što narušava čitljivost dijagrama, Slika 2.3.14.

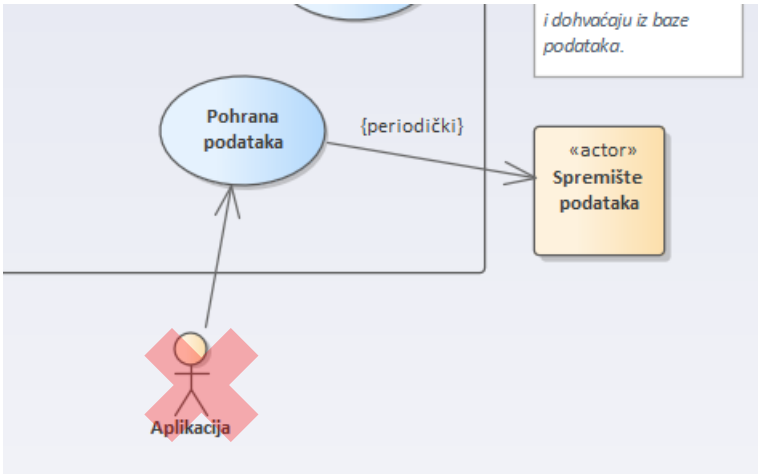


Slika 2.3.14 Česte pogreške: primjer nepostojanja granice sustava

Za razliku od dijagrama koji je predstavljen kao valjani dijagram slučajeva uporabe (Slika 2.3.11) ovdje se može zamijetiti da su aktori i slučajevi uporabe poziciono „izmiješani“ unutar prostora dijagrama i da pravilnu i urednu granicu sustava nije moguće lako ni povući. Vezano uz to se može dogoditi da postoji granica sustava ali da se neki od aktora nalazi unutar nje što narušava semantiku prikaza i smisao dijagrama.

Sustav kao poseban aktor

Nekad se u rješenjima studenata sustav prikazuje kao poseban aktor, kada se za modelirani sustav ili aplikaciju definira i umeće zaseban aktor izvan granica sustava koji je povezan s pojedinim slučajevima uporabe. Sustav ili aplikacija je predstavljena sadržajem prikazanim unutar granica sustava pa nema potrebe za umetanjem takvog aktora. Primjer toga bi u ovdje elaboriranom primjeru (Slika 2.3.11) mogao biti kada bi netko stavio posebnog aktora naziva 'Aplikacija' koji pokreće pohranu podataka, što bi bilo naznačeno vezom pridruživanja sa slučajem uporabe 'Pohrana podataka', Slika 2.3.15 .

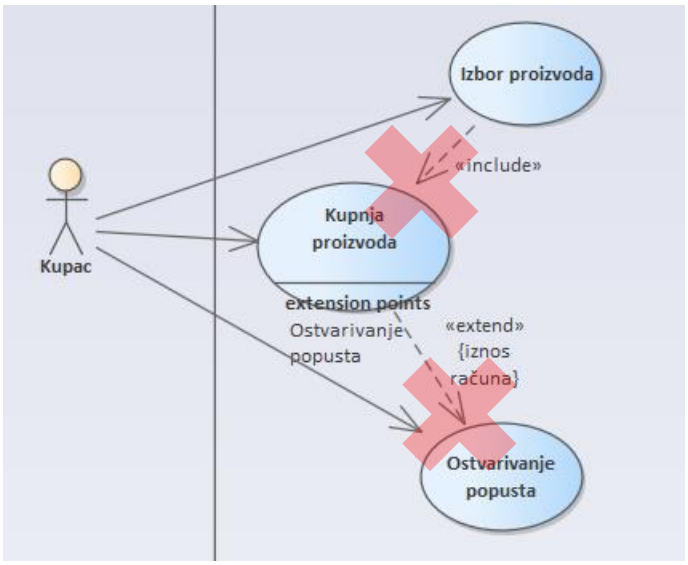


Slika 2.3.15 Česte pogreške: umetanje sustava kao posebnog aktora

Cijeli dijagram zapravo predstavlja aplikaciju koja se njima opisuje, gdje sadržaj unutar granice sustava predstavlja njezine funkcionalnosti, a aktori entitete koji se nalaze izvan aplikacije pa je umetanje takvog aktora loša praksa.

Krivi smjerovi veza

Smjerovi veza obuhvaćanja i uključivanja često mogu biti krivo naznačeni ili zamijenjeni, Slika 2.3.16.

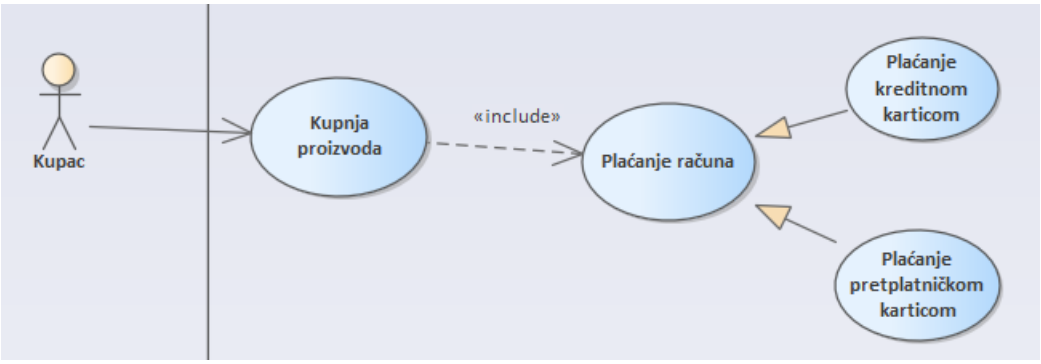


Slika 2.3.16 Česte pogreške: primjer krivih usmjerenja pojedinih veza

U primjeru sa slike usmjerenja veza obuhvaćanja i proširenja su zamijenjena. Kupnja proizvoda u sebi podrazumijeva ili obuhvaća izbor proizvoda koji se kupuju, a ostvarivanje popusta je opcionalna mogućnost. Naznačeni smjerovi su pri tome krivo navedeni.

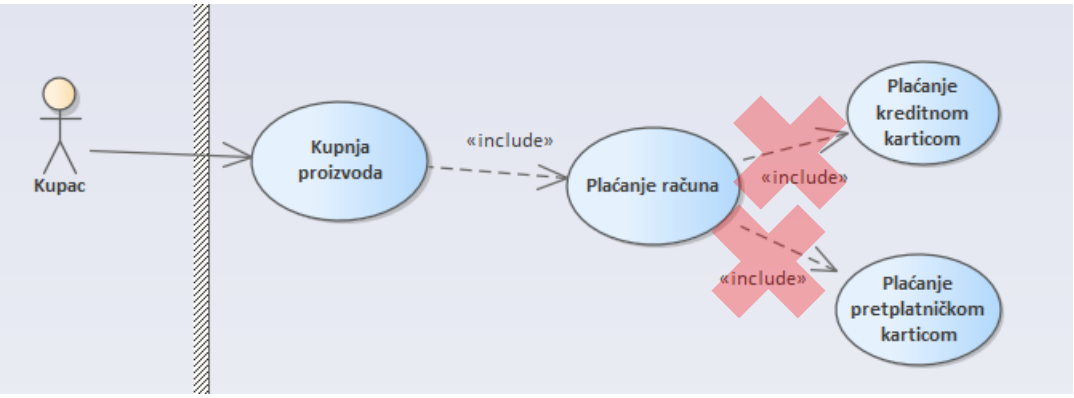
Zamjena semantike

U nekim slučajevima, zbog neprepoznavanja ili nerazumijevanja semantike poopćenja ili specijalizacije u odnosu na semantiku obuhvaćanja i proširenja slučajeva uporabe, dolazi do njihove zamjene u prikazu dijagrama.
Primjerice, kada bi u elaboriranom primjeru postojala dva načina plaćanja računa: kreditnom karticom i pretplatničkom karticom to bi trebalo prikazati specijalizacijama slučaja uporabe 'Plaćanje računa', Slika 2.3.17.



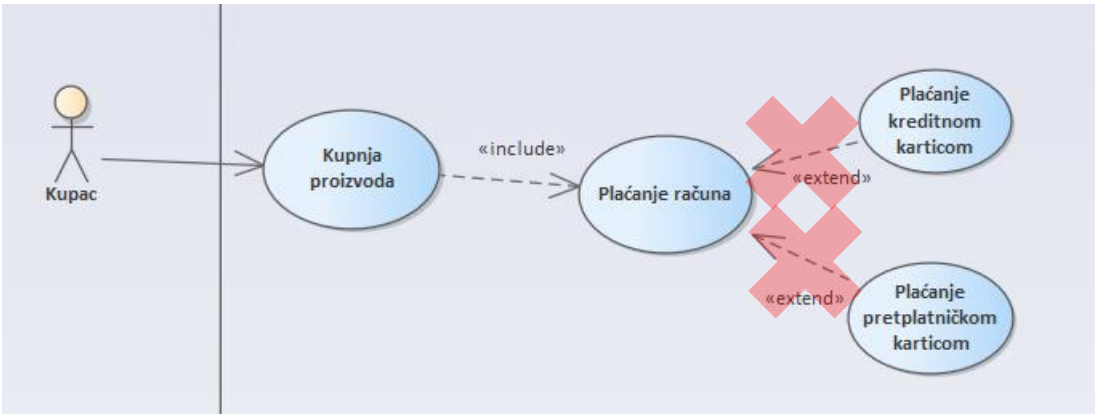
Slika 2.3.17 Primjer specijalizacija slučaja uporabe 'Plaćanje računa'

Ako se semantika poopćenja krivo shvati, često dolazi do umetanja semantike obuhvaćanja, koja bi u navedenom primjeru govorila da se kod svakog plaćanja računa obavlja plaćanje i kreditnom karticom i pretplatničkom karticom što je u praktičnim slučajevima nepotrebno, Slika 2.3.18.



Slika 2.3.18 Česte pogreške: zamjena semantike obuhvaćanja i poopćenja

Ovdje se neki općeniti slučaj uporabe koji ima svoje opcije ili specijalne slučajeve uporabe modelira kao osnovni slučaj uporabe koji obuhvaća sve svoje opcionalne slučajeve uporabe.
Isto tako, događaju se pogreške razumijevanja i u obratnom pogledu kada se osnovni slučaj uporabe koji obuhvaća niz drugih slučajeva uporabe prikaže kao poopćenje tih drugih slučajeva uporabe.
Moguće je i drugo krivo tumačenje i pogrešan prikaz postojanja različitih opcija plaćanja u smislu da se ta opcionalnost shvati u smislu proširenja osnovnog slučaja uporabe s neobaveznim opcijama.

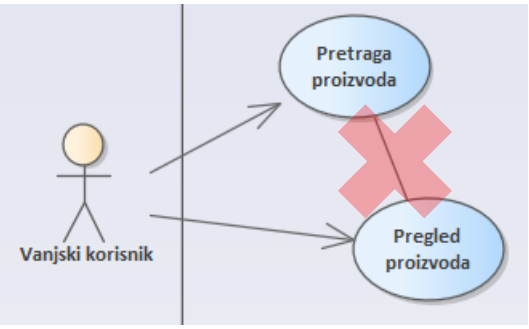


Slika 2.3.19 Česte pogreške: zamjena semantike proširenja i poopćenja

Ovdje se pogreška napravila u smislu načina razumijevanja tvrdnje da kupac može platiti kreditnom karticom a može platiti i pretplatničkom karticom. gdje se podrazumijeva da svakako nekako plati račun, na jedan ili drugi način. Ako se to shvati kao odnos proširenja dolazi se do određenog proturječja: osnovni slučaj uporabe u tom prikazu bi bio predstavljen kao potpun i poopćeno naveden kojemu bi navedeni prošireni slučajevi uporabe kao detalji načina plaćanja bili samo višak.

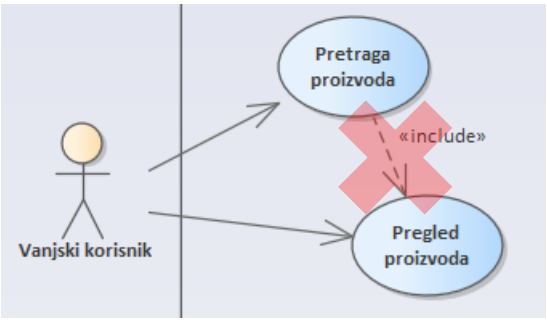
Međusobno povezivanje ili ulančavanje slučajeva uporabe

Odnos pridruživanja može postojati samo između aktora i slučaja uporabe. U nemogućnosti kvalitetne razrade svih slučajeva uporabe i njihovih odnosa nekad se pribjegava umetanju bilo kakve veze, npr. veze pridruživanja između pojedinih slučajeva uporabe što nije dozvoljeno, Slika 2.3.20.



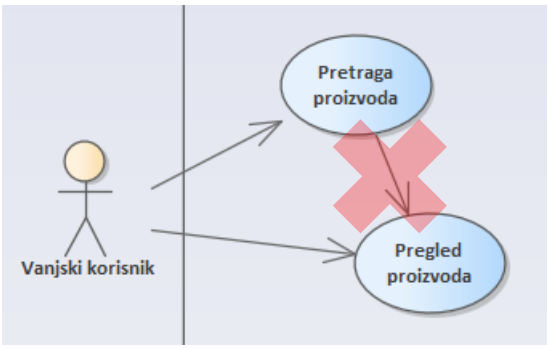
Slika 2.3.20 Česte pogreške: umetanje odnosa pridruživanja između slučajeva uporabe

U redovitim slučajevima između slučajeva uporabe nije potrebno umetati veze, ako je jasno da ne postoji semantika poopćenja, obuhvaćanja ili proširenja između njih.
Nekad se učini nužnom potreba naglaska vremenskog uređenja između pojedinih slučajeva uporabe, ali to nije dio namjene dijagrama slučajeva uporabe već dijagrama slijeda, dijagrama aktivnosti i dijagrama komunikacije koji će biti objašnjeni poslije u tekstu. Primjerice, ako se u elaboriranom primjeru želi reći da nakon pretraživanja proizvoda auto-dijelova slijeda u pravilu slijedi pregled nekog od nađenih proizvoda može se dogoditi pribjegavanje umetanju odnosa obuhvaćanja između dotičnih slučajeva uporabe, Slika 2.3.21.



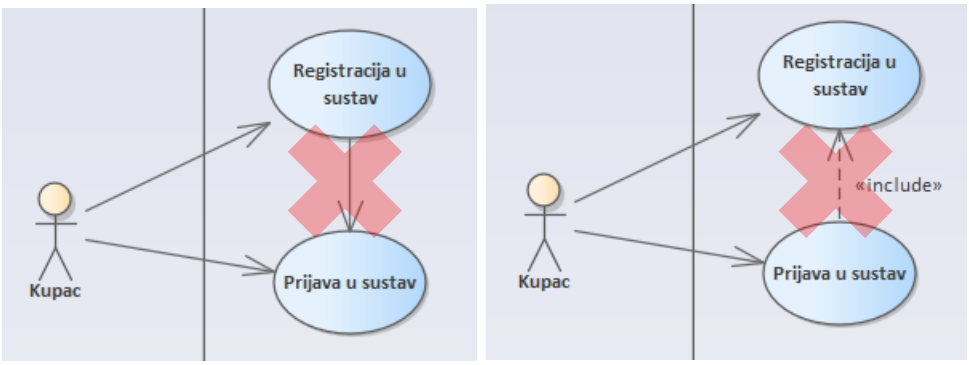
Slika 2.3.21 Česte pogreške: umetanje odnosa obuhvaćanja

To u ovom slučaju ima logičkog temelja, ali nije potpuno ispravno ni nužno jer neki korisnik može raditi pretragu bez pregleda detalja proizvoda.
Pokušaj naglaska vremenskog uređenja se na sličan način može pokušati naznačiti umetanjem usmjerenja prethodno spomenute veze pridruživanja što opet sliči na modeliranje procesa ili tokova kao u dijagramu aktivnosti, Slika 2.3.22.



Slika 2.3.22 Česte pogreške: „ulančavanje“ slučajeva uporabe

Ovim dijagramom se želi reći da je pretraga proizvoda preduvjet pregledu proizvoda što opet nije notacijski ispravno niti nužno za pokazati jer u ovom slučaju ni ne mora uvijek biti istina, npr. ako u korisničkom sučelju postoji izbor istaknutih proizvoda čijim se detaljima može izravno pristupiti bez posebno pretraživanja.
Primjer sličnih pokušaja povezivanja slučajeva uporabe se može vidjeti i na dijelu funkcionalnosti vezanom uz registraciju i prijavu korisnika odnosno kupca u ovom slučaju, Slika 2.3.23.

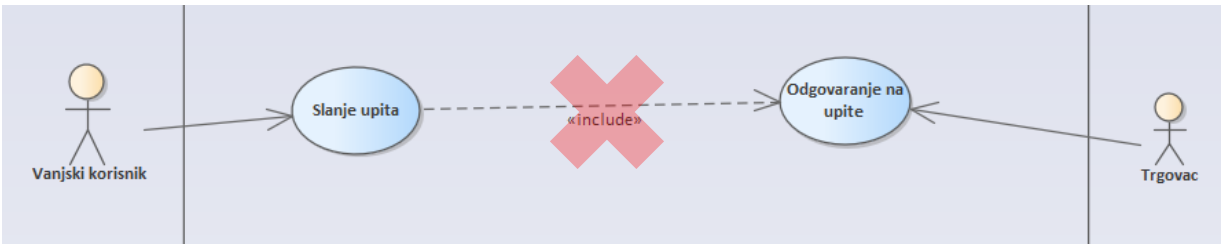


Slika 2.3.23 Česte pogreške: povezivanje slučajeva uporaba

Ako se želi reći da je prethodna registracija preduvjet budućim prijavama u sustav onda se to ne može naznačiti nedopuštenom vezom pridruživanja ni vezom obuhvaćanja jer je registraciji potrebno učiniti samo jednom a nakon toga se nebrojeni broj puta korisnik može prijaviti u sustav.

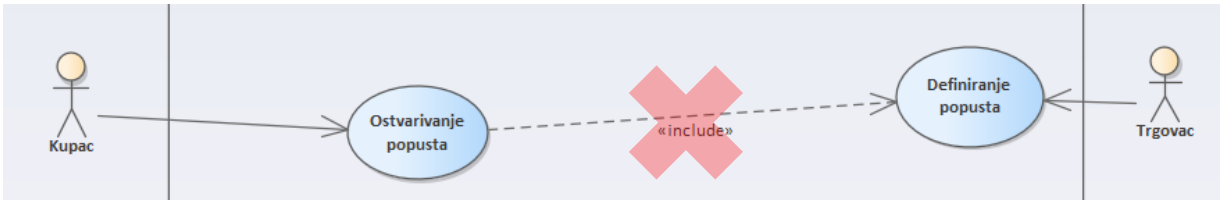
Obuhvaćanje slučajeva uporabe različitih inicijatora

Povezivanja nekih skupina slučajeva uporabe može se učiniti potpuno logičnom akcijom ako se ne uoči činjenica da su započeti od strane različitih aktora inicijatora, Slika 2.3.24.



Slika 2.3.24 Česte pogreške: povezivanje slučajeva uporabe različitih inicijatora

U pojedinim slučajevima to nije ni logički opravdano, jer npr. u prikazanom slučaju slanje upita ne podrazumijeva dobivanje odgovora koliko god to bilo poželjno, ali se zapravo radi narušavanju semantike odnosa obuhvaćanja: slučaj uporabe slanja upita nije ovisan te se završava neovisno o tome hoće li se odgovor na upit dobiti, odnosno poslati natrag onome tko ga je uputio.
Sličan primjer toga u elaboriranom primjeru bi bio onaj vezan uz ostvarivanje popusta pri kupnji Slika 2.3.25.



Slika 2.3.25 Česte pogreške: povezivanje slučajeva uporabe različitih inicijatora

U ovom slučaju, ako se želi reći da se ostvarivanje popusta temelji na prethodnoj funkcionalnosti definiranja određenih uvjeta ostvarivanja popusta to se logički može podrazumijevati i bez umetanja veze obuhvaćanja ili dodatno navesti umetanje prikladnog komentara.
Prikazani primjeri nisu do kraja semantički objašnjivi pa bi se umetanje takve veze u prvom primjeru moglo nepravilno čitati da 'Vanjski korisnik' šalje i ujedno odgovara na upite, a u drugom da 'Kupac' ostvaruje popust na način da ga ujedno ili usput definira.

2.4. Dijagram aktivnosti

Dijagram slučajeva uporabe u svojoj biti već prikazuje temeljne ili neke od temeljnih aktivnosti koje su oblikovane kao slučajevi uporabe s nazivima akcija. Prikazuju se slučajevi uporabe na način da se govori što se kroz sustav može napraviti od strane kojeg korisnika, ali ne postoji naznačeno vremensko uređenje ili redoslijed između slučajeva uporabe i aktivnosti korisnika koje provode, npr. na način kako je to prikazano nekim dijagramom toka. Slučajevi uporabe se u sebi sastoje od mnogo koraka koji nisu vidljivi na dijagramu slučajeva uporabe pa se dijagram aktivnosti može koristiti upravo za svrhu prikaza i razrade tih koraka.

Dijagram aktivnosti (engl. *activity diagram*), kao i dijagram slučaj uporabe, jest dijagram ponašajne vrste koji prikazuje redoslijed ili tijek akcija sudionika te odnose između njih, kao i između dijelova sustava, prikazom toka akcija unutar pojedinih aktivnosti. Upotrebljava se i za modeliranje poslovnih procesa kao tijeka i nadovezivanja pojedinih akcija više korisnika. Oni se ne moraju potpuno preslikavati na slučajeve uporabe koji su prethodno definirani već mogu prikazati i veću razinu detalja pri akcijama sudionika kao i niz njihovih uzastopnih akcija.

Aktivnost predstavlja specifikaciju ponašanja sudionika sustava najčešće predstavljenih nizom akcija povezanih u obliku usmjerenog grafa. Općenito, mogu se prikazati odnosi u aktivnostima na razini pojedinaca ili objekata izvedenih iz prethodno definiranih entiteta. Aktivnosti se sastoji od niza akcija koje mogu biti vizualno podijeljene prema sudionicima te mogu sadržavati prikaz više različitih upravljačkih tokova koji se oblikuju usmjerenim prijelazima između akcija.

Akcije su prikazane čvorovima u obliku zaobljenih pravokutnika povezanih usmjerenim vezama (lukovima) kao prijelazima između akcija. Akcije predstavljaju atomarne ili nedjeljive operacije a čvorovi u dijagramu mogu sadržavati i druge aktivnosti na različitim razinama razlučivosti ili granulacije. Usmjereni prijelazi izražavaju redoslijed izvršavanja akcija te mogu prikazati, osim upravljačkog toka, i tok objekata.

Upravljački tok na dijagramu aktivnosti se oblikuje umetanjem posebnih **čvorova odluke** (engl. *decision*) i **spajanja** (engl. *merge*) prikazanih notacijom geometrijskog lika romba kao čvora odluke koji se podudara sa semantikom uvjetnog grananja: na čvoru se postavlja pitanje ili kriterij grananja, a izlaznim granama se unutar uglatih zagrada prikazuju pojedini uvjeti izlaska iz čvora odluke. Notacija uglatih zagrada se koristi na više vrsta UML dijagrama u istu svrhu.

Posebni su još **početni čvor** (engl. *initial*), **završni čvor** (engl. *final*) aktivnosti i **čvor završetka pojedinačnog toka** (engl. *flow final*). Početni i završni čvor se prikazuju notacijom ispunjenog kružića gdje je završni čvor dodatno zaokružen. Čvor završetka pojedinačnog toka se označava s križićem, odnosno dvije prekrižene kose crte, unutar praznog kružića i predstavlja završetak jedne od postojećih grana upravljačkog toka dok završni čvor na dijagramu predstavlja završetak svih grana upravljačkog toka unutar prikazane aktivnosti. Moguće je da dijagram aktivnosti ima više početnih i završnih čvorova. Osim toga, na dijagramu je moguće prikazati paralelne aktivnosti ili nizove akcija, kao i uporabu posebnih vrsta signala unutar upravljačkog toka što će biti detaljnije prikazano u primjerima koji slijede.

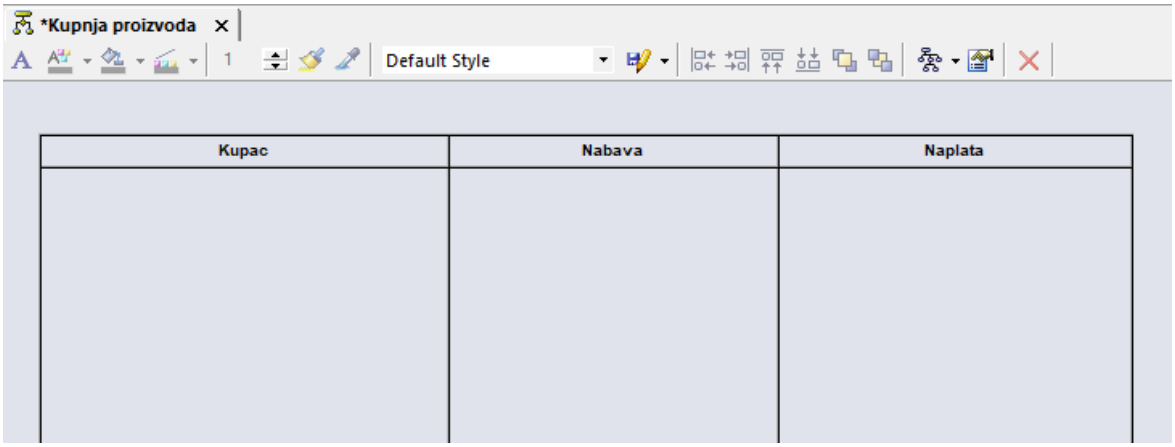
U primjerima oblikovanja prikazat će se izrada dijagrami aktivnosti kupnje proizvoda od strane kupca, narudžbe novih proizvoda od strane trgovca i održavanja baze podataka od strane administratora.

Kupnja proizvoda

Dijagramom aktivnosti želi se prikazati provedba kupnje proizvoda od strane kupca, od izbora do naplate proizvoda. Ta provedba uključuje nekoliko sudionika: kupca i podsustave nabave i naplate. Kupac naručuje neki proizvod, sustav nabave provjerava stanje primjeraka proizvoda na skladištu, a sustav naplate obavlja naplatu po izdanom računu.

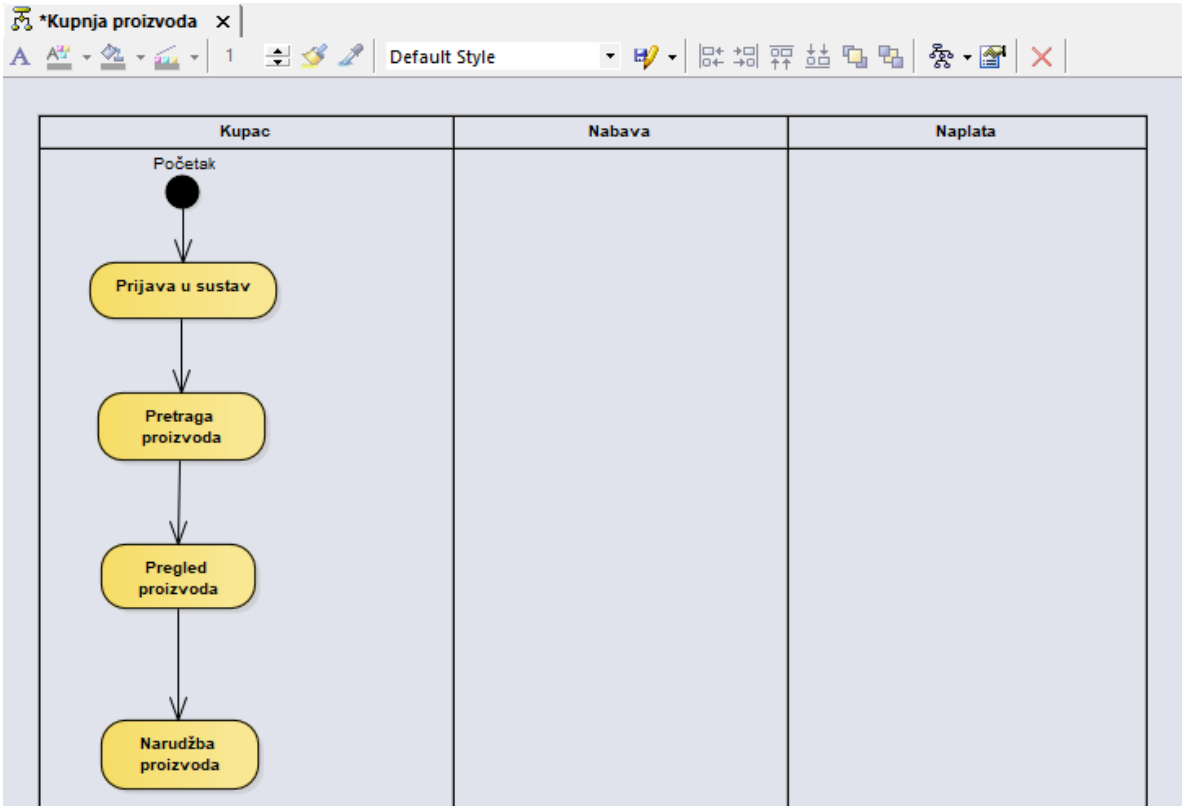
Prikaz dijagrama aktivnosti ima mogućnost podjele aktivnosti po **participama**, tj. odvojenim odjeljcima za raščlambu aktivnosti po sudionicima. S obzirom na to da takva podjela podsjeća na odvojene staze na sportskim natjecanjima, popularno se particije nazivaju i 'plivačkim stazama' (engl. *swimlanes*).

Slika 2.4.1 prikazuje sve potrebne plivačke staze za dijagram aktivnosti kupnje proizvoda.



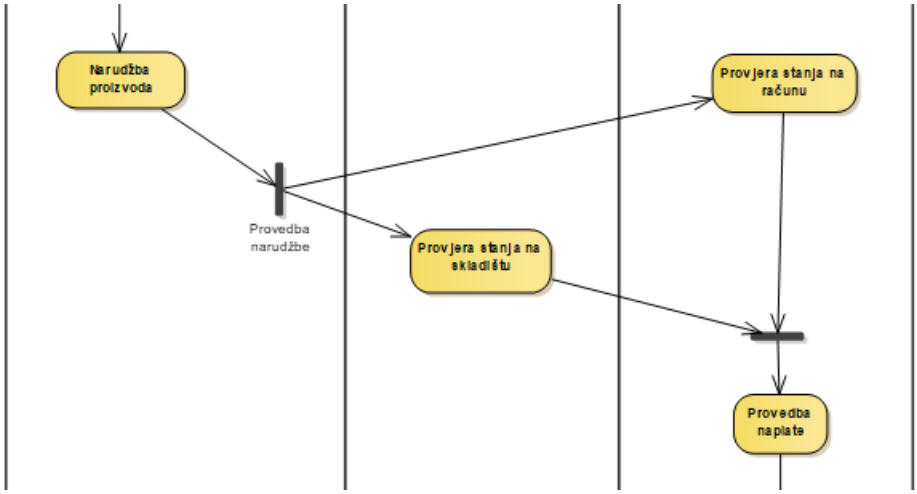
Slika 2.4.1 Postavljanje particija dijelova sustava koji sudjeluju na dijagramu

Niz aktivnosti koji slijede na početku nakon prijave korisnika u sustav uključuju pretragu proizvoda, pregled proizvoda i narudžbu proizvoda, Slika 2.4.2. Te aktivnosti se u dijagramu prikazane nakon početnog čvora, kao i ona prva, standardnim prikazom u obliku zaobljenog pravokutnika, a između njih su postavljeni usmjerene linije ili lukovi kao prijelazi između aktivnosti.



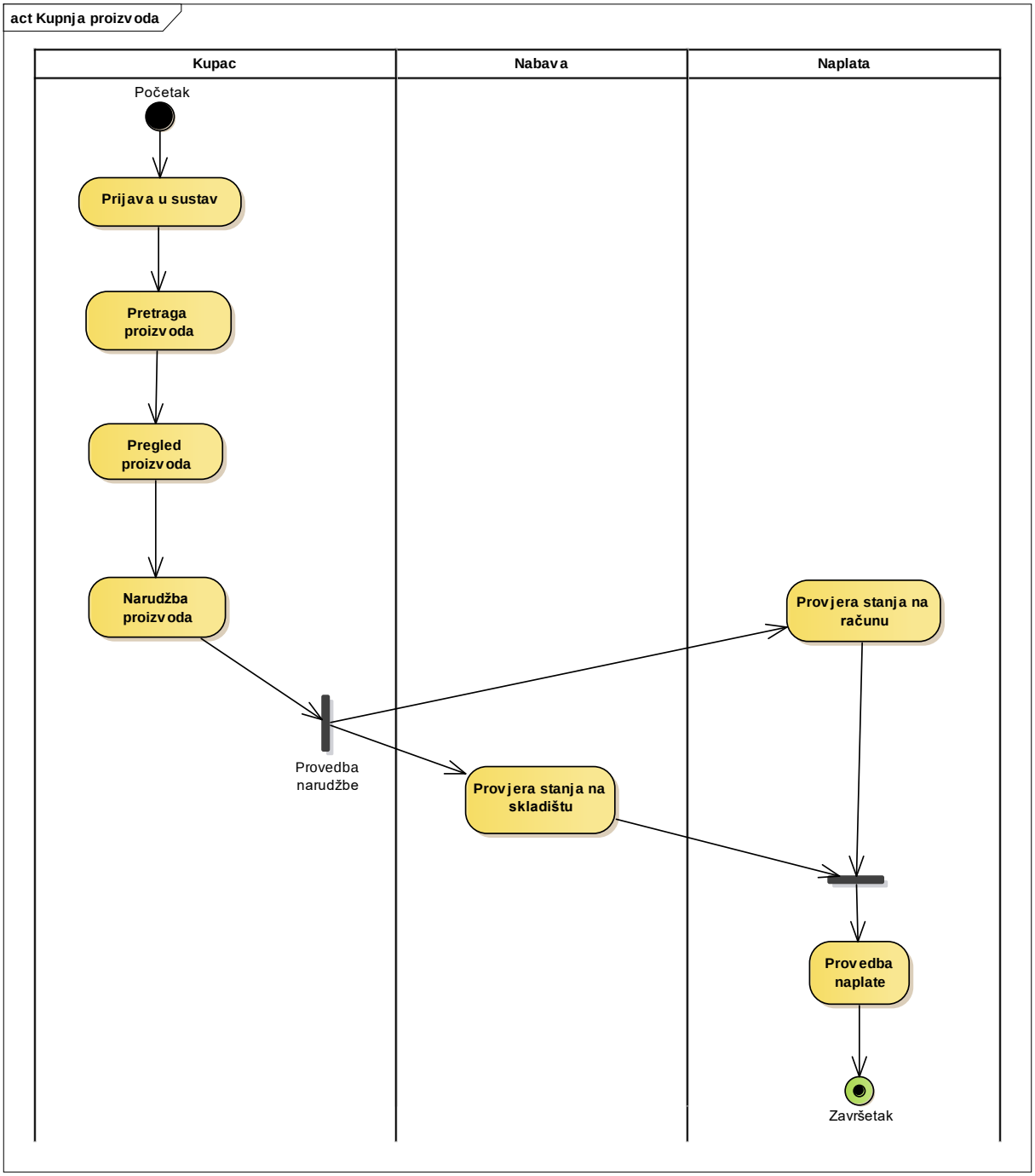
Slika 2.4.2 Postavljanje slijeda aktivnosti na dijagramu aktivnosti

Nakon što kupac pošalje svoju narudžbu događaju se provjere stanja na skladištu i stanja novčanih sredstava kupca na računu. Te provjere započinju istovremeno nakon postavljanja narudžbe, odvijaju se usporedno jedna s drugom pa je to potrebno naznačiti na dijagramu notacijom **račvanja** (engl. *fork*) i **skupljanja** (engl. *join*). Račvanje se događa u trenutku postavljanja narudžbe a skupljanje dviju usporednih grana upravljačkog toka se događa kao točka sinkronizacije prije provedbe same naplate. Račvanje i skupljanje imaju prikaz podebljane kratke linije koja može biti prikazana geometrijski uspravno ili položeno.



Slika 2.4.3 Primjer račvanja i skupljanja na dijagramu aktivnosti

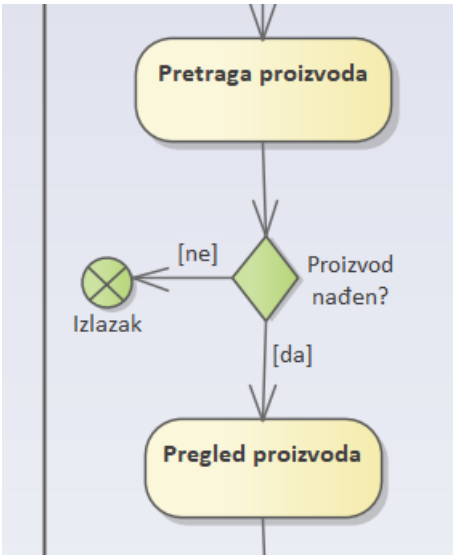
Slika 2.4.4 prikazuje dijagram aktivnosti pri očekivanom, uspješnom, postupku provedbe kupnje. Naznačen je redoslijed aktivnosti koji k tome vodi te se ne pretpostavljaju nikakvi drugi ishodi osim uspješne provedbe kupnje. Oblikovani dijagram aktivnosti kupnje proizvoda ne prikazuje sve mogućnosti prikaza upravljačkog toka aktivnosti, jer ne pokazuje i druge moguće ishode poduzete akcije kupnje proizvoda od strane kupca.



Slika 2.4.4 Konačni oblik dijagrama aktivnosti

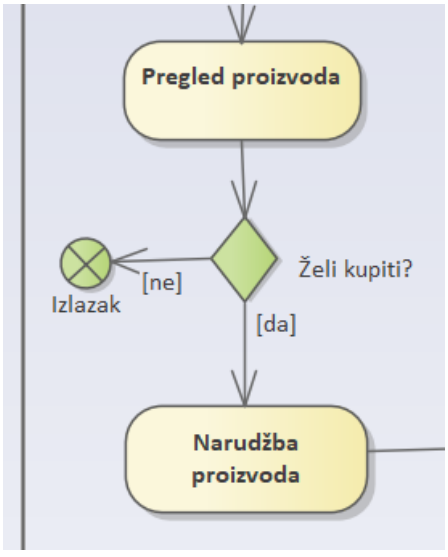
Pojedini dijelovi dijagrama mogu se detaljnije opisati s izričitijim prikazima pojedinačnih dijelova upravljačkog toka. Tako se primjerice mogu naznačiti i alternativni ili nepredviđeni tokovi

naznakom posebnih grana nakon čvorova odluke i uporabom čvorova završetaka tih zasebnih pojedinačnih tokova. Prilikom obavljanja nekih akcija moguće je posebno naznačiti mogućnost njezinog neuspjeha i prijevremenog izlaska iz aplikacije.
Nakon pretrage proizvoda moguće je da kupac nije našao onaj proizvod koji je tražio, Slika 2.4.5.



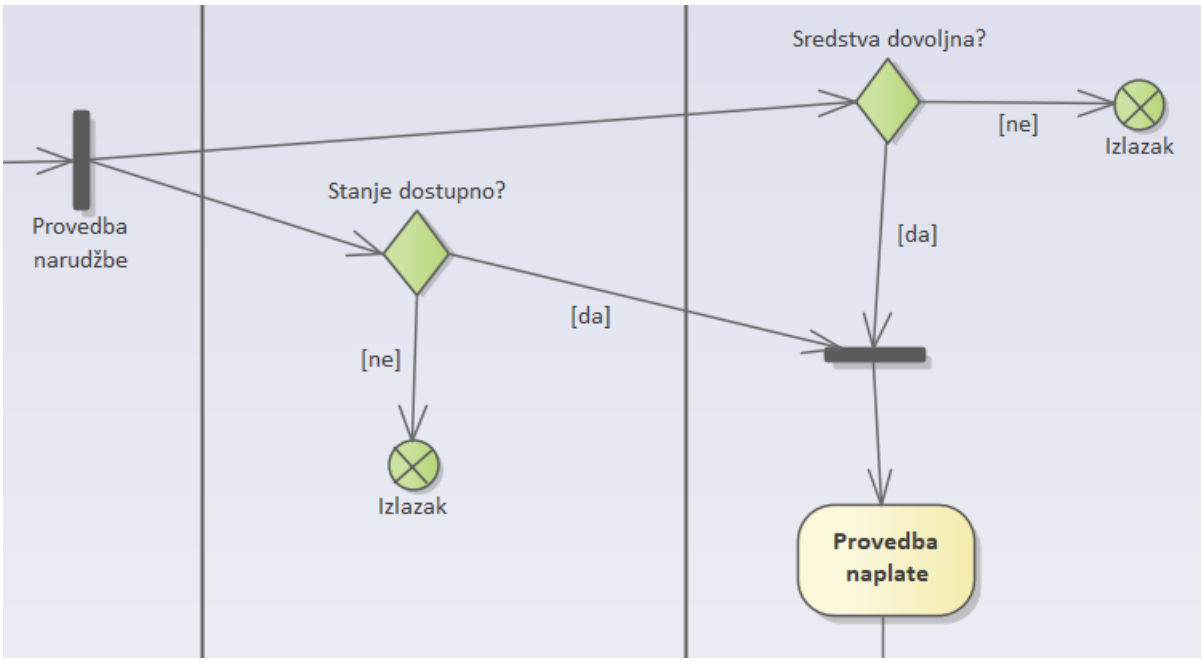
Slika 2.4.5 Prikaz situacija da proizvod nije nađen

Nakon pregleda detalja proizvoda moguće je da kupac nije zadovoljan njegovim karakteristikama pa je odustao od kupnje, Slika 2.4.6.



Slika 2.4.6 Prikaz situacije da kupac ne želi nađeni proizvod

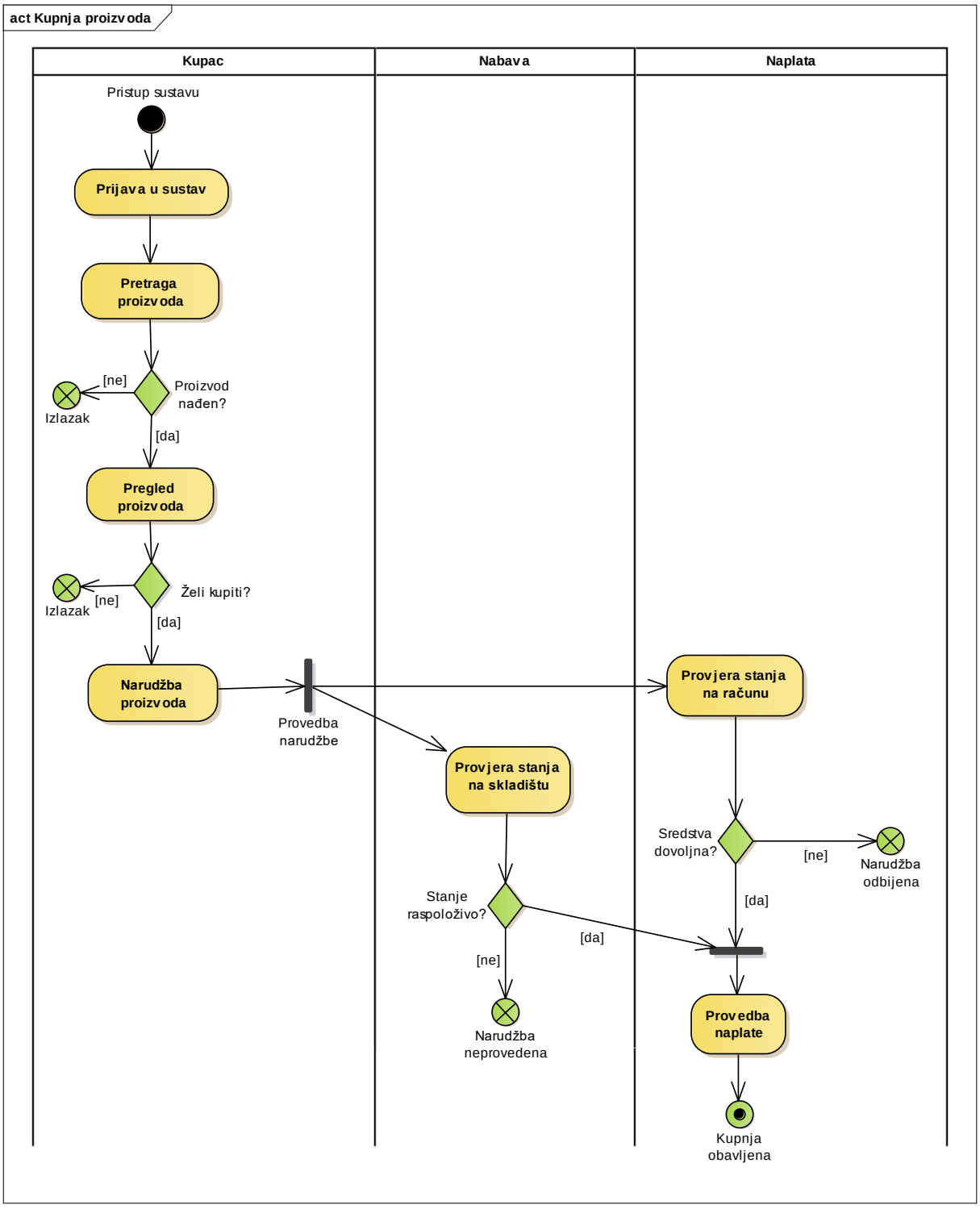
Nakon provjera stanja na skladištu trgovine i sredstava na računu kupca moguće je utvrditi da proizvod nije trenutno raspoloživ ili da kupac nema dovoljno novaca za naplatu pa se i tu događa neredoviti izlaz iz niza aktivnosti, Slika 2.4.7.



Slika 2.4.7 Prikaz opcija neuspješne kupnje proizvoda

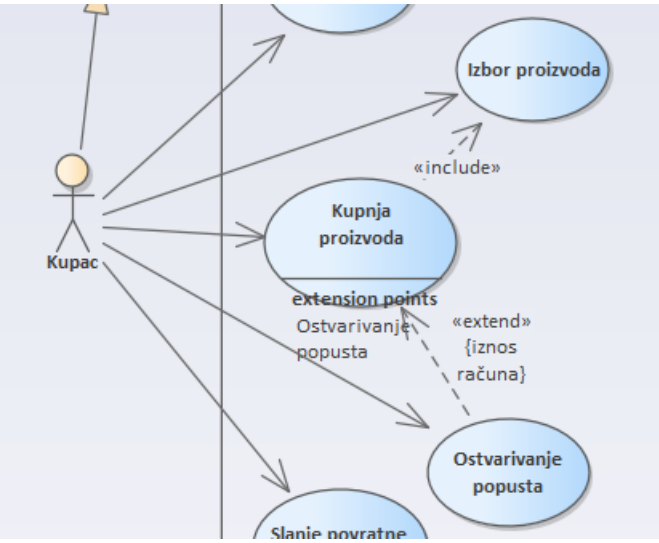
Slika 2.4.8 konačno prikazuje sveukupni dijagram aktivnosti kupnje proizvoda s dodatno razrađenim detaljima upravljačkog toka.

Svi početni i završni čvorovi su prikladno imenovani da bi se jasnije naznačila razlika između ishoda upravljačkih tokova koji s njima završavaju. Tako je npr. početni čvor označen kao 'Pristup sustavu' od strane kupca, očekivani završetak s 'Kupnja obavljena', izlazak iz aplikacije od strane kupca kao 'Izlazak' te neuspješna provedbe narudžbe kupca kao 'Narudžba neprovedena' i 'Narudžba odbijena'.



Slika 2.4.8 Detaljniji oblik dijagrama aktivnosti

U oba rješenja prikazan je primarni tok kupnje bez posebno naznačene aktivnosti izbora proizvoda niti mogućeg ostvarivanja popusta. Ne prikazuju se detalji načina izbora proizvoda niti uvjeta ostvarivanja popusta kao slučajevi uporabe koji su u dijagramu slučajeva uporabe obuhvaćeni ili proširujući, Slika 2.3.11 odnosno Slika 2.4.9.

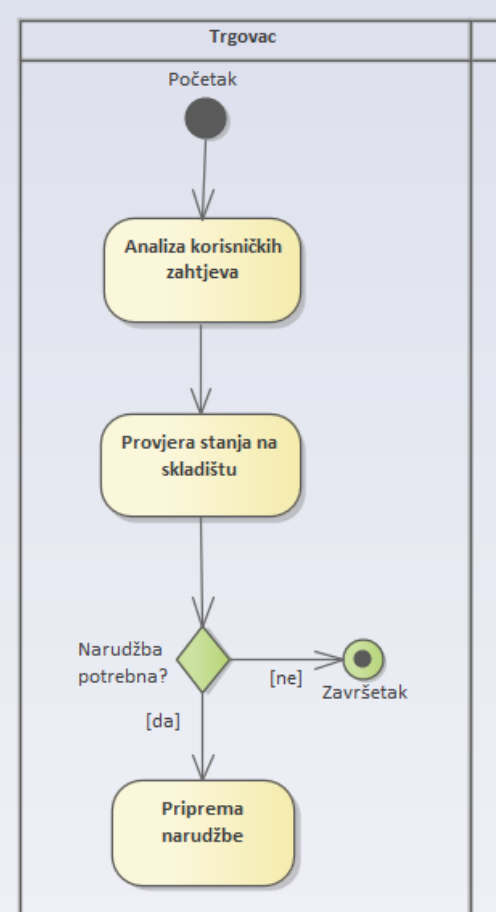


Slika 2.4.9 Isječak dijagrama slučajeva uporabe vezan uz provedbu kupnje proizvoda

S druge strane, u odnosu na referencirani dijagram slučajeva uporabe ovim dijagramima aktivnosti je naznačen neki prošireni skup aktivnosti koji obuhvaća određene slučajeve uporabe koji na dijagramu slučajeva uporabe sustava prodaje auto-dijelova nisu očigledno povezani sa slučajem uporabe kupnje proizvoda. Iako su obje vrste dijagrama ponašajne naravi postoje bitne razlike između njih. Na dijagramu slučajeva uporabe nema vremenskog uređenja između slučajeva uporabe kao što ima naznačeno vremenskog uređenje između aktivnosti. Dijagram aktivnosti prikazuje upravljački tok nekog poslovnog procesa, a dijagram slučajeva uporabe daje pregled funkcionalnih zahtjeva sustava.

Narudžba novih proizvoda

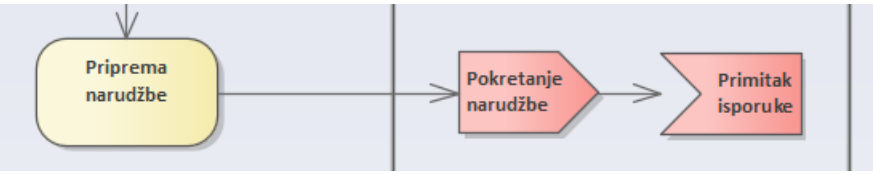
Niz akcija pri narudžbi novih proizvoda od strane trgovca predstavljaju aktivnosti trgovca, podsustava nabave i administratora, Slika 2.4.12.
Pretpostavka je da narudžba novih proizvoda od strane trgovca na početku uključuje niz koraka. Na temelju aktivnosti analize korisničkih zahtjeva i provjere stanja na skladištu trgovac utvrđuje je li potrebna narudžba novih proizvoda te, ako jest, kojih proizvoda u kojoj količini. U takvom slučaju trgovac priprema narudžbu te je daje podsustavu nabave na provedbu. Sustav nabave nakon isporuke narudžbe šalje potvrdu isporuke trgovcu koji ju zaprima i daje u svrhu provedbe nove evidencije unutar cjelokupnog sustava prodaje administratoru.
Koraci koji prethode kontaktiranju službe nabave od strane trgovca oblikuju se kao niz aktivnosti ili akcija, Slika 2.4.10.



Slika 2.4.10 Aktivnosti trgovca prilikom narudžbe novih proizvoda

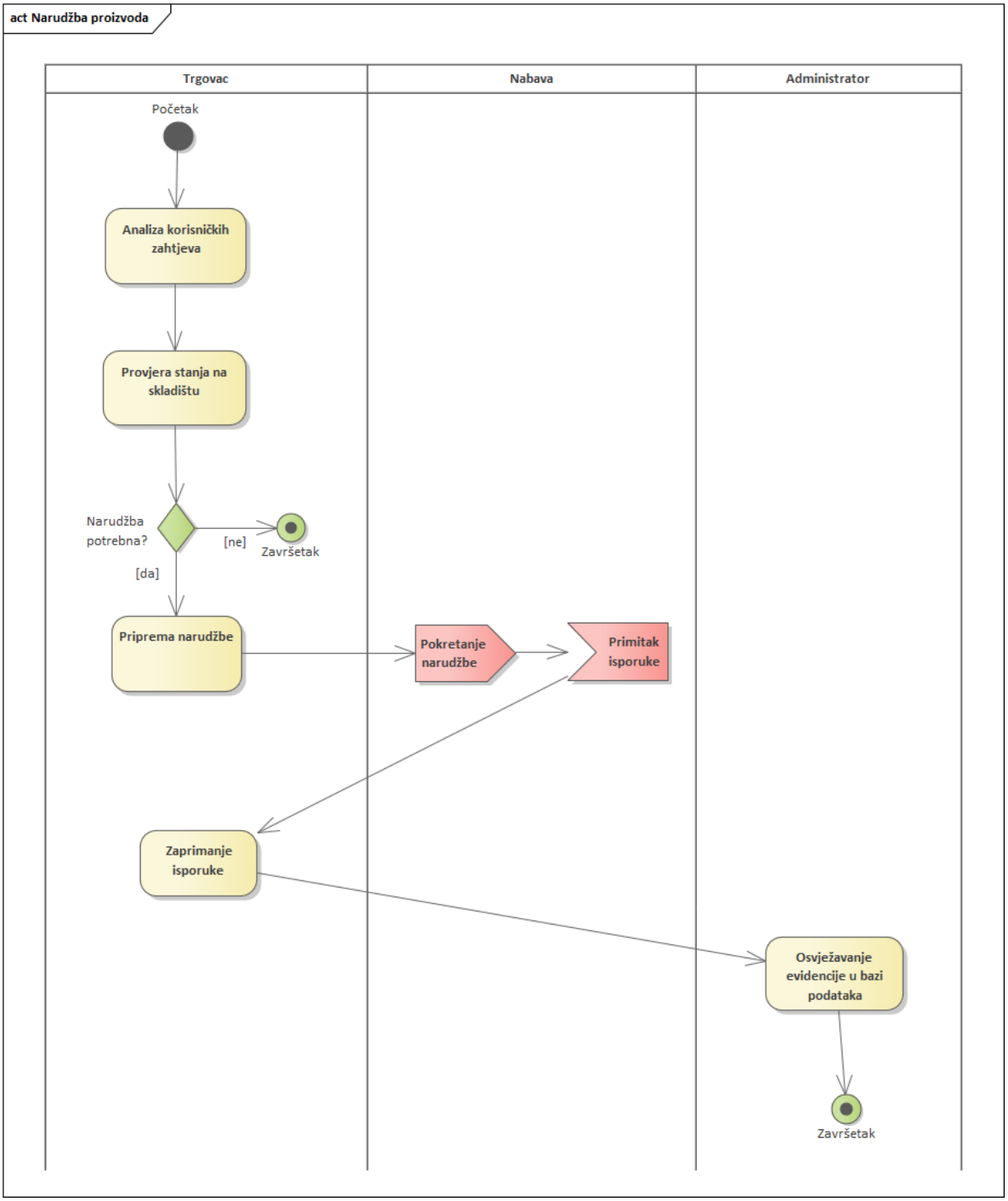
Nakon pripreme narudžbe ista se pokreće kontaktiranjem službe nabave. Na postupak nabave se sa strane web trgovine gleda kao neovisnu uslugu ili vanjski proces. Na dijagramu se stoga ne prikazuju detaljno akcije službe nabave prilikom izvršenja postupka nabave već je bitan samo konačni ishod nabave, tj. da je naručena roba isporučeno. Takav pogled se na dijagramu modelira uporabom upravljačkih čvorova slanja signala i prihvaćanja signala ili događaja koji predstavljaju mehanizam poticanja i primanja akcija i odgovora od strane nekog vanjskog

entiteta koji nije prikazan u dijagramu. Notacija obje vrste navedenih čvorova ima načelno pravokutan oblik gdje je jedna od stranica zamijenjena usmjerenim izbočenjem ili udubljenjem oblika strelice čime se naglašava radi li se o signalu koji se šalje ili prima, Slika 2.4.12.



Slika 2.4.11 Primjer primjene signala u dijagramu aktivnosti

Nakon izvršenja narudžbe te primitka isporuke trgovcu koji je narudžbu pokrenuo, narudžba se zaprima i evidentira novo stanje u bazi podataka čime se zaokružuje prikaz postupka narudžbe novih proizvoda uporabom dijagrama aktivnosti, Slika 2.4.12.

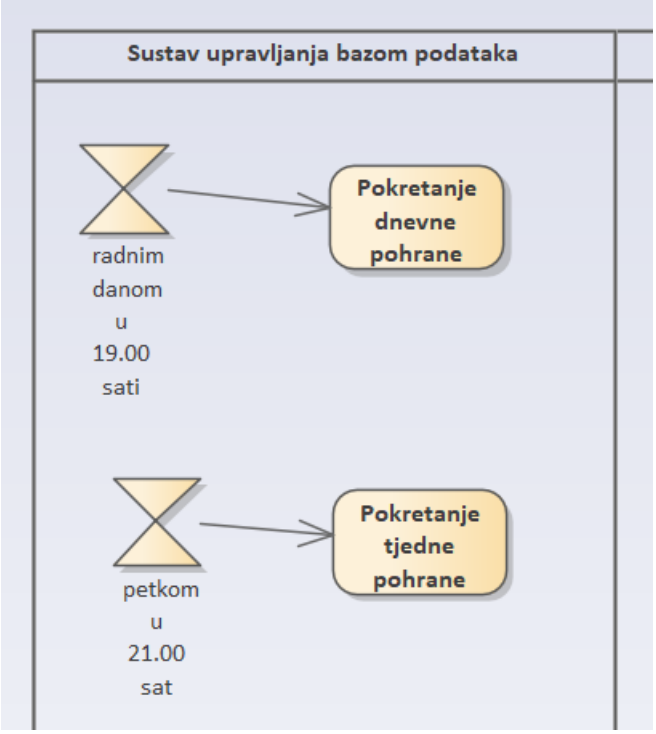


Slika 2.4.12 Dijagram aktivnosti narudžbe novih proizvoda

Upravljački čvorovi slanja signala „'Pokretanje narudžbe'“ i prihvaćanja događaja „'Primitak isporuke' zapravo predstavljaju nadomjestak za semantiku „događaja“ koji nije generički pretpostavljen za dijagram aktivnosti a koji su inače temeljna pretpostavka za dijagram stanja gdje potiču prijelaze između stanja, vidjeti poglavlje 2.7. Postoji i treća vrsta takvog tipa čvora za dijagram aktivnosti koja će biti prikazana u primjeru koji slijedi.

Redovita pohrana podataka

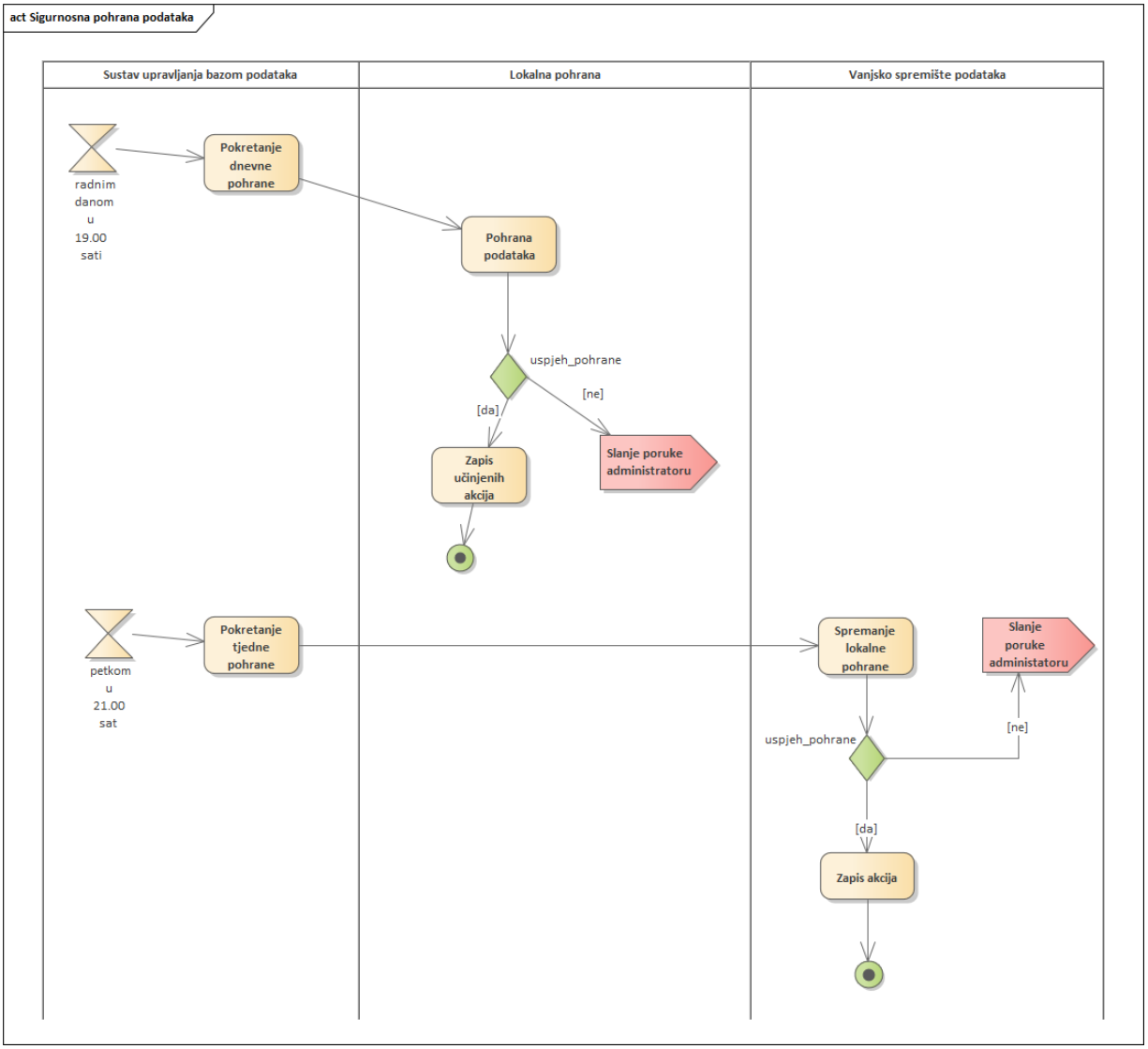
Pretpostavka je da se redovita sigurnosna pohrana podataka (engl. *backup*) radi automatski u točno određenim vremenskim razmacima što je poticano uređenim vremenskim mehanizmom. Takvo vremenski vođeno poticanje akcija pohrane se u dijagramu aktivnosti prikazuje upravljačkim čvorom vremenskog prihvaćanja notacijom nalik pješčanom satu, Slika 2.4.13.



Slika 2.4.13 Primjer primjene vremenskog signala u dijagramu aktivnosti

Svakodnevna lokalna pohrana podataka se automatski pokreće svakog radnog dana u 19 sati, a tjedna spremljene lokalne pohrane šalje u vanjsko spremište podataka svakog petka u 21 sat. Te aktivnosti odnosno akcije ne pokreće nitko izričito već je sustav za upravljanje bazom podataka podešen ili tempiran na taj način.

Tokovi akcija za dvije neovisne aktivnosti pohrane su zasebno modelirani te završavaju zapisom učinjenih akcija u slučaju uspješne pohrane ili slanje poruke administratoru u slučaju neuspješne pohrane, Slika 2.4.14.



Slika 2.4.14 Dijagram aktivnosti sigurnosne pohrane podataka

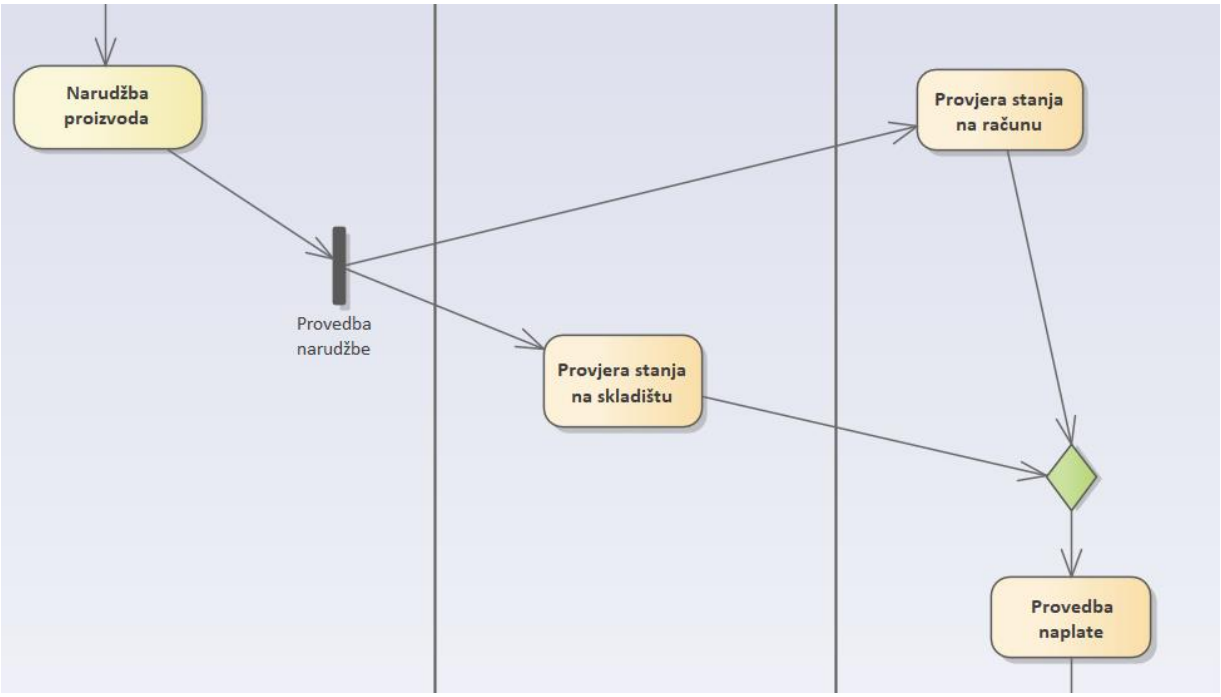
Upotrijebljeni su upravljački čvorovi slanja signala administratoru, inače tok dijagram završava sa završnim čvorova. Umjesto početnih čvorova na dijagramu su upotrijebljeni čvorovi vremenskog signala kao inicijatori aktivnosti na dijagramu s obzirom na to da dijagram može imati proizvoljan broj početnih čvorova ali ih ne mora imati uopće.

Česte pogreške

U praksi modeliranja primijećene su neke češće ponavljane pogreške ili nerazumijevanja u izradi dijagrama aktivnosti, a vezano su uz sintaksu i semantiku uvjetnog grananja i prikaza usporednih tokova.

Miješanje semantike račvanja i skupljanja sa semantikom uvjetnog grananja

Kombiniranje notacije grananja s račvanjem i skupljanjem je moguća pogreška: račvanjem se započinju usporedni tokovi i to se ne može kombinirati s čvorom spajanja koji se koristi za spajanje različitih grana koje se međusobno isključuju nakon čvora odluke. Čvor odluke nema značenje početka toka usporednih tokova kao što ni čvor spajanja nema značenje kraja toka usporednih tokova. Čvor odluke zajedno sa čvorom spajanja predstavlja semantiku uvjetnog grananja gdje se pojedine grane međusobno isključuju, a čvor račvanja zajedno sa čvorom skupljanja ima značenje prikaza usporednih tokova koji se istovremeno izvršavaju. Slika 2.4.15 prikazuje primjer nedozvoljenog kombiniranja te dvije semantike.

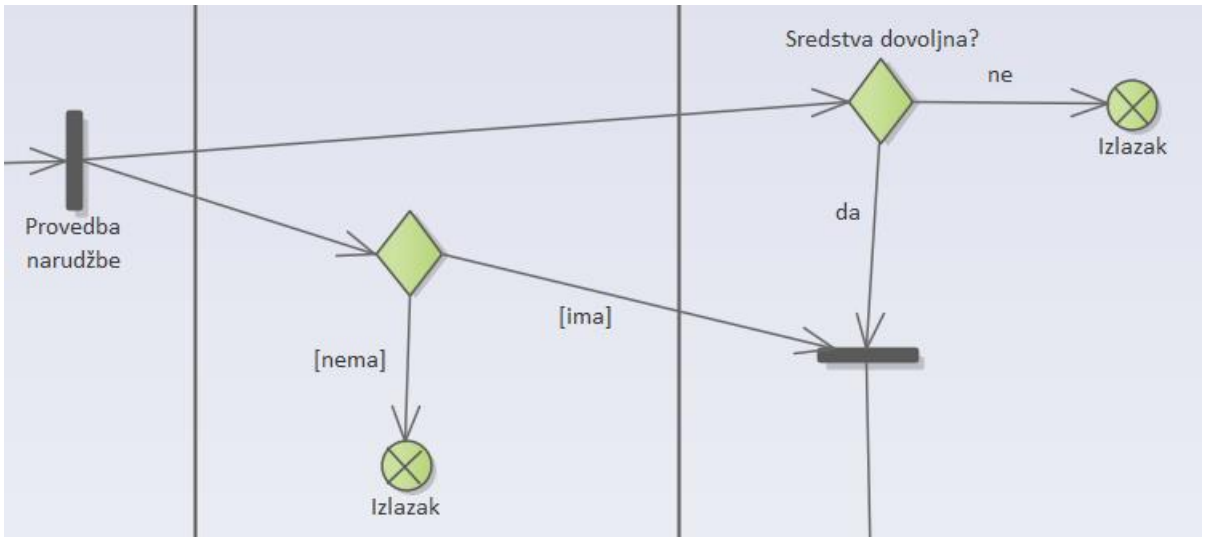


Slika 2.4.15 Primjer miješanja semantike račvanja sa semantikom uvjetnog grananja

Prilikom provedbe narudžbe usporedno se obavljaju akcije provjere stanja na skladištu i stanja na računu kupca te je neispravno ta dva toka na kraju povezati čvorom spajanja kao da se radi o uvjetnom grananju.

Neispravna notacija uvjeta kod uvjetnog grananja

Nenavođenje ili kriva notacija uvjeta grananja i izlaznih grana čvora odluke: potrebno je naznačiti unutar uglatih zagrada koji su uvjeti izvođenja pojedinih grana nakon čvora odluke. Također, treba jasno navesti koji je uvjet grananja, tj. koje se pitanje postavlja na čvoru odluke. Slika 2.4.16



Slika 2.4.16 Primjer neispravne notacije kod uvjetnog grananja

U primjeru na slici u jednom slučaju radi se o tome da nije jasno navedeno o kojoj se konkretno odluci prije grananja radi već su samo navedeni uvjeti izvođenja pojedinih grana: '[ima]' i '[nema]'. U slučaju drugog čvora odluke jasno je navedeno pitanje na kojem se temelji odluka o izboru izlazne grane, tj. 'Sredstva dovoljna?', ali za izlazne grane nije ispravno navedena notacija uvjeta pod kojim se izvode jer je izostavljena notacija uglatih zagrada.

2.5. Dijagram slijeda

Dijagrami slijeda ili sekvencijski dijagrami (engl. *sequence diagrams*) detaljiziraju pojedine slučajeve uporabe na način da prikazuju komunikaciju, ili razmjenu poruka, koja se odvija unutar pojedinog slučaja između aktora i objekata u sustavu. Pritom naglašavaju vremenski slijed razmjene poruka te prikazuju odgovornosti i suradnju elemenata sustava do ostvarenja pojedinog dijela funkcionalnosti.

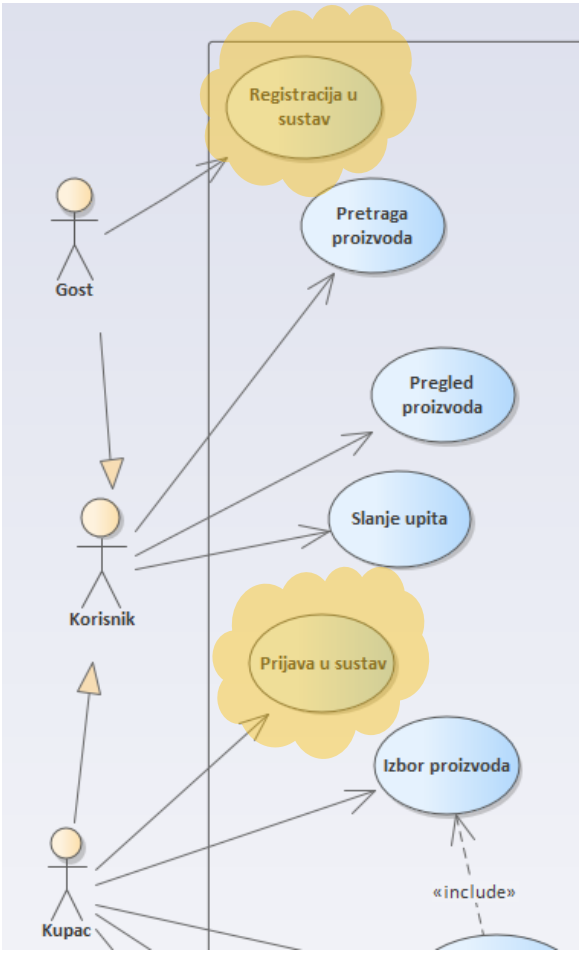
Ova vrsta dijagrama spada u interakcijske ponašajne dijagrame zajedno s dijagramima komunikacije koji se opisuju u poglavlju nakon ovoga. Za pojedine sudionike, aktore i objekte sustava, prikazuju se njihove **životne crte** na kojima se označavaju razdoblja njihove aktivnosti koje se oblikuju međusobnim **razmjenama poruka**. Dijagram ima vizualno naglašeno vremensko uređenje takvih interakcija između sudionika koji su predstavljeni svojim životnim crtama te se redoslijed izmjena poruka prilikom interakcija započinje i prati od gore prema dolje i slijeva nadesno. Sva interakcija se ostvaruje izmjenama poruka koje predstavljaju pozive dostupnih metoda sudionika interakcije. Poruke su stoga obilježene nazivom, argumentima i povratnom vrijednošću te tipovima podataka s kojima se upravlja, i uvjetima pod kojima se šalju. Razlikuju se **poruke sinkronog i asinkronog tipa** gdje se kod prvih očekuje i čeka odgovor od strane onog kome je poruka poslana, dok nakon slanja drugih pošiljatelj šalje daljnje poruke neovisno o tome je li primio odgovor. Postoji posebna notacija kojom se označavaju **povratne poruke**. Poruke se mogu dodatno označavati rednim brojeva s više razina pobrojavanja da bi se lakše prepoznao povezani slijed poruka, koje su početne poruke i koje poruke izravno proizlaze iz njih.

Razmjena poruka se dodatno opisuje blokovima **fragmenata** kao elementima kojima se oblikuje i prikazuje upravljački tijek interakcije. Tako primjerice postoje fragmenti vrste uvjetnog grananja (***alt***) i petlje (***loop***) prilikom slanja i razmjene poruka, kao i opcionalne interakcije (***opt***).

U potpoglavljima koji slijede uporabom dijagrama slijeda razlažu se pojedini slučajevi uporabe ili skupine povezanih slučajeva uporabe prikazujući interakciju između aktora i objekata sustava.

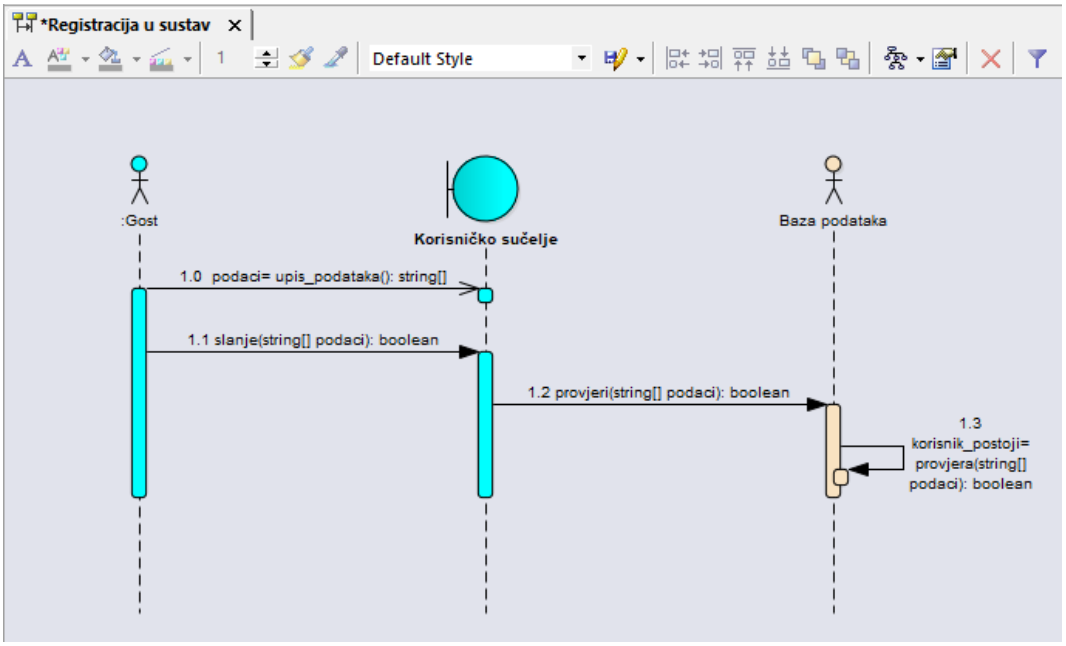
Slučaj uporabe 'Prijava u sustav' (UC4, UC5)

Dijagram slijeda je namijenjen detaljnijem prikazu interakcija koje se događaju unutar pojedinih slučajeva uporabe, a moguće je i prikazivati i semantičke povezane interakcije više srodnih ili povezanih slučajeva uporabe. Primjerice, registracija korisnika vrste 'Gost' i prijava korisnika vrste 'Kupac' u sustav se mogu kombinirati unutar istog dijagrama slijeda, Slika 2.5.1. Iako u dijagramu slučajeva uporabe nisu izravno vezano posebnom vrstom veze postoje logičke veze između njih. Ako korisnik nije registriran ne može se niti prijaviti u sustav već se najprije mora registrirati da bi napravio prijavu.



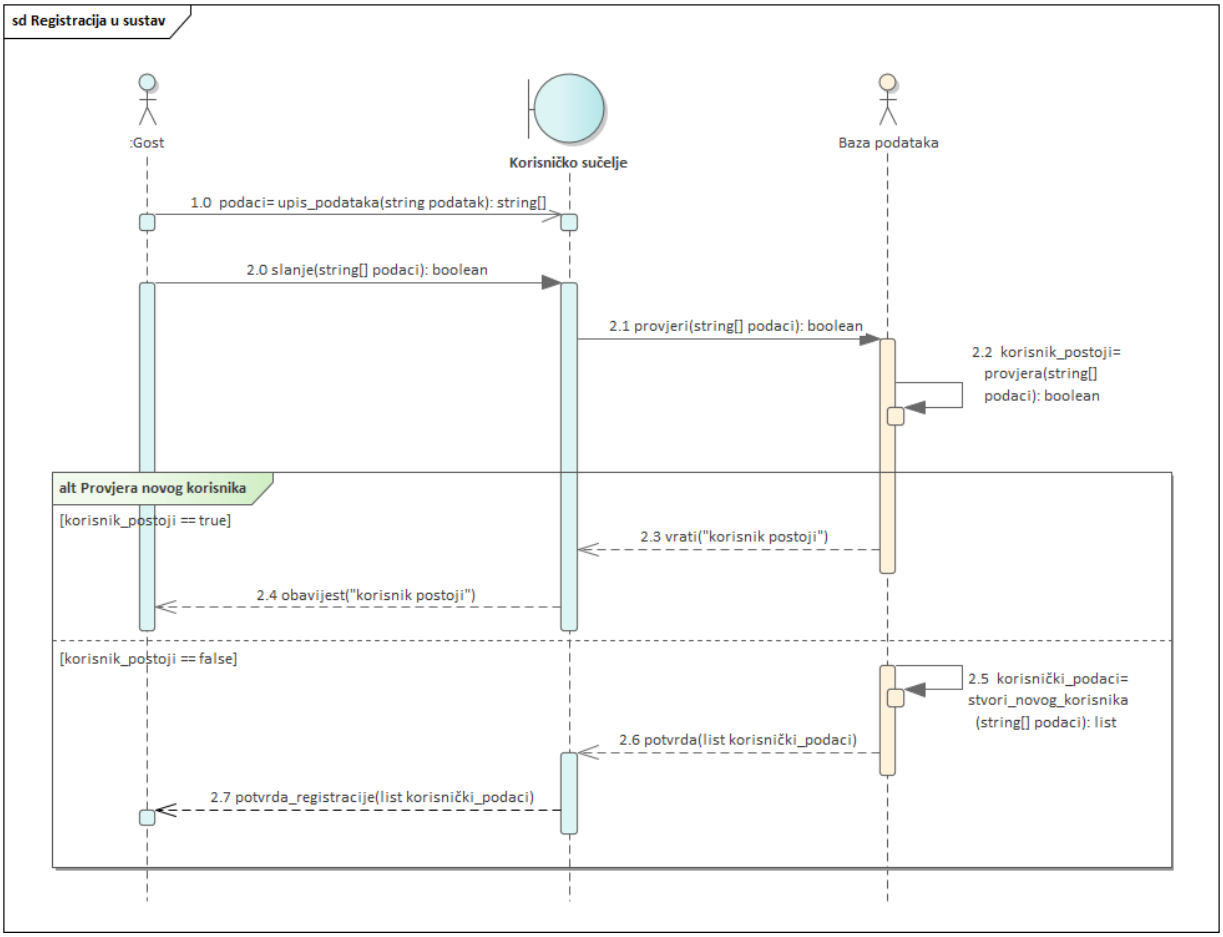
Slika 2.5.1 Kontekst slučajeva uporabe registracije i prijave korisnika u sustav

Najprije se zasebno oblikuje dijagram slijeda registracije u sustav gdje se nakon upisa podataka za registraciju u web sučelje ti isti podaci prosljeđuju na provjeru u bazu podataka, Slika 2.5.2.



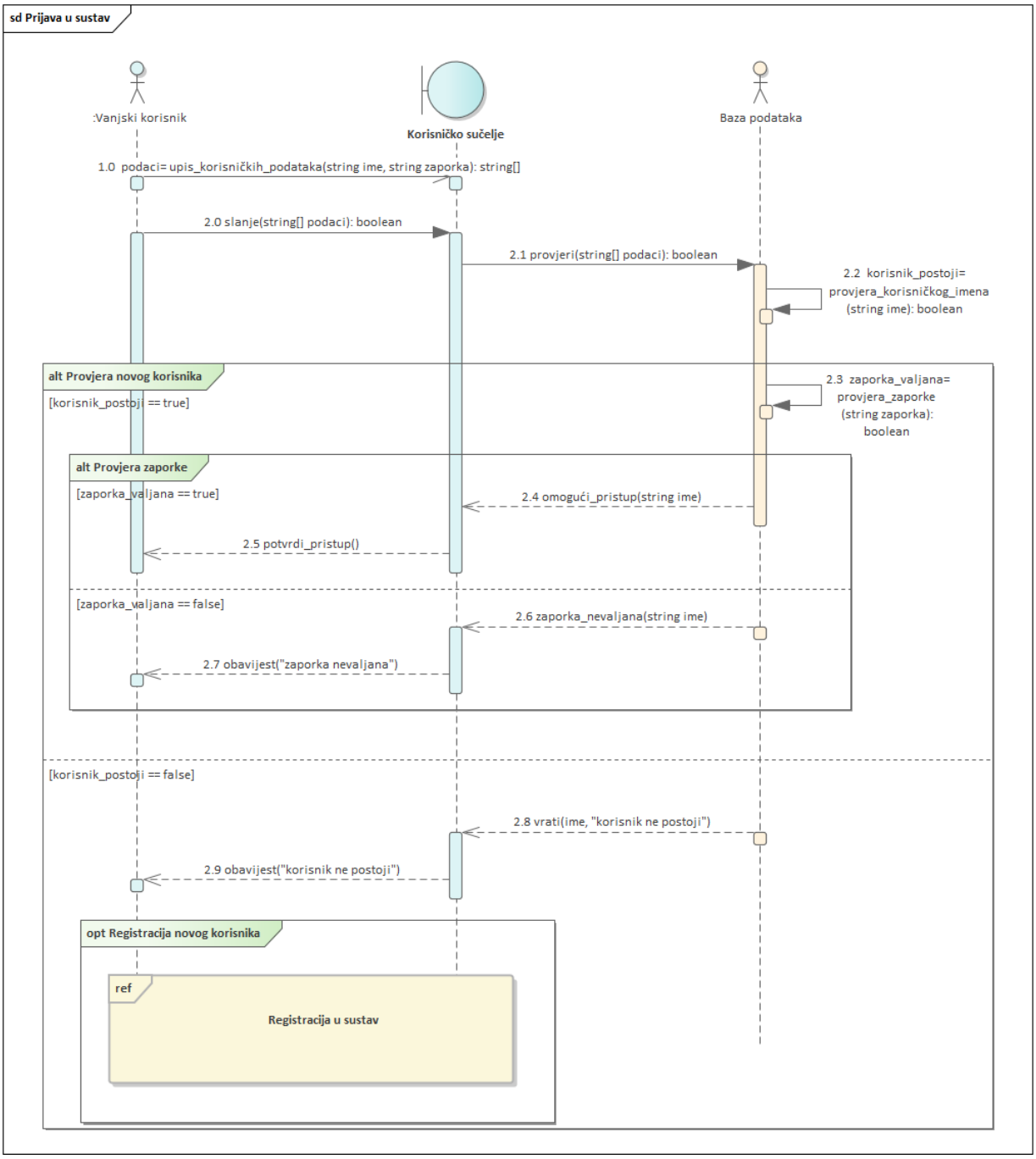
Slika 2.5.2 Registracija korisnika u sustav – početak oblikovanja

Unos podataka se formalno modelira slanjem poruke korisničkom sučelju naziva 'upis_podataka' koje se događa prilikom utipkavanja podataka u dani obrazac. Upisuju se podaci općeg tekstualnog tipa *string* te se isti pohranjuju u povratnu varijablu 'podaci'. Poruka je asinkronog tipa jer korisnik izričito ne očekuje odgovor sustava sve dok posebnom porukom sinkronog tipa naziva 'slanje' ne pošalje zahtjev za registraciju s unesenim podacima.
Nakon toga, pretpostavlja se da klijentski dio aplikacije predstavljen instancom korisničkog sučelja šalje upit s primljenim podacima pozadinskom dijelu aplikacije unutar kojeg se nalazi baza podataka aplikacije. Provjera postojećih korisnika u bazi podataka oblikuje se „porukom samom sebi“ ili unutarnjom procedurom unutar tog dijela aplikacije. Odgovor aplikacije korisniku je takav da, ovisno o tome nalazi li se već spomenuti korisnik u evidenciji korisnika (tj. je li već registriran u prošlosti) moguća su dva ishoda (može se pretpostaviti da je ključ jedinstvenosti npr. adresa e-pošte s kojom se korisnik registrira). Ako korisnik već postoji, potrebno ga je o tome obavijestiti, a ako nije, potrebno ga je stvoriti kao novog korisnika.
U tu svrhu definira se fragment uvjetnog grananja *alt* s dvije grane odnosno s dva zasebna odjeljka u kojima je naznačeno pod kojim uvjetima se koji daljnji slijed izvršava: korisniku se ispisuje poruka o uspješnoj ili neuspješnoj registraciji. Nakon što se definira slijed poruka u granama kako je pretpostavljeno, dobiva se konačni oblik dijagrama slijeda za slučaj uporabe registracije korisnika u sustav, Slika 2.5.3. Na njemu je naknadno još odvojeno pobrojavanje poruka upisa podataka za registraciju u sučelju (počinje s *1.0*) od poruka slanja upisanih podataka u sustav (počinje s *2.0*).



Slika 2.5.3 Dijagram slijeda registracije korisnika

Dijagram slijeda prijave u sustav slično kao i kod registracije korisnika započinje upisom i slanjem podataka za prijavu sve do provjere u bazi podataka, Slika 2.5.4.

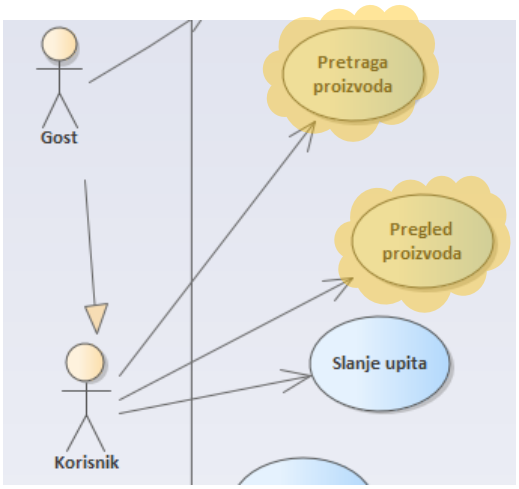


Slika 2.5.4 Dijagram slijeda prijave korisnika u sustav

Ako korisnik postoji, provjerava se upisana zaporka te se ovisno o valjanosti zaporke korisniku odobrava ili ne odobrava prijava u sustav. Ako korisnik ne postoji, o tome ga se obavještava uz mogućnost registracije. Mogućnost registracije se prikazuje kao ponuđena opcija za provođenje slučaja uporabe 'Registracija korisnika' koji se u ovom dijagramu slijeda prikazuje kao referenca (***ref***, kao poseban tip elementa na dijagramu) za prethodno izgrađeni dijagram slijeda (Slika 2.5.3) unutar fragmenta tipa *opt*.

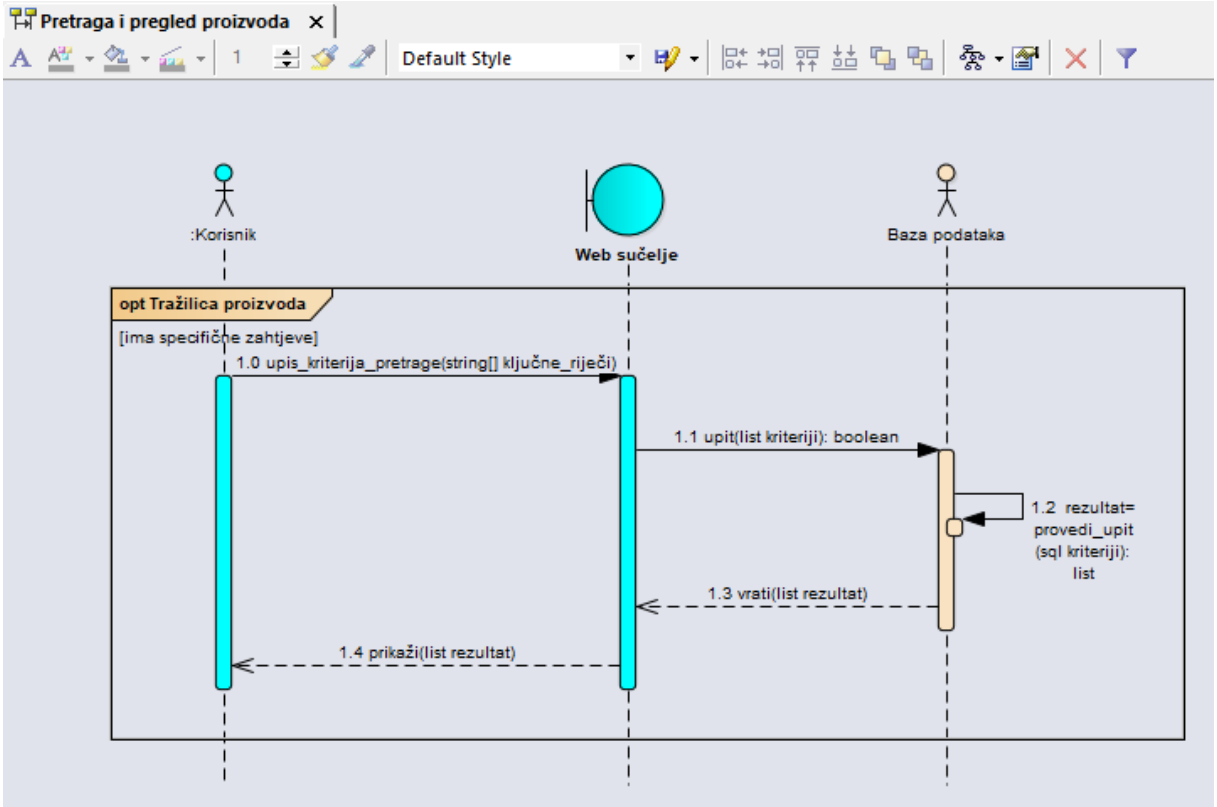
Slučaj uporabe 'Pregled proizvoda' (UC1, UC2)

Slučaj uporabe 'Pregled proizvoda' (UC2) logički je povezan sa slučajem uporabe 'Pretraga proizvoda' (UC1) jer korisnik u nekim situacijama treba najprije pretražiti proizvode da bi pogledao detalje željenog i nađenog proizvoda, Slika 2.5.5. Zato će se u primjeru jednog dijagrama slijeda pokazati detalji interakcije korisnika sa sustavom pri pretrazi i pregledu proizvoda.



Slika 2.5.5 Isječak dijagrama slučajeva uporabe koji se odnosi na pretragu i pregled proizvoda

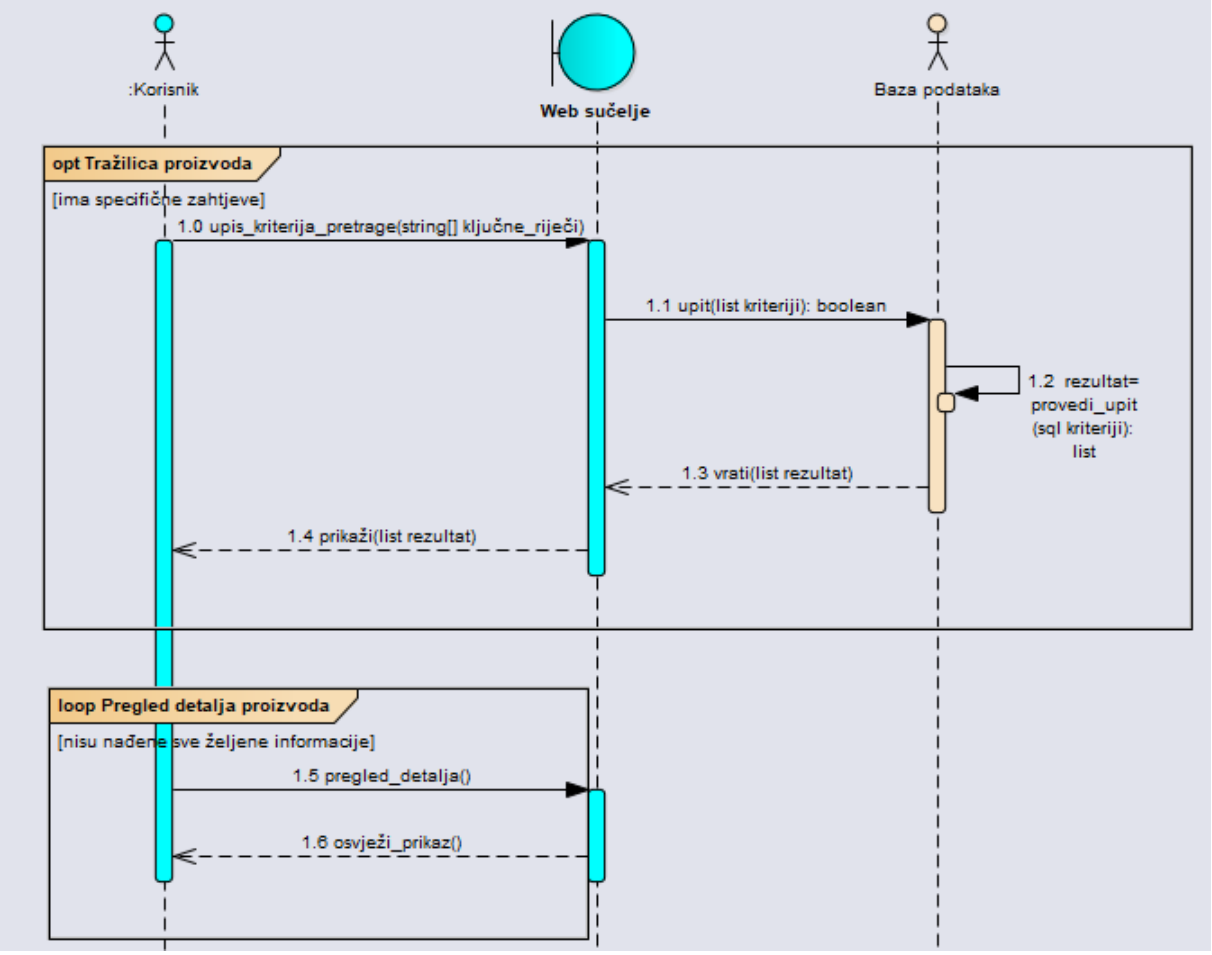
Ako se pretpostavi da korisnik na početku u web sučelju ima izlistane ili na neki drugi način prikazane pojedine proizvode na izbor, on može klikom na pojedini proizvod dobiti detalje. Isto tako, korisnik bi trebao imati mogućnost posebne pretrage proizvoda upisom ključnih riječi u polje za to pripremljene tražilice. To je opcija koju korisnik može upotrijebiti ne bi li pronašao upravo onaj proizvod koji traži. Zato se pretraga proizvoda stavlja unutar fragmenta *opt* u dijagramu kao interakcija koja je opcionalna i događa se pod uvjetom specificiranim unutar uglatih zagrada. U ovom konkretnom slučaju unutar tog fragmenta korisniku se prikazuju funkcionalnosti vezane uz pretrage: unos ključnih riječi, upit na bazu podataka i povratni odgovor, Slika 2.5.6.



Slika 2.5.6 Primjer fragmenta opt *na dijagramu slijeda*

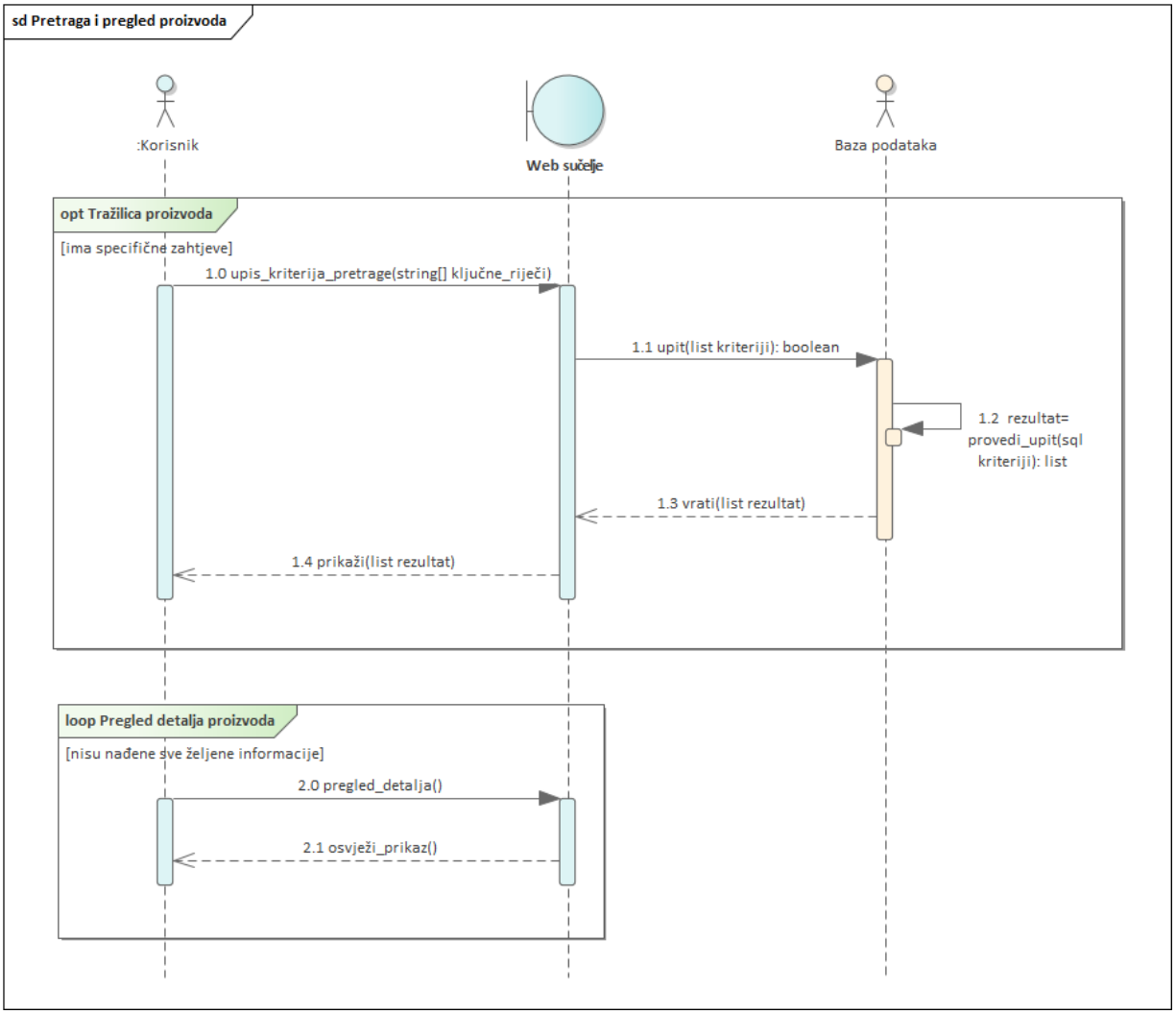
Korisnik sve akcije obavlja kroz sučelje web aplikacije pa je zato u dijagram stavljen i odgovarajući objekt koji će predstavljati web sučelje aplikacije. Sve akcije pretrage i pregleda temelje se na upitima i odgovorima iz baze podataka pa je u dijagram stavljen i aktor 'Baza podataka'. Korisnik postavlja kriterije pretrage po ključnim riječima i akcijom potvrde zadaje pretragu, što se može modelirati kao poziv metode ili poruka pozadinskoj aplikaciji koja se nalazi iza pristupne aplikacije web sučelja. Može se pretpostaviti da se na temelju upisa korisnika oblikuje lista kriterija za upit nad bazom podataka. Taj upit se onda na dijagramu oblikuje kao poruka od web aplikacije prema bazi podataka nakon čega baza podataka provodi zadani upit i vraća rezultate upita koji se prikazuju korisniku unutar web sučelja. To je svakako moguće prikazati još detaljnije na način da se izdvojeno prikažu svi slojevi web aplikacije, tj. da se odvojeno prikažu logika pristupne (engl. *frontend*) od pozadinske aplikacije (engl. *backend*). Upit na bazom podataka je predstavljen porukom naziva 'upit' od strane web aplikacije s kriterijima pretrage kao parametrima poruke i povratnom vrijednošću tipa *boolean* čime se simbolizira uspješan ili neuspješan prihvat upita od strane baze podataka. Baza podataka provodi upit pozivom svoje unutarnje procedure koja kao povratnu vrijednost vraća neki 'rezultat'. Ta vrijednost se upotrebom povratnog parametra vraća korisniku putem web aplikacije i sučelja kroz dvije povratne poruke: jedne od baze podataka web aplikaciji i druge koja predstavlja prikaz rezultata u web sučelja korisnika.

Nakon što je niz proizvoda izlistan u web sučelju, korisnik može zatražiti prikaz detalja za pojedini proizvod sve dok nije dobio sve potrebne informacije, Slika 2.5.7. Tu se koristi fragment vrste *loop* koji predstavlja semantiku izvršavanja u petlji da se naznači da korisnik opetovano pretražuje detalje proizvoda.



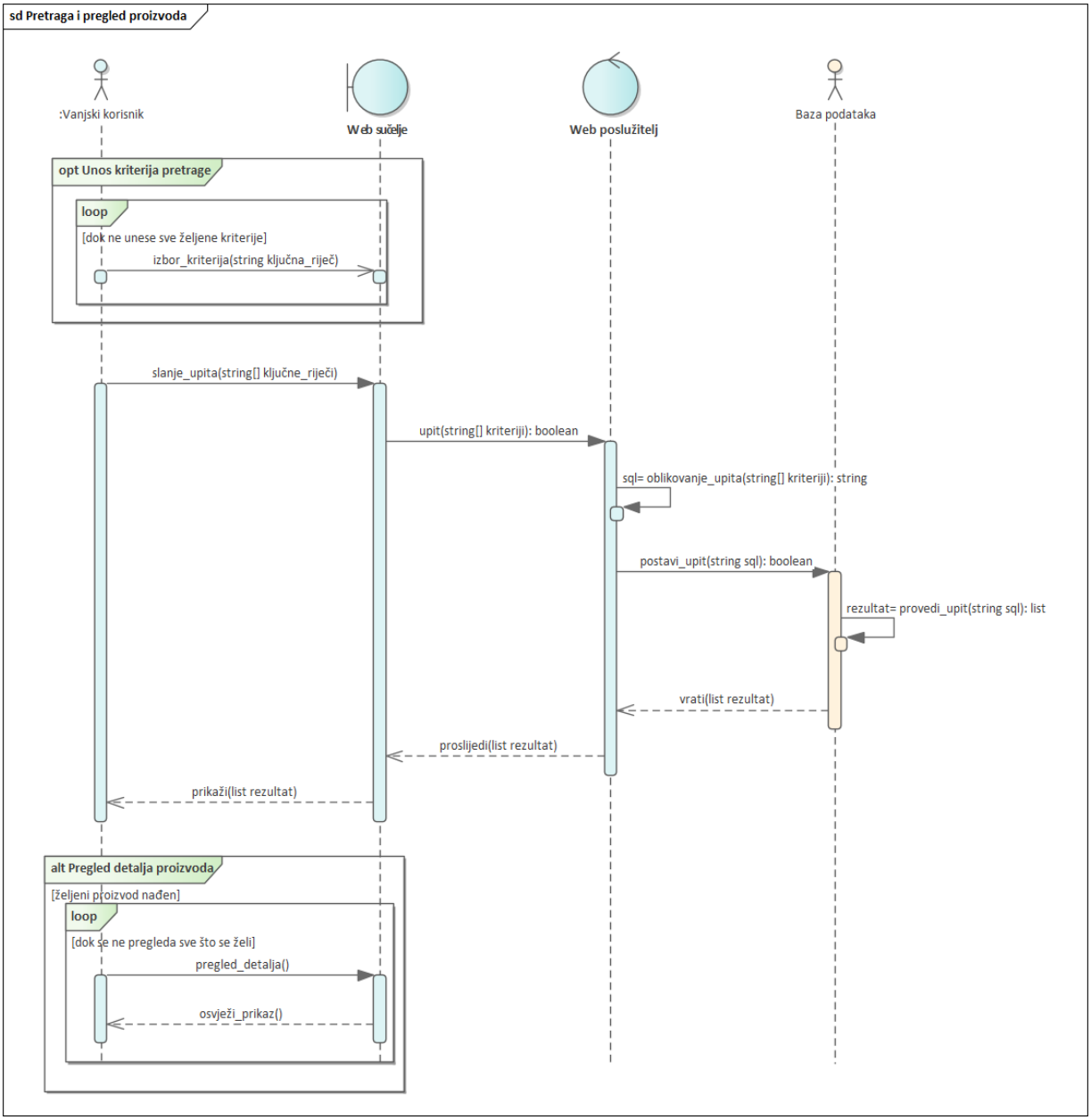
Slika 2.5.7 Dijagram slijeda s pretragom i pregledom detalja

Pregled detalja proizvoda predstavlja jednu logičku odvojenu cjelinu koja se može promatrati zasebno te se to i vizualno na dijagramu može naglasiti. S porukom 'pregled_detalja()' kao da započinje jedna zasebna skupina aktivnosti pa je uputno tome prilagoditi i pobrojavanje slijeda poruka koje je na dijagramu podešeno da ide linearno, tj. s oznakama *1.0*, *1.1*, …, do *1.6*. Ako se želi, to se može podesiti na način da se prikaže kako pregledom detalja proizvoda započinje jedan zaseban dio interakcije korisnika sa sustavom, odnosno jedna nova skupina poruka na dijagramu koja započinje pobrojavanje poruka s 2.0. Slika 2.5.8 tako prikazuje primjer potpuno oblikovanog dijagrama slijeda za slučajeve uporabe 'Pretraga proizvoda' i 'Pregled proizvoda' gdje su detalji interakcije za ta dva slučaja uporabe dodatno razdvojeni zasebnim pobrojavanjem poruka.



Slika 2.5.8 Primjer dijagrama slijeda slučajeva uporabe "Pretraga proizvoda" i "Pregled proizvoda"

Nadalje, moguće je dijagram slijeda za ova dva slučaja uporabe još detaljnije prikazati, Slika 2.5.9. Npr. moguće je naglasiti pristup korisnika web sučelju i odvojiti upis kriterija od postavljanja upita. Unutar bloka *opt* 'Unos kriterija pretrage' posebno je naglašen iterativni unos kriterija kao obilježja poruke što je na dijagramu prikazano oznakom '*' ispred naziva poruke 'izbor_kriterija'. Također, ta poruka je definirana kao poruka asinkrone vrste jer se ne čeka nužno odgovor sustava za djelomice upisane kriterije pretrage.



Slika 2.5.9 Primjer detaljnijeg dijagrama slijeda za slučajeve uporabe "Pretraga proizvoda" i "Pregled proizvoda"

Osim toga, u prvom primjeru rješenja dijagrama objekt 'Web sučelje' podrazumijeva u sebi prihvat i obradu te prosljeđivanje kriterija pretrage bazi podataka. Uvođenjem novog objekta 'Web poslužitelj' može se pokazati novi sloj arhitekture koji prihvaća kriterije pretrage te oblikuje i postavlja SQL upit bazi podataka te potom prihvaća rezultat upita i vraća ga u sučelje. Također, može se naglasiti pregled specifično traženog proizvoda blokom *alt* koji predstavlja konstrukt uvjetnog grananja s prethodno definiranim iterativnim pregledom detalja unutar sebe. To bi označavalo da ako specifično traženi proizvod nije nađen ne događa se daljnja interakcija korisnika sa sustavom.

Slučaj uporabe 'Kupnja proizvoda' (UC6, UC7, UC8)

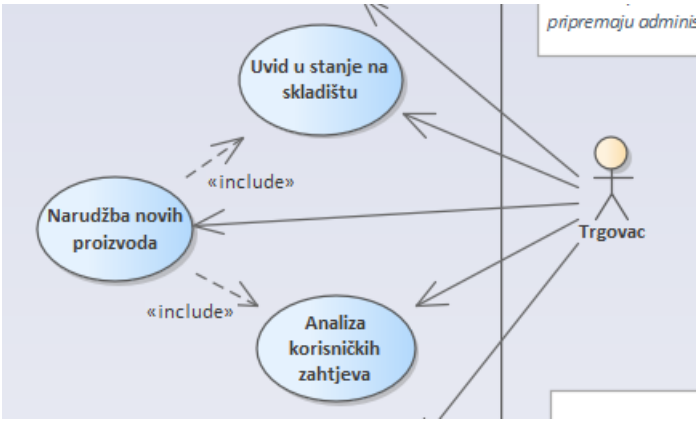
Slučaj uporabe 'Kupnja proizvoda' u sebi uključuje ili može uključiti slučajeve uporabe 'Izbor proizvoda' te 'Ostvarivanje popusta', Slika 2.4.9. jer pretpostavlja prethodni izbor proizvoda za kupnju pa zato i uključuje slučaj uporabe 'Izbor proizvoda'. Zato se slučaj uporabe 'Izbor proizvoda' može zasebno oblikovati vlastitim dijagramom slijeda i uključiti kao referenca u dijagram slijeda za slučaj uporabe 'Kupnja proizvoda' ako se smatra da bi izrada jedinstvenog dijagrama slijeda za ova dva slučaja uporabe kao ishod dala previše složen dijagram slijeda. Inače bi izbor proizvoda bio prikazan kao detaljniji uvod u prikaz kupnje proizvoda.

S druge strane, slučaj uporabe 'Ostvarivanje popusta' je opcionalan te se u dijagram slijeda slučaja uporabe 'Kupnja proizvoda' uključuje pod uvjetom.

Oblikovanje dijagrama slijeda, jednog ili više njih, za ove slučajeve uporabe studenti mogu napraviti za vježbu.

Slučaj uporabe 'Narudžba novih proizvoda' (UC12, UC13, UC14)

Slučaj uporabe 'Narudžba novih proizvoda' u sebi uključuje analizu korisničkih zahtjeva i uvid u stanje na skladištu, Slika 2.5.10.



Slika 2.5.10 Isječak dijagrama slučajeva uporabe vezan uz narudžbu novih proizvoda

Zato je moguće zasebno oblikovati dijagrame slijeda uključenih slučajeva uporabe i referencirati ih u dijagramu slijeda (bilo kojim redoslijedom ili iterativno) ili, ovisno o složenosti tih slučajeva uporabe, kombinirati ih u jedinstvenom dijagramu slijeda s osnovnim slučajem uporabe.
Oblikovanje dijagrama slijeda za ove slučajeve uporabe može se također napraviti za vježbu.

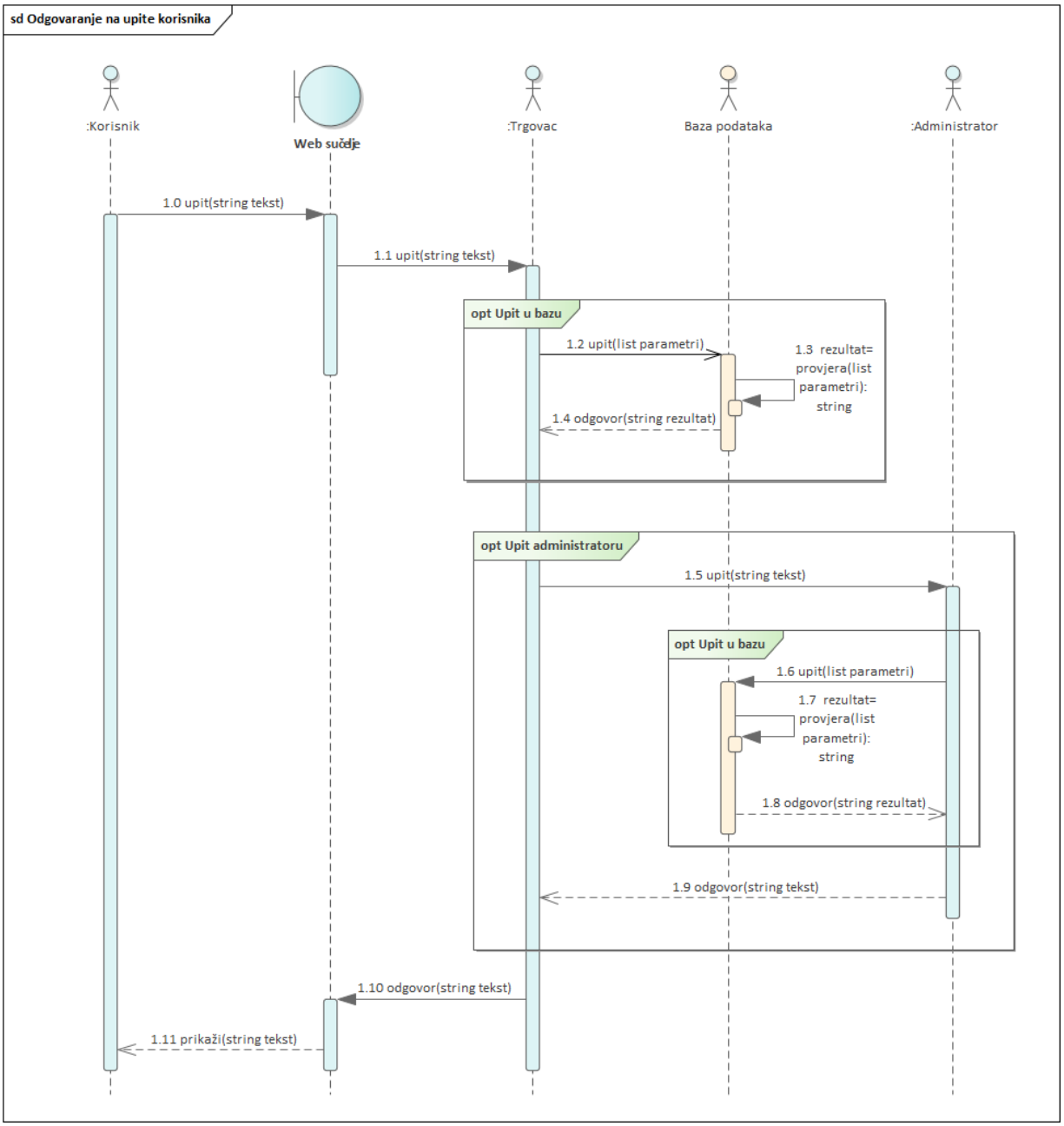
Slučaj uporabe 'Odgovaranje na upite korisnika' (UC3, UC10)

Odgovaranje na upite korisnika logički podrazumijeva da je upit uopće i postavljen, Slika 2.5.11.



Slika 2.5.11 Isječak dijagrama slučajeva uporabe vezan uz korisničku podršku korisniku

Slika 2.5.12 prikazuje moguće rješenje dijagrama slijeda za korisničku podršku odnosno odgovaranje na upite koji su postavljeni od strane korisnika.



Slika 2.5.12 Dijagram slijeda kod odgovaranja na upite korisnika

Na temelju upita korisnika trgovac odgovor provjerava u bazi podataka ili se konzultira administratorom u svezi toga što je na dijagramu prikazano upotrebom više *opt* fragmenata.

Ostali slučajevi uporabe

Slučajevi uporabe koji nisu obuhvaćeni dodatnom analizom i prikazom dijagramima slijeda u prethodnim poglavljima mogu se svaki zasebno oblikovati svojim dijagramom slijeda. Tako

slučaj uporabe 'Slanje povratne informacije' (UC9) od strane kupca sadrži samo upotrebu web sučelja za upis teksta povratne informacije i slanje u sustav. Slučajevi uporabe trgovca 'Uvid u korisničke narudžbe' (UC11), 'Definiranje popusta' (UC15), 'Uvid u statistiku prodaje' (UC16) pretpostavljaju uporabu korisničkog sučelja s ovlastima trgovca kao i slučajevi uporabe administratora s posebnim ovlastima za 'Oblikovanje izvješća' (UC17) i 'Osvježavanje baze podataka' (UC18) te se neće posebno prikazivati na ovom mjestu već se predlažu kao mogućnost za samostalnu izradu i vježbu studentima.

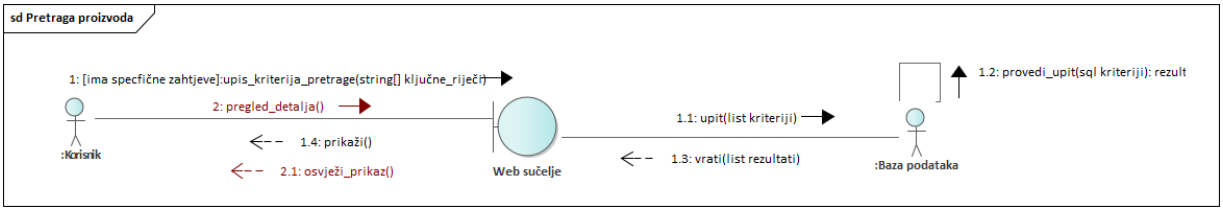
2.6. Dijagram komunikacije

Dijagrami komunikacije ili dijagrami kolaboracije (engl. *communication diagram*, *collaboration diagram*) slično kao dijagram slijeda prikazuju suradnju ili razmjenu poruka između različitih sudionika sustava s naglaskom na prikaz toga koji sudionici razmjenjuju poruke. Vremenski slijed se ne naglašava toliko izričito ili vizualno kao na dijagramu slijeda već samo oznakama pobrojavanja poruka koje se razmjenjuju. U odnosu na dijagramu slijeda ova vrsta dijagrama izgleda bitno sažetije.

Kako je bitno uočiti razliku između te dvije vrste dijagrama, u sljedećim odlomcima se za dijagrame slijeda, odnosno slučajeve uporabe za koje su dijagrami slijeda izrađeni, a koji su prikazani u prethodnom poglavlju, ukratko prikazuju odgovarajući dijagrami komunikacije.

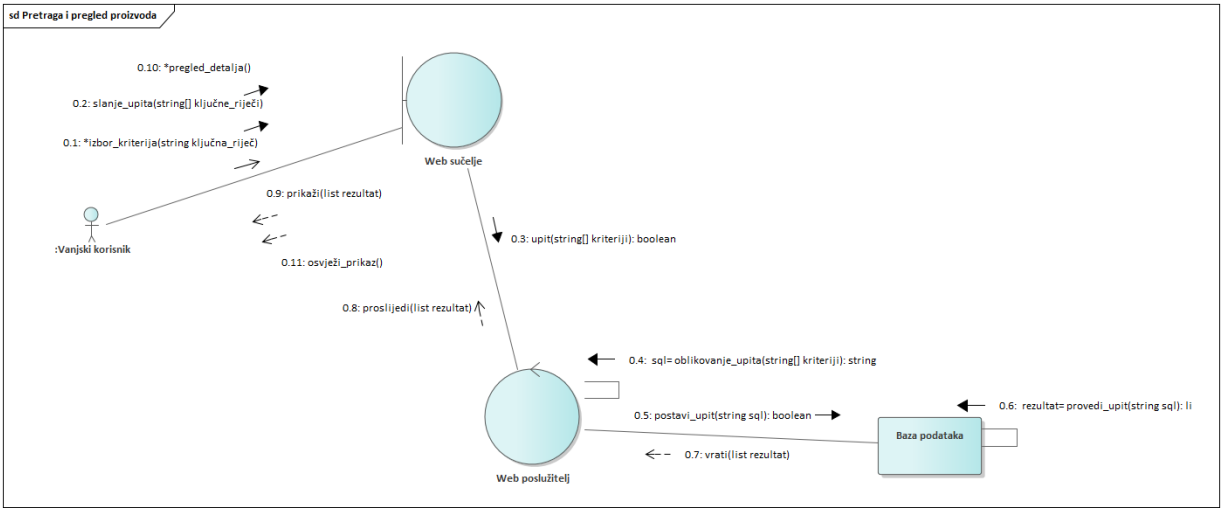
Slučaj uporabe 'Pregled proizvoda' (UC1, UC2)

Slika 2.6.1 prikazuju dijagram komunikacije za pretragu i pregled proizvoda od strane korisnika. Notacija aktora ili objekata koji su u komunikaciji te poruka koje se izmjenjuju je identična onoj uporabljenoj kod dijagrama slijeda, ali nema životnih linija aktora tako da dijagram izgleda kao sažeti oblik dijagrama slijeda, bez izričito vizualnog vremenskog uređenja u slanju poruka. Vidljivo je koji aktori i objekti su u međusobnoj komunikaciji.
Poruke su pobrojane na isti način kao i kod dijagrama slijeda, u dvije skupine, s obzirom na dio koji se odnosi na pretragu i dio koji se odnosi na pregled detalja proizvoda, Slika 2.5.8.



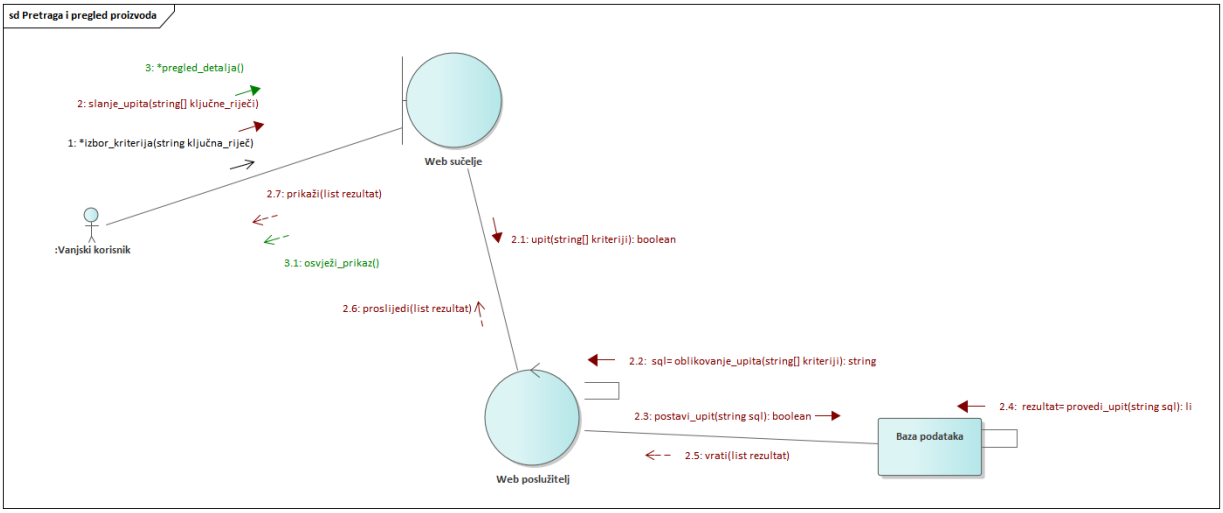
Slika 2.6.1 Primjer dijagrama komunikacije slučaja uporabe "Pretraga proizvoda"

Slika 2.6.2 prikazuje dijagram komunikacije za pretragu i pregled proizvoda s detaljnije prikazanom strukturom aktora i objekata u komunikaciji kako je to bilo priloženo i s dijagramom slijeda, Slika 2.5.9. Ovaj dijagram je primjer dobiven na temelju automatskog generiranja dijagrama komunikacije iz dijagrama slijeda (uporabom alata programske potpore) te je pobrojavanje poruka poravnato na istoj razini, označenom s 0.*x*.



Slika 2.6.2 Automatski generirani dijagram komunikacije za slučajeve uporabe 'Pretraga proizvoda' i 'Pregled proizvoda'

Slika 2.6.3 prikazuje dijagram komunikacije pretrage i pregleda proizvoda nakon dorada u pobrojavanju poruka.

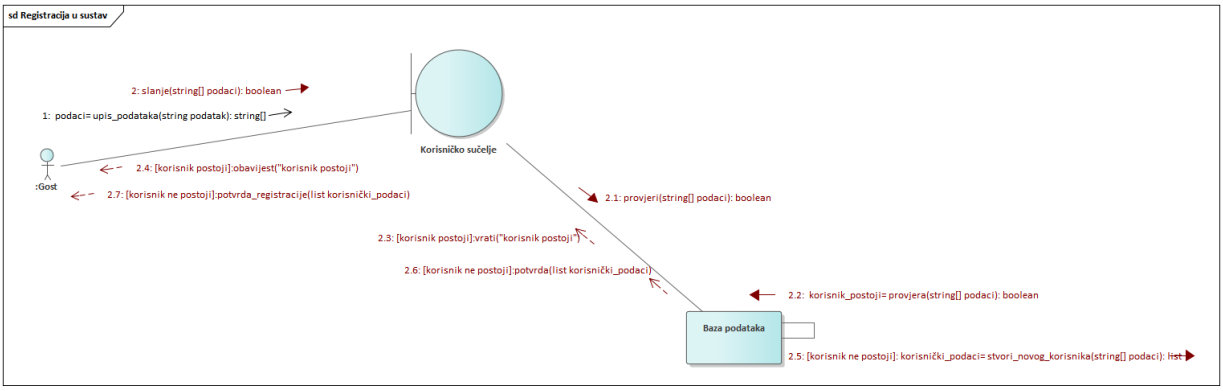


Slika 2.6.3 Dorađeni dijagram komunikacije slučajeva uporabe 'Pretraga proizvoda'" i 'Pregled proizvoda'

Konkretno, u ovom slučaju može se malo bolje vidjeti razlika u skupinama poruka po različitim bojama notacije poruke. Različite skupine poruka započinju s različitim akcijama korisnika te je na taj način vidljivo koji niz poruka između drugih sudionika odgovara pojedinoj inicijativnoj akciji korisnika. Primjerice, time je bolje vidljivo da povratna poruka 'prikaži(list rezultati)' s brojem 2.7 predstavlja odgovor na poruku 'slanje_upita(string[] ključne_riječi)' s brojem 2, a da povratna poruka 'osvježi_prikaz()' s brojem 3.1 odgovara početnoj poruci 'pregled_detalja()' s brojem 3.

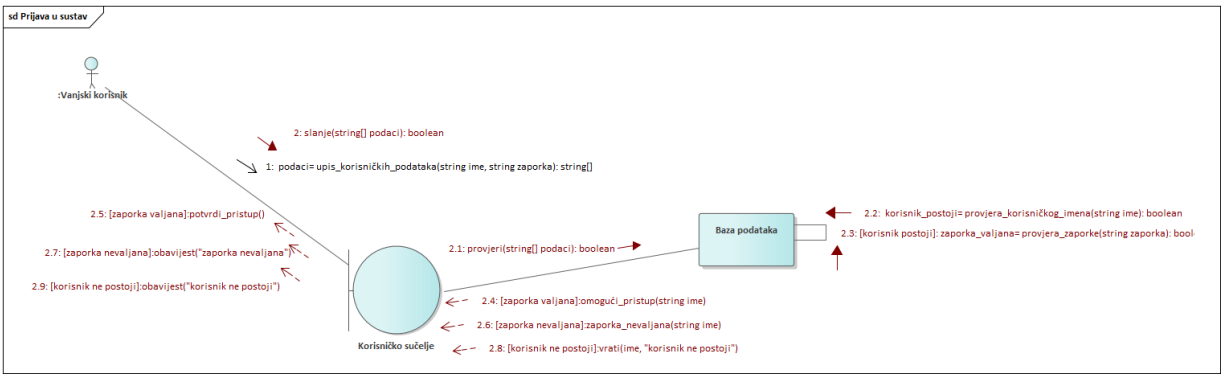
Slučaj uporabe 'Prijava u sustav' (UC4, UC5)

Slika 2.6.4 prikazuje dijagram komunikacije za registraciju korisnika. Uvjetno grananje je naznačeno notacijom navođenja uvjeta unutar uglatih zagrada.



Slika 2.6.4 Dijagram komunikacije za registraciju korisnika

Slika 2.6.5 prikazuje dijagram komunikacije prijave korisnika u sustav.

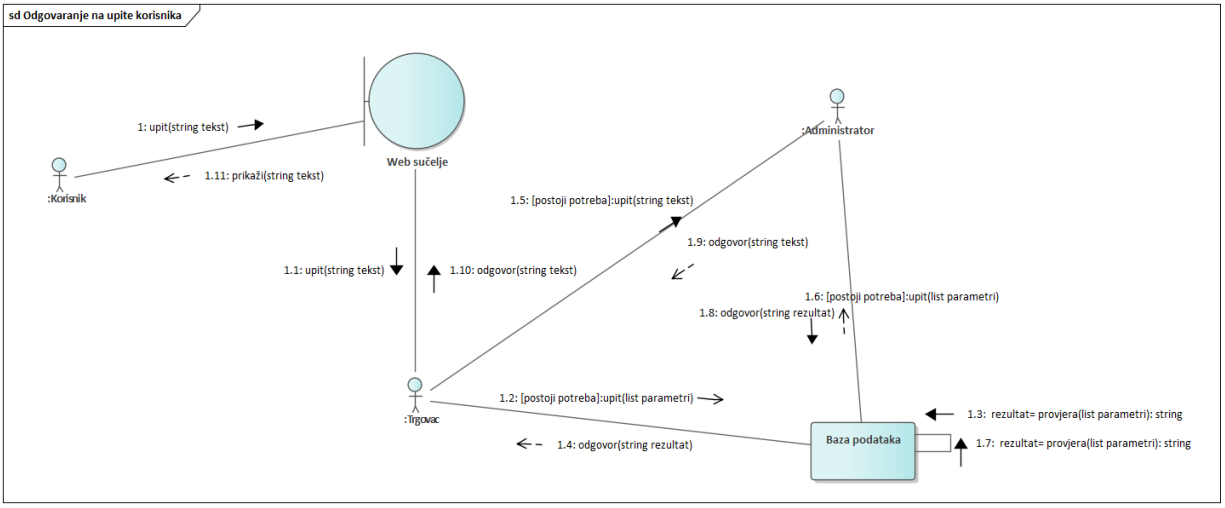


Slika 2.6.5 Dijagram komunikacije za prijavu korisnika u sustav

U slučaju nepostojanja korisnika ili korisničkog imena s kojim bi se netko htio prijaviti samo se ispisuje odgovarajuća poruka o tome.

Slučaj uporabe 'Odgovaranje na upite korisnika' (UC3, UC10)“

Slika 2.6.6 prikazuje dijagram komunikacije kod odgovaranja na upite korisnika.



Slika 2.6.6 Dijagram komunikacije kod odgovaranja na upite korisnika

Notacija opcije koja je prikazana odgovarajućim fragmentom u dijagramu slijeda ovdje je prikazana navođenjem uvjeta unutar uglatih zagrada.

2.7. Dijagram razreda

Dijagram razreda ili dijagram klasa (engl. *class diagram*) predstavlja temelj modeliranja objektno-orijentirane arhitekture sustava. Prikazuju se razredi i odnosi između njih. Mogu se modelirati na više razina apstrakcije gdje se mogu ali i ne moraju prikazati pojedine skupine atributa i metoda. Na temelju toga se razlikuju "domenski" i "implementacijski" dijagrami razreda. Prvi je na višoj razini apstrakcije i služi za razradu koncepta i razumijevanje domene, a potonji za dokumentiranje implementiranog sustava.

Općenito, model razreda predstavlja središnji koncept pregleda, razvoja i dokumentiranja razvoja programske potpore jer predstavlja vezu između funkcionalnih zahtjeva, domene problema koji se rješava i projekta programskog ostvarenja koje se oblikuje.

Svaki **razred** ili **klasa** (engl. *class*) predstavlja neki entitet koji posjeduje neke podatke i odgovornosti koje se mogu na razini apstrakcije jezika UML i prikaza predstaviti neovisno o programskom jeziku i tehnologiji koja se rabi za njegovo ostvarenje. U širem smislu razvoja programske potpore taj entitet predstavlja shemu ili plan oblikovanja na temelju kojeg će se razvijati stvarni programski objekti i komponente. U užem, programerskom smislu, njime se definira nova vrsta ili tip podatka, uz one postojeće generičke tipove podataka koji karakteriziraju pojedinu tehnologiju izrade.

Dijagram razreda, za razliku od dijagrama slučajeva uporabe, dijagrama aktivnosti, dijagrama slijeda i dijagrama komunikacije, spada u skupinu strukturnih statičkih dijagrama jer postavlja manje ili više čvrstu strukturu kao okosnicu arhitekture programske potpore.

Model domene sustava koji se njime oblikuje više je konceptualne naravi i služi prepoznavanju i definiciji bitnih elemenata ili entiteta sustava, organizacije i odgovarajućeg područja interesa ili „domene“ problema. U prvom dijelu ovog poglavlju se podrazumijeva takav, „konceptualni“, pristup oblikovanju dijagrama razreda za razliku od dijagrama razreda opisa konačnog ostvarenja o kojima će biti više riječi u drugom dijelu poglavlja gdje razredi predstavljaju definicije objekata konkretne tehnologije implementacije.

Kada se radi o dijagramu razreda na jednoj nižoj razini apstrakcije koji u sebi uključuje i neke arhitekturne prikaze i ovisnosti moglo bi se reći onda da bi to bio „implementacijski dijagram razreda“ ili „dijagram razreda ostvarenja“ u odnosu na ovaj koji je modeliran kao „domenski dijagram razreda“ ili „konceptualni dijagram razreda“ s naglaskom na sloju podatkovnog modela i do razine apstrakcije koja bi trebala biti lakše razumljiva pri pogledu na dijagram. Takav dijagram svakako nije sveobuhvatan u smislu da ide samo do određene razine detalja u prikazu. Taj konceptualni dijagram razreda se može gledati kao početna točka prema razvoju konačne arhitekture koja će biti predstavljena dijagramima razreda i komponenti ostvarenja.

Razina domene (konceptualna)

U početnim fazama razvoja programske potpore dijagram razreda se koristi za modeliranje osnovnih entiteta iz domene koji su najčešće predstavljeni razredima podatkovnog sloja sustava (ili „modela“), npr. u arhitekturnom obrascu **model-pogled-nadglednik** ili **MVC** (engl. *Model-View-Controller*. Taj sloj se uvelike poklapa sa strukturom baze podataka i sadržajem odgovarajućeg dijagrama **E-R** (engl. *Entity-Relationship*), ali, uz podatkovne članove (atribute),

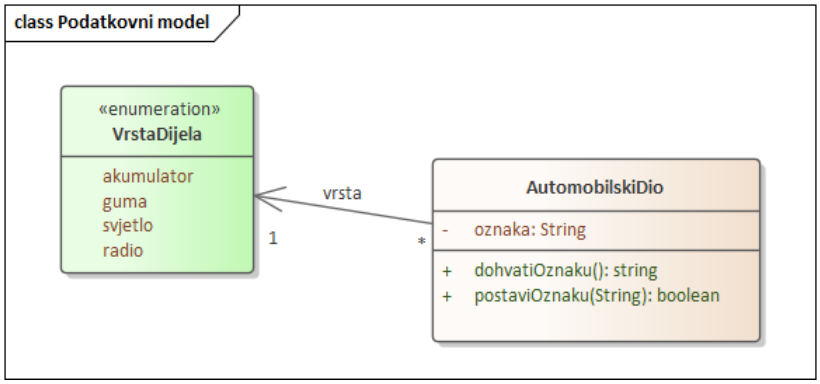
uvodi apstrakciju razreda s odgovarajućim operacijama ili metodama kao odgovornostima koji neki novi entitet nosi.

Dijagram razreda podatkovnog modela

Podatkovni model domene sustava prodaje auto-dijelova sadrži neke temeljne entitete koji se mogu iščitati iz opisa projektnog zadatka:

- automobilski dio,
- model automobila,
- marka vozila,
- proizvođač vozila,
- proizvođač dijelova,
- dobavljač dijelova.

Središnji entitet oko kojeg se sustav prodaje oblikuje je automobilski dio kao primarni predmet prodaje. Stoga se i definira kao zaseban entitet ili razred 'AutomobilskiDio' čija će se evidencija i prodaja pratiti. Svaki automobilski dio se identificira po jedinstvenoj oznaci proizvođača pa se kao prvi atribut razreda može definirati 'oznaka' vrste. Osim toga, svaki automobilski dio se svrstava u neku od mogućih ili predefiniranih vrsta dijelova. Ovdje će se za primjer navesti samo nekoliko njih. Taj skup različitih vrsta auto-dijelova je u svakom slučaju ograničen pa će se modelirati kao entitet pobrojavanja ili enumeracije (engl. *enumeration*) naziva 'VrstaDijela'. Odnos između entiteta 'AutomobilskiDio' i 'VrstaDijela' se može prikazati kao **usmjerena veza pridruživanja**, Slika 2.7.1.



Slika 2.7.1 Prikaz usmjerene veze pridruživanja (strelica) u dijagramu razreda

Atribut 'oznaka' definiran je kao općeniti tekstni podatak, ili *string*, privatnog dosega vidljivosti te su definirane metode da njegov dohvat (engl. *get*) i postavljanje (engl. *set*). Za atribute i metode, osim naziva i tipova podataka, navode se i razine pristupa ili vidljivosti. Način označavanja i značenja vidljivosti atributa i metoda prikazan je u Tablica 2.7.1.

Tablica 2.7.1 Pregled razina vidljivosti atributa i metoda razreda

Razina vidljivosti	*Oznaka*	*Opis*
privatno (engl. *private*))	-	dostupno samo unutar istog razreda
zaštićeno (engl. *protected*))	#	dostupno unutar istog razreda i izvedenih razreda
javno (engl. *public*)	+	dostupno svim razredima i paketima
paket (engl. *package*)	~	dostupno razredima unutar istog paketa

Javna i privatna razina su dvije krajnje mogućnosti razine zaštite sadržaja razreda, dok se zaštićena razina veže uz odnos **nasljeđivanja** koji može postojati između razreda gdje je novi razred izveden iz postojećeg i predstavlja njegovu **specijalizaciju** ili podtip. Razred koji je naslijeđen u tom smislu predstavlja **poopćenje** ili **generalizaciju** ili nadtip svih svojih izvedenih razreda.

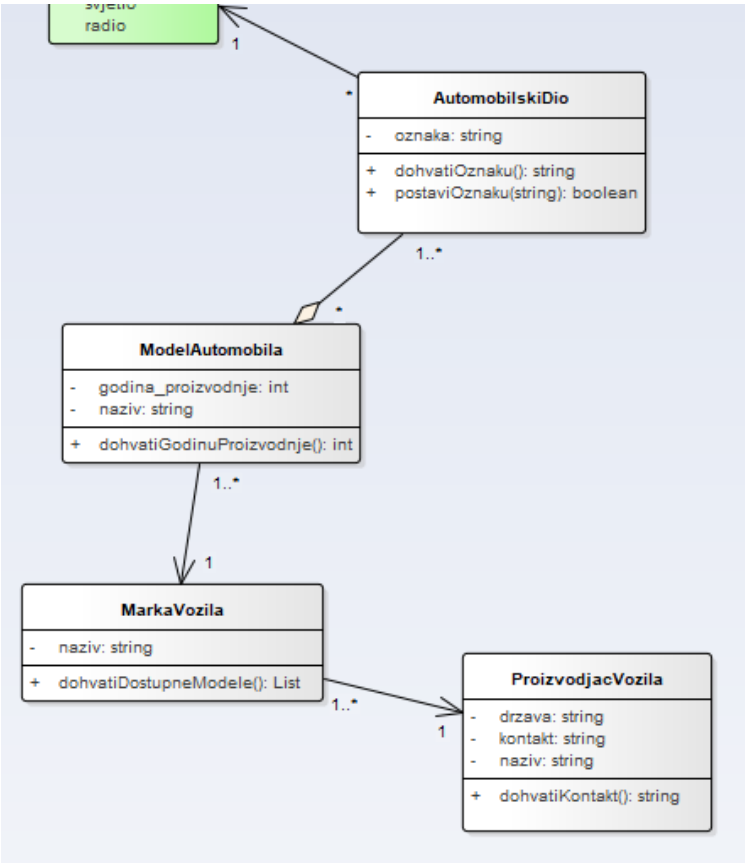
Za razliku od entiteta automobilskog dijela koji je predstavljen razredom 'AutomobilskiDio', entitet modela automobila za koji je pojedini dio namijenjen je nešto složeniji entitet koji će biti predstavljen razredom 'ModelAutomobila'. Osim naziva modela, potrebno je znati i marku automobila te njezinog proizvođača. Nadalje, godina proizvodnje automobila je također važan podatak, jer istovjetni dio može biti proizveden u različitim serijama za novije i starije primjerke istog modela automobila. Temeljni skup podataka svojstven modelu automobila koji će biti naveden u skupu atributa dotičnog razreda uključuje njegov naziv i godinu proizvodnje, dok će marka i proizvođač automobila biti posebno oblikovani zasebnim razredima jer jednoj marki automobila može pripadati više modela automobila, a jedan proizvođač proizvodi više marki i modela automobila.

Osim odnosa poopćenja ili specijalizacije kojim se opisuju podtipovi nekog tipa koji je definiran osnovnim razredom, druga temeljna vrsta odnosa je opisana pojmom **pridruživanja** između različitih razreda kojim se označavaju razne vrste pripadnosti između razreda. Vezom između entiteta 'AutomobilskiDio' i 'VrstaDijela' prikazan je jednostavan odnos pridruživanja koji označava posjedovanje reference unutar instance jednog na instance nekog drugog razreda.

U primjeru koji se ovdje elaborira pojedini model automobila se sastoji od skupa njemu pripadajućih dijelova pa se tu može definirati veza **agregracije** koja označava odnos koji znači posjedovanje u smislu „cjelina-dio“, tj. da se instanca jednog razreda sastoji od skupa instanci nekog drugog razreda. U slučaju ovdje konkretno definiranog takvog odnosa ukidanjem pojedinog modela ne moraju nužno nestati i svi njegovi dijelovi jer se pojedini dio može koristiti u više modela automobila, inače bi se upotrijebila veza **kompozicije** kao jakog oblika agregacije što podrazumijeva da dijelovi koji čine cjelinu ne mogu postojati neovisno o cjelini koju čine.

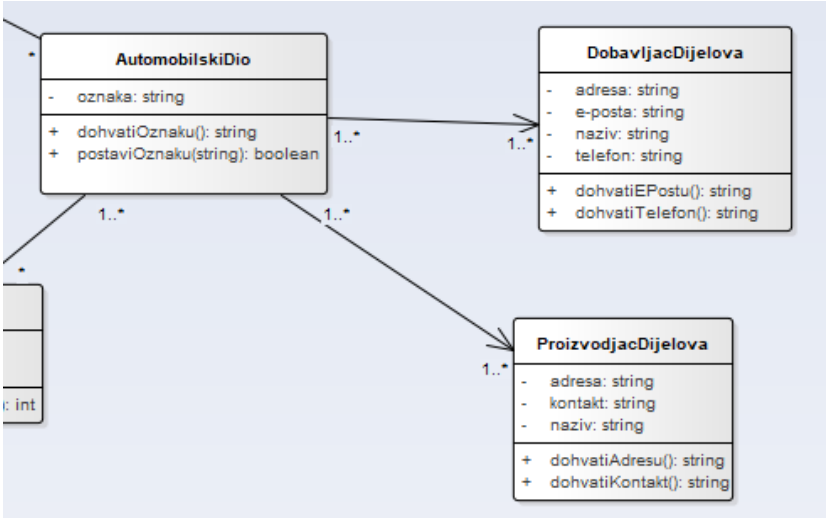
Situacija u ovom konkretnom slučaju bi se mogla konačno oblikovati s tri dodatna razreda 'ModelAutomobila', 'MarkaVozila' i 'ProizvodjacVozila' koji su povezani vezama agregacije i pridruživanja, Slika 2.7.2. Na vezi agregacije se vidi da se jedan automobilski dio može ali i ne mora općenito koristiti u jednom ili više modela automobila, (naznačeno sa '*'), a jedan model automobila se sastoji od barem jednog dijela (naznačeno s '1 .. *') čime se definiraju **višestrukosti** na vezama pridruživanja između razreda. Usto, jedan model automobila pripada samo jednoj marki automobila, a jedna marka vozila se proizvodi od strane samo jednog proizvođača vozila te se ta dva odnosa predstavljaju vezama pridruživanja.

Kako je prethodno i najavljeno, između razreda 'ModelAutomobila' i 'AutomobilskiDio' definirana je veza agregacije jer skup dijelova čini cjelinu modela automobila a usmjerenje te veze je od razreda koji predstavlja cjelinu prema razredu koji predstavlja entitete dijelova te cjeline. Također, od razreda 'ModelAutomobila' prema razredu 'MarkaVozila' definirana je usmjerena veza pridruživanja čime se naznačava da model automobila posjeduje referencu na neku marku vozila čiji je predstavnik gdje jedan model pripada samo jednoj marki vozila. Iz istog razloga postoji usmjerena veza pridruživanja između razreda 'MarkaVozila' i razreda 'ProizvodjacVozila'.



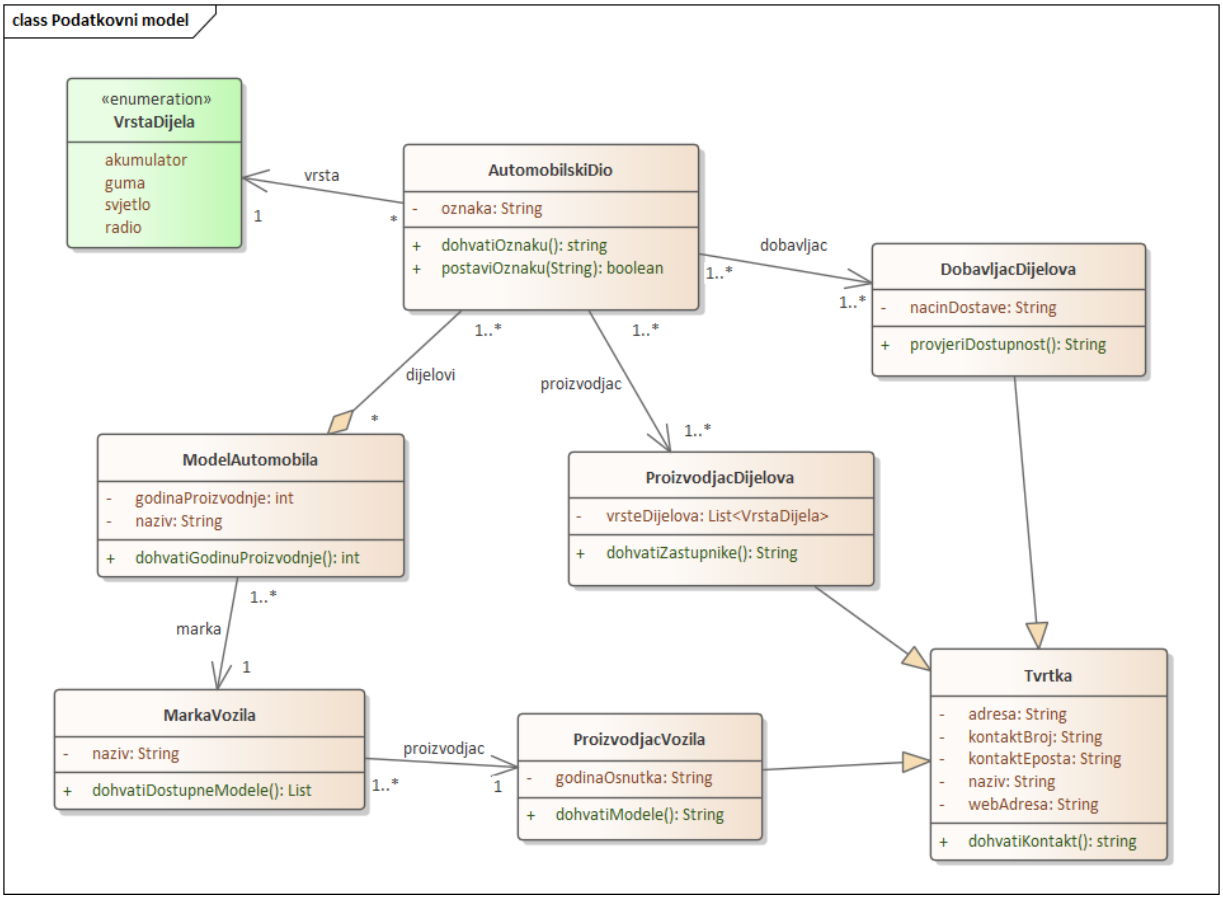
Slika 2.7.2 Dio dijagrama razreda vezan uz model automobila

Sljedeća važna informacija za automobilski dio je njegov proizvođač pa se definira novi razred 'ProizvodjacDijelova' za koji je važno imati kontakt adresu i podatke. Osim proizvođača, važno je imati i podatke o dobavljačima pa se definira i razred 'DobavljacDijelova'. Ta dva razreda se povezuju s razredom 'AutomobilskiDio' usmjerenom vezom pridruživanja, Slika 2.7.3.



Slika 2.7.3 Dio dijagrama razreda s proizvođačima i dobavljačima dijelova.

Za razrede proizvođača vozila, proizvođača dijelova i dobavljača dijelova postoji sličan zajednički skup informacija i popratnih metoda. Štoviše, radi se o nekim tvrtkama o kojima je važno u ovom slučaju znati uglavnom kontakt podatke pa se taj dio sustava može oblikovati razredom poopćenja ta tri razreda. Definira se razred 'Tvrtka' koji je nadrazred za ta tri razreda i sadrži atribute i metode koje su važne kod pristupa objektima tih vrsta podrazreda, Slika 2.7.4. U podrazredima su ostali samo atributi i metode koji mogu biti specifični za pojedini razred. Ti podrazredi imaju atribute koji označavaju neke dodatne informacije: razred 'ProizvodjacVozila': godinu osnutka, razred 'ProizvodjacDijelova' vrste dijelova koje proizvodi, 'DobavljacDijelova' podatak o načinu dostave dijelova. Radi se o odnosu poopćenja tri specifična entiteta razredom 'Tvrtka' odnosno njegovim specijalizacijama u obliku tri navedena razreda sa svojim specifičnostima.



Slika 2.7.4 Dijagram razreda vezan za podatkovni model razreda AutomobilskiDio.

Mogućnosti drugačijih rješenja zadatka

U pogledu mogućih drugih rješenja za pojedine vrste odnosa moglo bi se reći sljedeće:

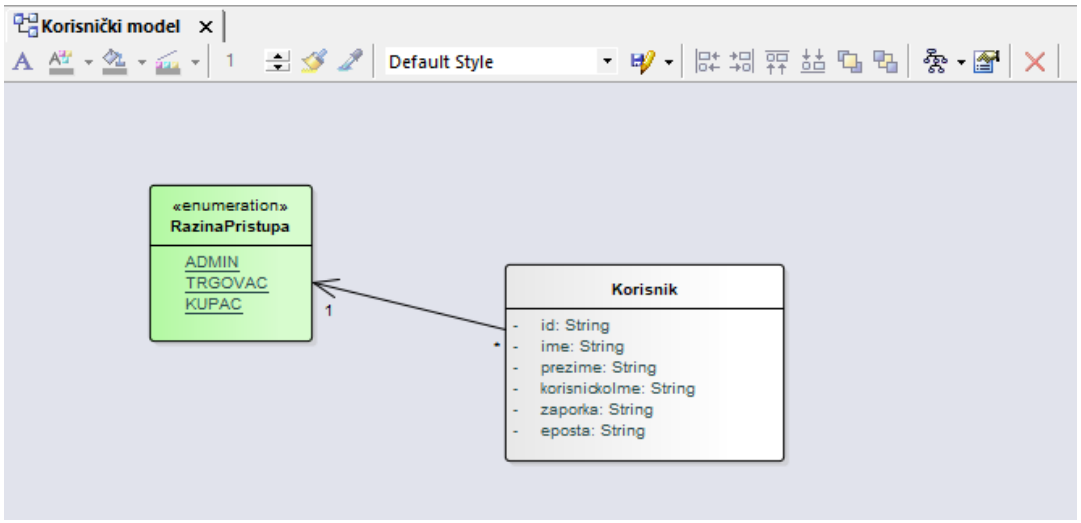
- Između razreda 'ProizvodjacDijelova' i pobrojavanja 'VrstaDijela' nije umetnuta izričita veza pridruživanja zbog preglednosti dijagrama, ali je implicirana postojanjem liste vrsta dijelova odnosno pojedinaca razreda 'VrstaDijela' u razredu 'ProizvodjacDijelova'.
- Veza između razreda 'ProizvodjacVozila' i razreda 'MarkaVozila' nije oblikovana kao odnos agregacije ili kompozicije iako je normalno za pretpostaviti da svaki proizvođač dijelova posjeduje informaciju o svim svojim markama vozila. Marke vozila same po sebi ne čine cjelinu proizvođača vozila kako to automobilski dijelovi čine za određeni model automobila, iako je moguće za pretpostaviti da proizvođač vozila može proizvesti više različitih marki. Dijagram razreda je ovdje uvelike konceptualne naravi i prikazuje strukturu razreda do određene razine i s nastavkom programskog ostvarenja je uvelike podložan promjenama. Nije uvijek moguće ni potrebno na istom dijagramu konceptualne naravi uvijek prikazati i najsitnije detalje prikazati, pogotovo u ranijem razdoblju razvoja. U kontekstu opisa koji se zasniva na razini automobilskih dijelova usredotočenost ne mora biti na obilježjima proizvođača vozila i njihovih marki vozila.
- Odnos razreda 'ModelAutomobila' i 'MarkaVozila' modeliran je usmjerenom vezom pridruživanja iz sličnih razloga kao i odnos između razreda 'MarkaVozila' i 'ProizvodjacVozila' iako bi imalo opravdanja da se i to definira kao odnos agregacije

između marke vozila i modela automobila koji pripadaju pojedinoj marki. Na ovom dijagramu koji je konceptualne naravi radi se o karakteristici opisa odnosa kao načelne pretpostavke što se tijekom faze implementacije početne pretpostavke konačno može potvrditi ili opovrgnuti.

Gledano sa strane programskog ostvarenja ono može biti takvo da se između navedenih razreda ostvari veza agregacije ili kompozicije ovisno o tome hoće li se pojedinci agregiranih razreda brisati iz podatkovne memorije programa po brisanju pojedinaca agregirajućeg razreda ili ne. U tom slučaju već govorimo o dijagramu razreda na jednoj nižoj razini apstrakcije koji u sebi uključuje i neke arhitekturne prikaze i ovisnosti. Moglo bi se reći onda da bi to bio „implementacijski dijagram razreda“ ili „dijagram razreda ostvarenja“ u odnosu na ovaj koji je modeliran kao „domenski dijagram razreda“ ili „konceptualni dijagram razreda“ s naglaskom na sloju podatkovnog modela i do razine apstrakcije koja bi trebala biti lakše razumljiva pri pogledu na dijagram. Takav dijagram svakako nije sveobuhvatan u smislu da ide samo do određene razine detalja u prikazu. Taj konceptualni dijagram razreda se može gledati kao početna točka prema razvoju konačne arhitekture koja će biti predstavljena dijagramima razreda i komponenti ostvarenja. Npr., pretpostavka je da svaki razred u konačnom ostvarenju ima u popisu metoda i one za dohvat (engl. *get*) i postavljanje (engl. *set*) vrijednosti svojih atributa. U konceptualnom dijagramu to nije nužno prikazati jer se podrazumijeva, ali se i može dodatno naglasiti u prikazu komentara.
Zaključno bi se prema tome moglo reći da se kod razrade dijagrama na konceptualnoj razini ne treba uvijek inzistirati na semantički prejakim odnosima kao što su agregacija i kompozicija što ionako može biti predmet daljnjih razmatranja prilikom implementacije modela.

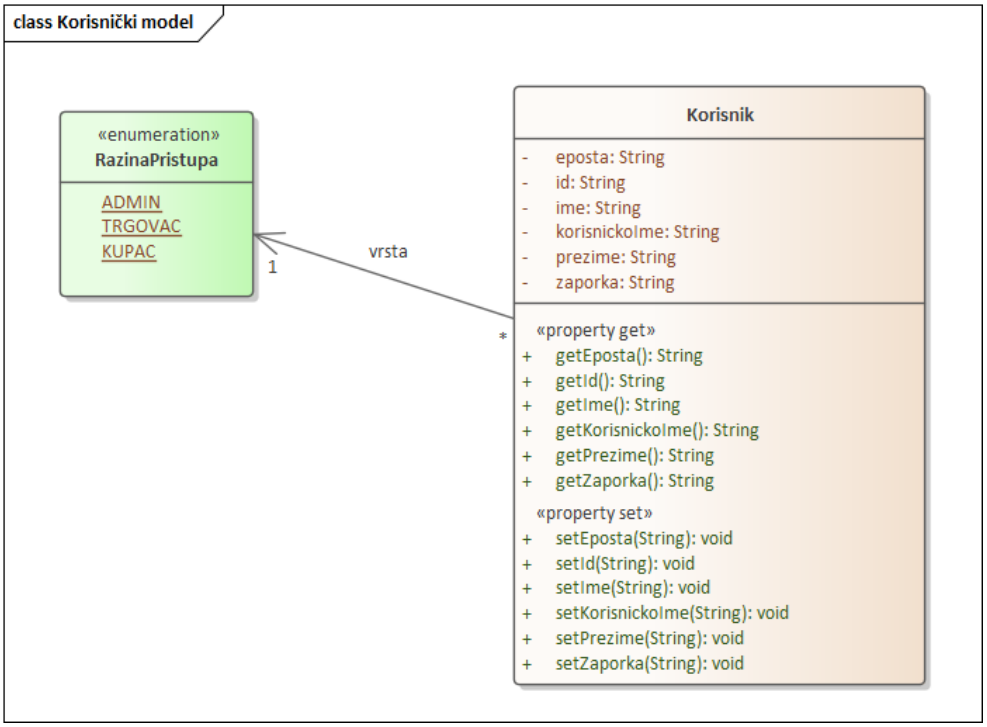
Izrada izvornog koda

Na temelju modela koji su izrađeni kreće se u implementaciju odnosno pisanje izvornog koda, ali, ovisno o alatu koji se koristi za modeliranje, može postojati i mogućnost generiranja izvornog koda ili automatizacije kao jedne od dviju temeljnih ideja modelno usmjerenog razvoja. Primjer na kojem će se demonstrirati automatsko generiranje, ili izrada izvornog programskog koda, je korisnički model s razredom 'Korisnik' koji se referencira na entitet pobrojavanja 'RazinaPristupa', Slika 2.7.5. Radi se o dosta jednostavnom primjeru koji je odabran radi razumljivosti.



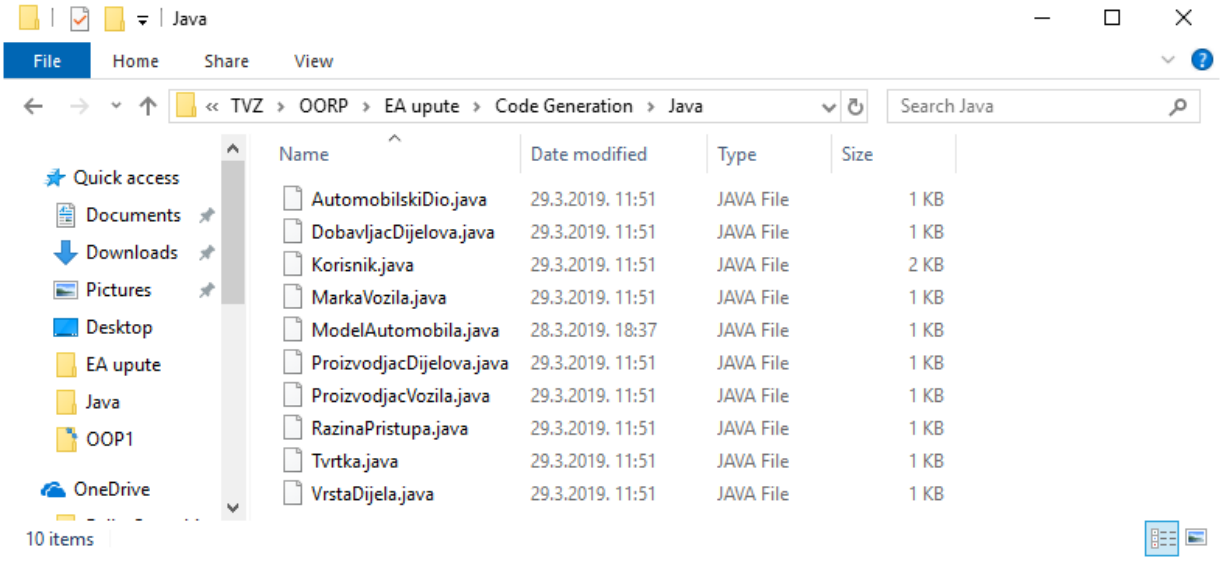
Slika 2.7.5 Početni dijagram razreda korisničkog modela

Veza između njih označava da jedan korisnik može imati samo jednu razinu pristupa, a da jedna razina pristupa može vrijediti za nebrojeno korisnika. Razred 'Korisnik' je definiran sa svojim atributima kojima je **doseg vidljivosti** ili **razina pristupa** privatna (engl. *private*) što je na dijagramu naznačenu simbolom '-'. Općenito se pretpostavlja da razred 'Korisnik' neće imati nekih posebnih odgovornosti osim pohrane podataka o korisniku koji su definirani njegovim atributima te će trebati samo metode za postavljanje (*set*) i dohvat (*get*) vrijednosti tih atributa. Definiranjem privatnog dosega vidljivosti atributa i posebnih *Property* metoda (pogledati poglavlje 4.2) za postavljanje i dohvat implementira se enkapsulacija podataka, odnosno sprječava se izravno manipuliranje podacima od strane vanjskog svijeta i stvara mogućnost postavljanja pravila i ograničenja u načinu obrade podataka. Ako se to načelo primijeni i na ostale atribute dobit će se konačni oblik dijagrama razreda korisničkog modela koji je pripravljen za generiranje izvornog koda iz njega, Slika 2.7.6.



Slika 2.7.6 Konačni oblik dijagrama razreda korisničkog modela

U dodatku pod nazivom „Generirani programski kodovi“ (poglavlje 7.1) pod naslovom „Dijagram razreda korisničkog modela“ mogu se vidjeti automatski generirani programski kodovi za pobrojavanje 'RazinaPristupa' i razred 'Korisnik' pohranjeni u odgovarajuće datoteke izvornog koda. To se može ponoviti i za preostale domenske dijagrame razreda, Slika 2.7.7.



Slika 2.7.7 Pregled stvorenih datoteka izvornog koda

Ipak, treba reći da to sve predstavlja tek polaznu točku koja se daljnjim uključivanjem u cjelokupnu strukturu koda i dodatnim detaljiziranjima i prilagodbama dorađuje prema konačnoj implementaciji.

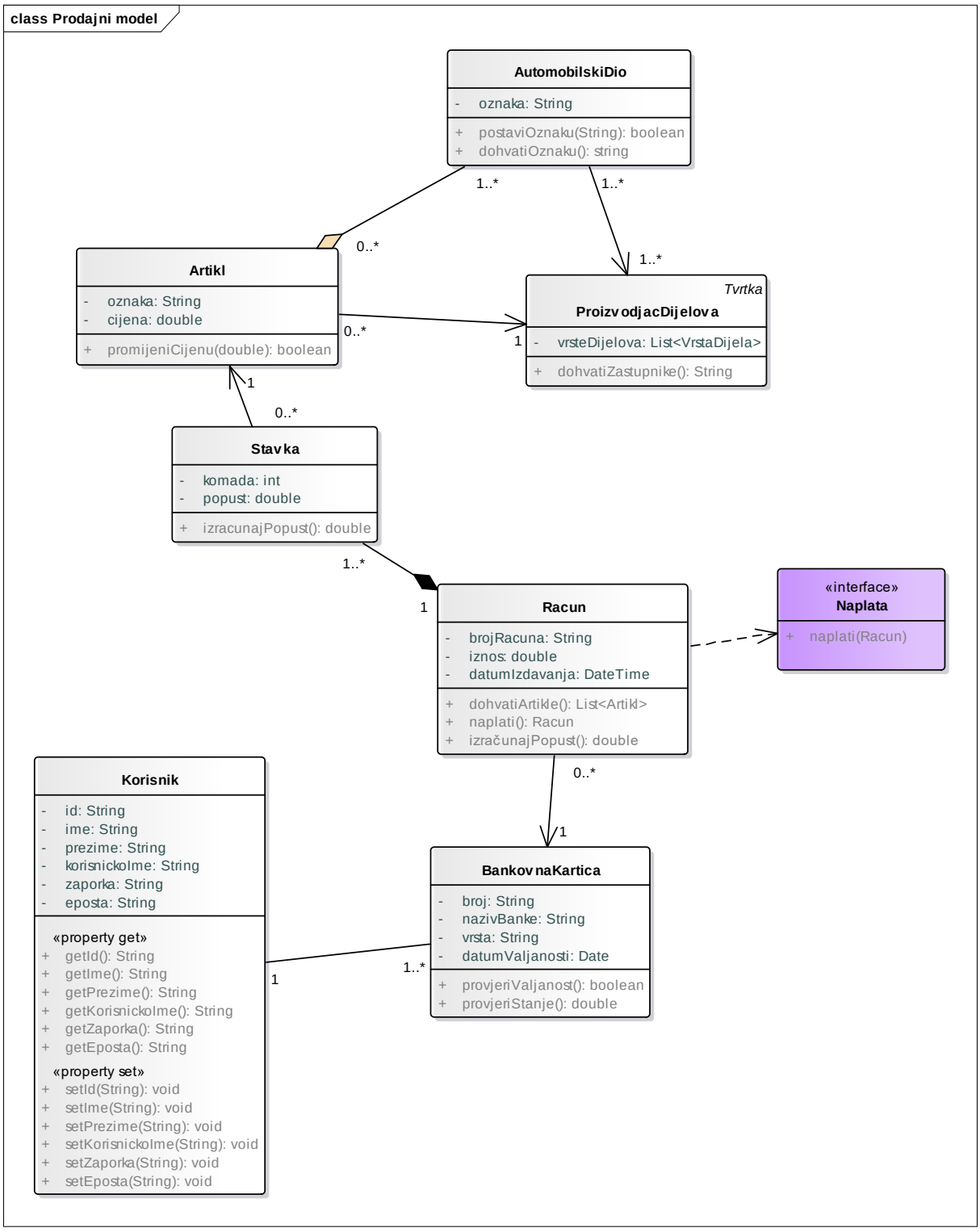
Dijagram razreda prodajnog modela

Nakon definiranja i prikaza korisničkog i podatkovnog modela koji mogu u konačnici predstavljati zasebne podsustave ciljnog programskog proizvoda na isti način se može izlučiti dio sustava koji se odnosi na samu provedbu prodaje auto-dijelova. Dijagram razreda jednog takvog paketa 'Prodajni model' prikazuje Slika 2.7.8. Na već prethodno izlučeni entitet 'AutomobilskiDio' nadovezuju se nekoliko dodatno definiranih razreda.

Prije svega, definira se razred 'Artikl' kao entitet koji definira prodajni artikl. Iako se prodajni artikl u velikoj mjeri može poistovjetiti sa samim automobilskim dijelom jer se u predviđenom sustavu prodaju samo automobilski dijelovi i oprema, razlika je u tome da prodajni artikl može u sebi sadržavati više istih ili različitih automobilskih dijelova. Npr. automobilska guma se može prodavati zasebno po komadu, ali i u kompletu od četiri automobilske gume kao zaseban prodajni artikl. Isto tako, kao zaseban prodajni artikl se može prodavati i komplet raznovrsnih dijelova ili opreme. Zato je između razreda 'AutomobilskiDio' i 'Artikl' postavljena veza agregacije koja govori da jedan artikl može u sebi sadržavati više auto-dijelova, a da neki dio može ali i ne mora bio dio nekog artikla, ne mora biti automatski ponuđen za prodaju. Ako se neki artikl makne iz prodaje, sami dijelovi se mogu maknuti iz prodaje, prodavati zasebno ili uklopiti u neki drugi prodajni artikl. Sâm prodajni artikl ima referencu na svojeg proizvođača pa zato i postoji usmjerena veza pridruživanja na razred 'ProizvodjacDijelova'. Svaki prodajni artikl ima svoju cijenu što je zadano odgovarajućim atributom 'cijena'.

Nadalje, postoji razred 'Stavka' koji se nadovezuje na razred 'Artikl' i predstavlja stavku tog artikla na računu koji se izdaje prilikom kupnje. Također se radi o usmjerenoj vezi pridruživanja gdje se jedna stavka računa odnosi samo na jedan prodajni artikl te tu treba primijetiti jedan od detalja definicije razreda 'Stavka'. Postoji atribut 'komada' koji govori da se u okviru jedne stavke računa može navesti količina ili broj artikala koji uključuje. Ako se kupuju četiri istovjetne automobilske gume zasebno to je na računu moguće navesti, primjerice, u obliku četiri stavke artikla izabrane vrste automobilske gume ili u obliku jedne stavke s navodom, tj. postavljenim atributom 'komada' na vrijednost '4'. Ovdje se postavlja opravdano pitanje ne bi li između razreda 'Stavka' i razreda 'Artikl' trebala biti definirana veza agregacije ili oznaka višestrukosti '1..*' (umjesto '1') na strani razreda 'Artikl'. I to je moguće u nekom rješenju, ali ovdje se želi reći da jedna stavka predstavlja izričito samo jednu vrstu prodajnog artikla, tj. ne može pod jednom stavkom računa biti navedeno više različitih prodajnih artikala (zato je navedena višestrukost '1') te da jedna stavka u sebi ne sadrži već pokazuje na određeni artikl (zato je usmjerena veza pridruživanja a ne agregacija). Također je dopušteno postaviti određeni prodajni popust za pojedinu stavku pa zato postoji i odgovarajući atribut 'popust'.

Razred 'Račun' u sebi strogo sadrži više (ali barem jednu) prodajnih stavki i zato prema razredu 'Stavka' postoji veza kompozicije gdje jedna stavka može postojati samo na jednom računu (ne može se doslovno isti primjerak artikla „dvaput prodavati“). Svaki račun ima svoj jedinstveni broj, iznos i vrijeme izdavanja.



Slika 2.7.8 Dijagram razreda podsustava prodaje auto-dijelova

Svaki račun daje se na naplatu posebnom podsustavu naplate koji je na dijagramu predstavljen elementom **sučelja** (engl. *interface*) kao vanjskom uslugom koja obavlja naplatu računa.
Pri naplati računa poziva se metoda sučelja 'naplati' s parametrom koji identificira pojedini račun. S obzirom na to da je za naplatu potreban iznos računa prodaje, ali i broj bankovne instance, račun s kojeg se iznos naplaćuje, tj. razred 'Račun' mora i o tome imati takvu

informaciju koja je uobličena usmjerenom vezom pridruživanja na razred 'BankovnaKartica' koji u sebi sadrži potrebne informacije za plaćanje. Svaka bankovna kartica pripada jednom korisniku, a jedan korisnik može imati više bankovnih kartica te se tu definira veza pridruživanja između razreda 'BankovnaKartica' i 'Korisnik'. Ta veza je dvosmjerna, jer se pretpostavlja da u oba entiteta postoji referenca, odnosno informacija o onom drugom, tj. na bankovnoj kartici piše kome pripada, a svaki korisnik u svakom slučaju mora imati podatke o svojim bankovnim karticama.

Dijagram razreda modela nabave

Sljedeći podsustav koji bi se mogao modelirati zasebnim dijagramom razreda je sustav nabave koji provodi trgovac čiji slučajevi uporabe su već posebno bili naznačeni uz osnovni slučaj uporabe 'Narudžba novih proizvoda', Slika 2.5.10.

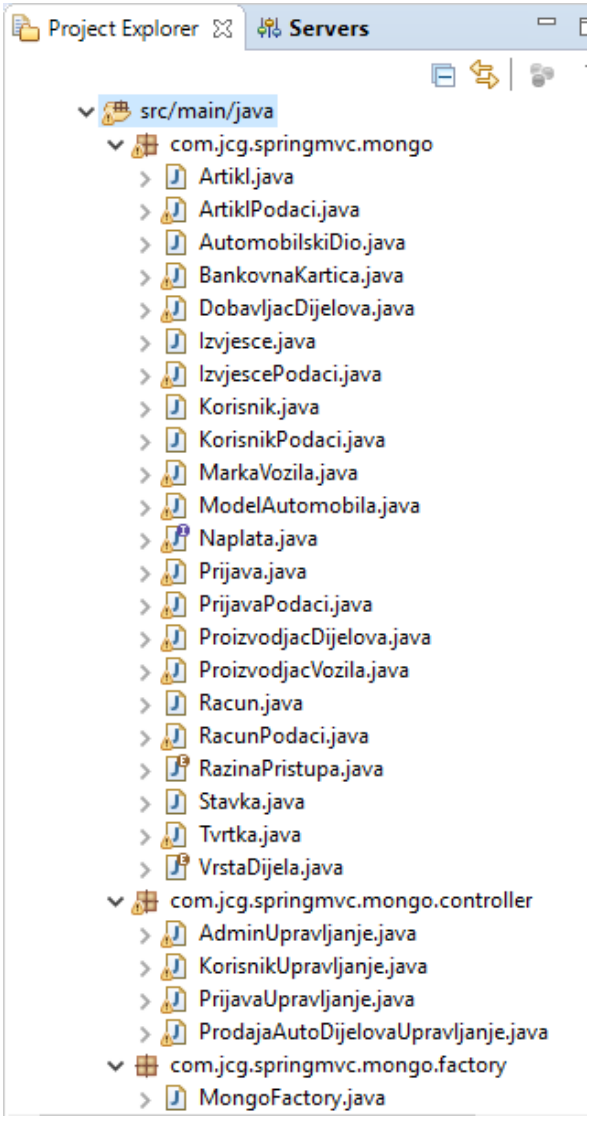
Uključen je uvid u stanje na skladištu i analiza korisničkih zahtjeva koji bi se mogli modelirati u obliku posebnih vrsta izvješća kojima upravlja administrator a sama narudžba bi se mogla ostvariti preko poziva vanjske usluge koja bi bila predstavljena svojim sučeljem.

To je jedan od mogućih pristupa oblikovanju dijagramu razreda nabave koji se može napraviti za vježbu.

Razina ostvarenja (implementacijska)

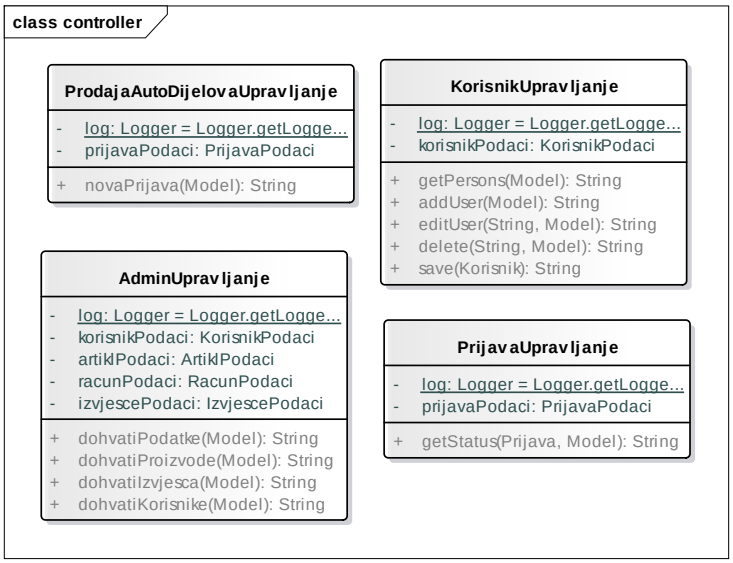
Dijagram razreda na razini ostvarenja će biti izrađen na temelju primjera programskog koda koji je dio jedinstvenog projekta u nekom trenutku razvoja programske potpore. U ovom primjeru pretpostavka je da je programski kôd razvijen u programskom jeziku Java po predlošku MVC za web aplikaciju rađenu u radnom okviru Spring MVC uz pristup bazi MongoDB. Ostvaren je dio potrebnih funkcionalnosti, a među njima i funkcionalnosti koje se odnose na ovlasti administratora. Kao podsjetnik, administrator se bavi održavanjem baze podataka proizvoda, ima uvid u statistiku prodaje, generiranje izvješća, a također ima uvid u registrirane korisnike sustava.

Slika 2.7.9 prikazuje datoteke izvornog koda koje se nalazi u tri odvojena paketa: 'com.jcg.springmvc.mongo' koji uključuje usluge pristupa podacima (dio Model iz predloška MVC), 'com.jcg.springmvc.mongo.controller' koji predstavlja upravljački dio aplikacije (dio Controller iz predloška MVC) i 'com.jcg.spring.mvc.mongo.factory' koji predstavlja vezu prema bazi podataka.

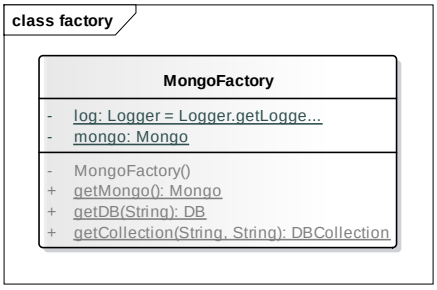


Slika 2.7.9 Prikaz strukture datoteka izvornog koda

Dijagrami razreda se mogu prije svega napraviti zasebno po paketima ili se prema potrebi mogu kombinirati različiti dijelovi sustava za prikaz unutar pojedinih dijagrama. Ovisno o mogućnostima alata za modeliranje i dokumentiranje moguć je i uvoz gotove strukture programskog koda i automatizirano stvaranje dijagrama iz danog koda. Ovdje će se najprije pokazati pojedinačni dijagrami razreda za svaki paket kako su generirani alatom ali također i primjer kombiniranja razreda iz različitih paketa.
Unutar svakog paketa stvara se dijagram razreda tog paketa (poimence 'mongo', 'controller' i 'factory') uz pojavu niza razreda koji su sadržaj paketa, odnosno novostvorenih dijagrama. Tako sadržaj paketa 'controller' čine razredi upravljanja pojedinim instancama sustava, Slika 2.7.10. Paket 'factory' sadrži razred povezivanja s bazom podataka, Slika 2.7.11. Sadržaj paketa 'mongo' čine razredi upravljanja podacima i čine dio podatkovnog modela sustava, Slika 2.7.12.



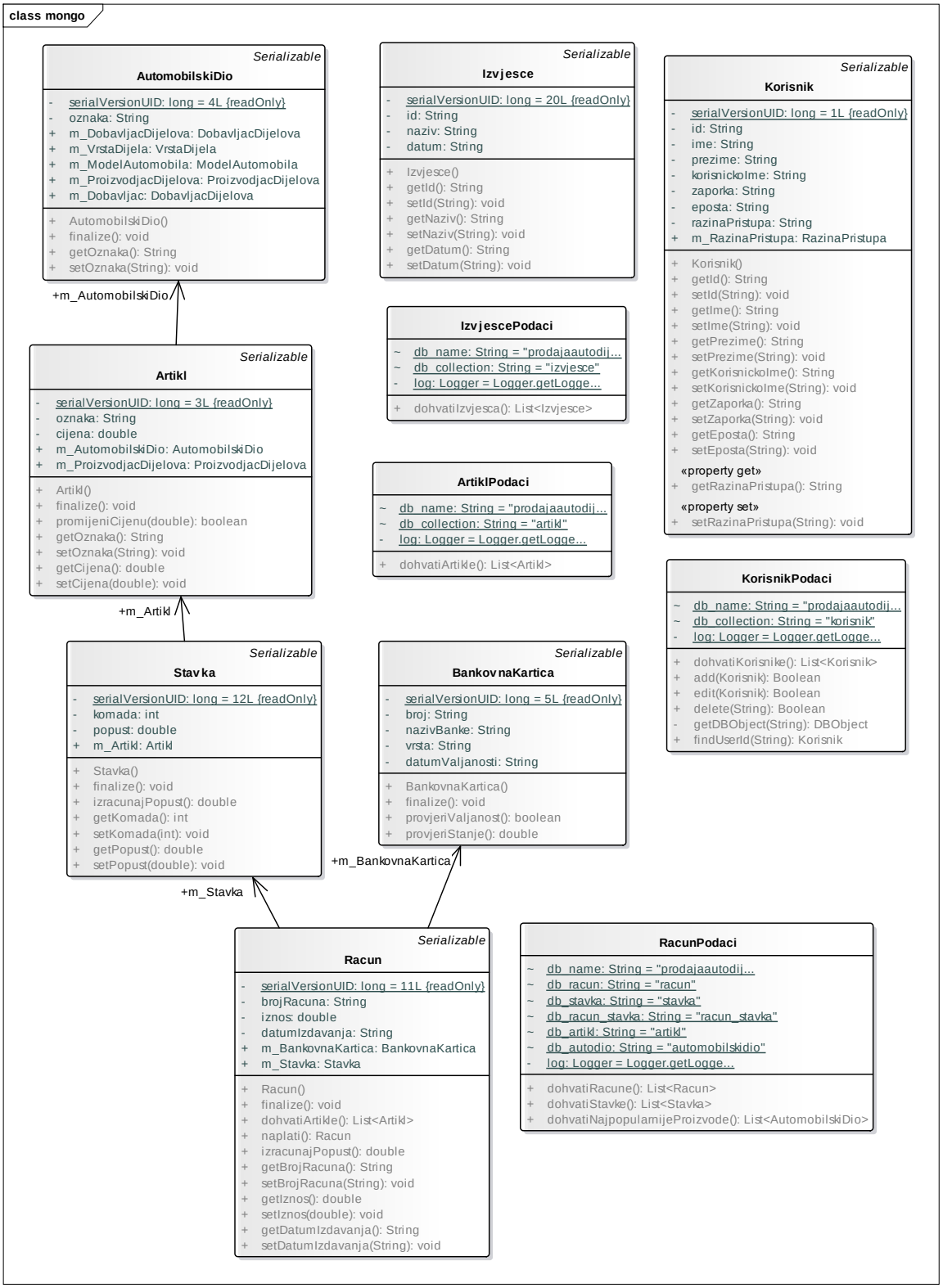
Slika 2.7.10 Razredi upravljačkog dijela programske potpore (paket 'controller')



Slika 2.7.11 Razred veze prema bazi podataka (paket 'factory')

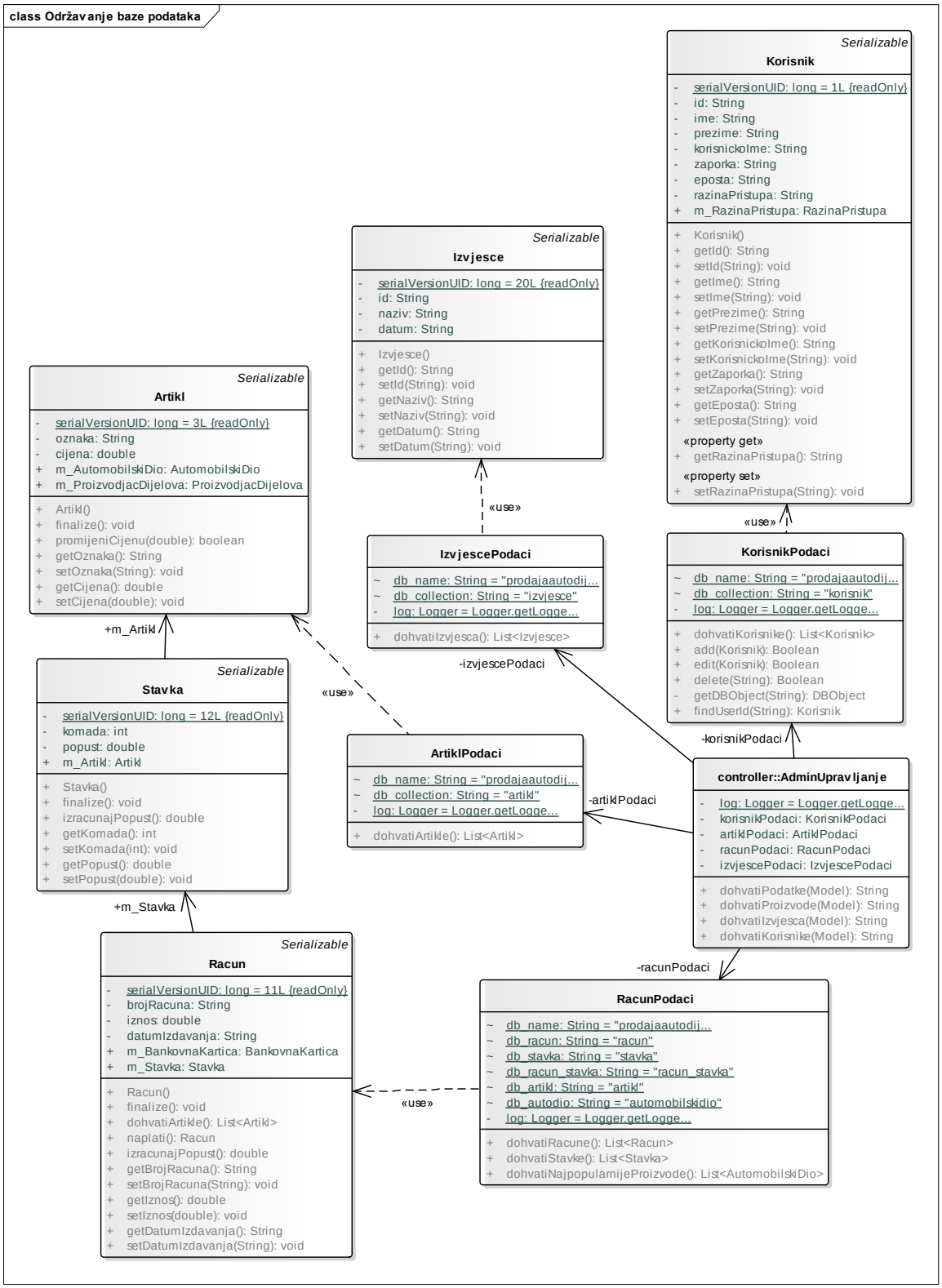
Generirani su dijagrami razreda po paketima kao logičkim cjelinama implementacije gdje je dio navedenih razreda paketa 'mongo' nastao na temelju generiranja programskog koda iz prethodno oblikovanih dijagrama razreda na konceptualnoj razini (kako je opisano u poglavlju „Izrada izvornog koda“). Na dijagramu je naznačen tek dio veza pridruživanja između razreda koji su prethodno bili definirani, a veći dio razreda je ostao nepovezan. Zato je moguće nadopunjavati pojedine dijagrame vezama koje u potpunosti opisuju odnose između pojedinih

razreda, a moguće je i objedinjavati dijagrame kombiniranjem razreda različitih paketa, Slika 2.7.12.



Slika 2.7.12 Razredi dohvata i upravljanja podacima (paket 'mongo')

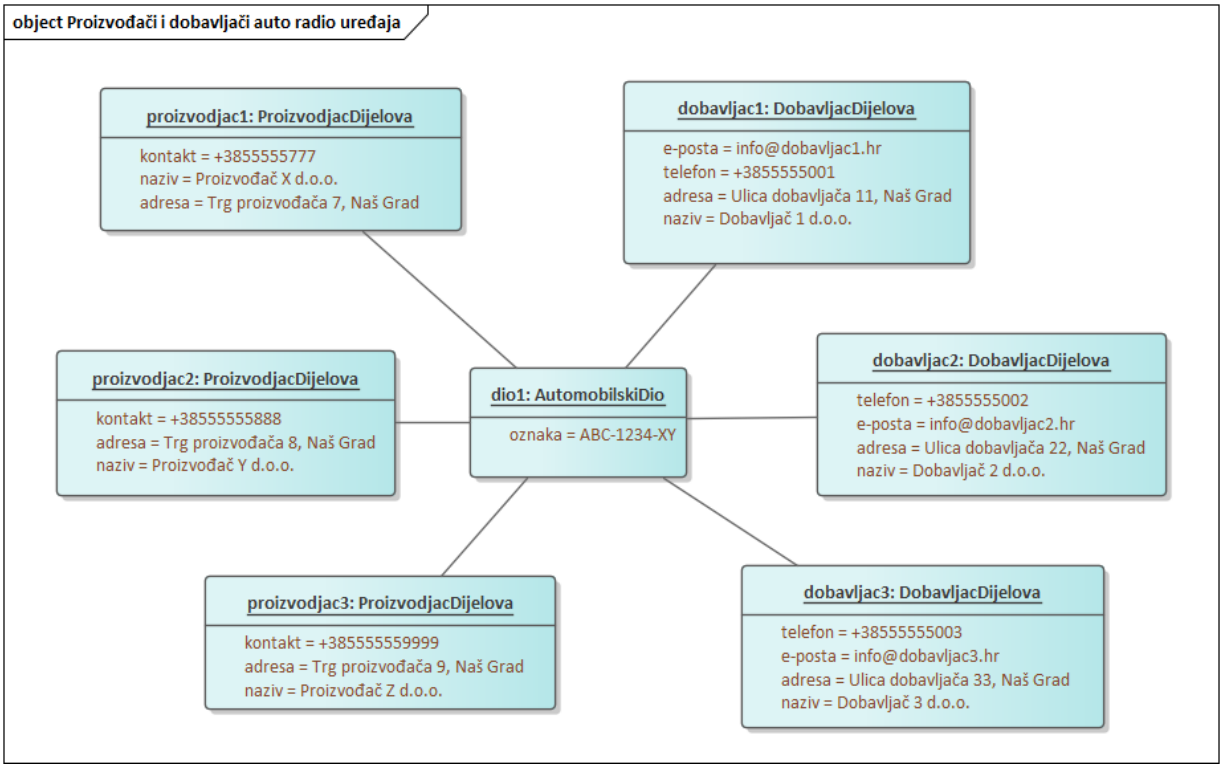
Primjer izbora i kombiniranja razreda iz različitih paketa, odnosno dopuna dijagrama razreda dohvata i upravljanja podacima s elementima bitnim za prikaz administratorskih ovlasti, zaključuje se dijagram razreda za tu cjelinu, Slika 2.7.13. To predstavlja primjer pristupa oblikovanju dijagrama putem uvoza programskog koda dijela razvijenog sustava s potrebnim doradama za zaokruživanje cjeline koju taj dio predstavlja.



Slika 2.7.13 Dijagram razreda ostvarenja administratorskih ovlasti, kombiniranje razreda iz više paketa

2.8. Dijagram objekata

Dijagram objekata (engl. *object diagram*) je usko vezan uz dijagram razreda jer jedan objekt predstavlja instancu ili pojedinca razreda. U notaciji se objekt razlikuje od razreda jer unutar pravokutnog oblika ima dva umjesto tri odjeljka, tj. svaki objekt ima naziv u kojem je naznačeno da se radi o instanci nekog razreda te atribute koji imaju konkretne vrijednosti. Primjerice, za dio dijagrama razreda koji se odnosi na dobavljače i proizvođače automobilskih dijelova (Slika 2.7.3) prikazuje se za neki automobilski dio (s kataloškom oznakom „ABC-1234-XY“) o kojim se konkretno dobavljačima i proizvođačima tog dijela radi, Slika 2.8.1.



Slika 2.8.1 Dijagram objekata dobavljača i proizvođača nekog auto-dijela

Dijagram objekata predstavlja stanje sustava u nekom konkretnom trenutku te prikazuje konkretne odnose između objekata povezanih običnom vezom pridruživanja na temelju dijagrama razreda iz kojeg je izveden. Na njemu se izričito prikazuju višestrukosti s dijagrama razreda koje predstavljaju konkretne instance pojedinih razreda što pomaže u razumijevanju složenijih dijagrama razreda. U slučaju na prethodnoj slici postoji više dobavljača i više proizvođača za konkretni automobilski dio što je omogućeno višestrukostima naznačenima na dijagramu razreda. Na dijagramu objekata za razliku od dijagrama razreda se koriste samo obične veze pridruživanja, tj. svi mogući odnosi s dijagrama razreda se svode na jednostavan odnos pridruživanja ili asocijacije.

Dijagram objekata ima sličnosti s dijagramom komunikacije koji također prikazuje povezanost i odnose između objekata uz dodatni prikaz međudjelovanja između objekata.

2.9. Dijagram stanja

Dijagram stanja (engl. *state machine*, *statechart*) prikazuje ponašanje sustava i objekata u sustavu u pogledu stanja u kojima se mogu naći tijekom njihova životnog vijeka. Može se modelirati promjena stanja nekog objekta, ili dijela podsustava, primjerice korisničkog sučelja. Promjena stanja objekta je pri tome izričito vođena **događajima** (engl. *event*) kao poticajima za promjenu stanja.

U odnosu na dijagram slučajeva uporabe i dijagrame interakcije ovaj dijagram prikazuje procese na još finijoj razini detalja, preciznije definirajući ponašanje objekata nekog razreda. Semantika dijagrama podrazumijeva da se definiraju moguća stanja objekta čije se ponašanje želi opisati te definiraju situacije prijelaza između pojedinih stanja. U notaciji prijelaza se navede događaj ili okidač (engl. *trigger*) koji uzrokuje prijelaz između stanja, mogući **uvjet** (engl. *guard*) pod kojim se prijelaz događa te **akcija** (engl. *action*) koja se može izvršiti prilikom prijelaza. Ta semantika se u prikazu linije prijelaza između stanja označava notacijom *događaj[uvjet]/akcija* te se može čitati: „KADA se dogodi *događaj*, AKO je zadovoljen *uvjet*, dolazi do PRIJELAZA između stanja, uz IZVRŠENJE *akcije*".

Nadalje, unutar stanja se događaju određene akcije gdje se razlikuju akcije pri ulasku u stanje (vrsta događaja *entry*), tijekom trajanja određenog stanja (vrsta događaja *do*) i prilikom izlaska iz stanja (vrsta događaja *exit*) što predstavlja skup predefiniranih vrijednosti događaja ili okidača za pojedine akcije unutar stanja. Osim njih, mogu se definirati i dodatni unutarnji okidači koji pokreću specifične akcije unutar nekog stanja.

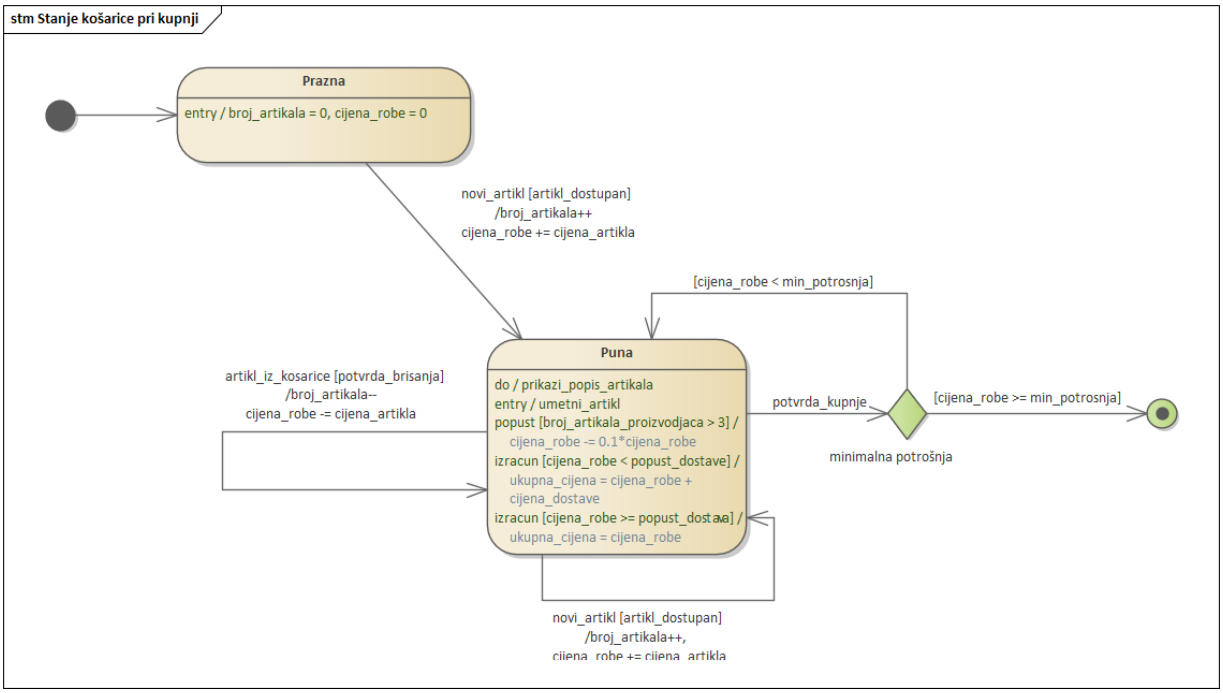
U sustavu prodaje auto-dijelova dijagramom stanja bi se primjerice mogle prikazati promjene stanja košarice pri kupovini proizvoda, promjene stanja narudžbenice pri narudžbi novih proizvoda i promjena stanja baze podataka tijekom procesa sigurnosne pohrane podataka. Isto tako, moglo bi se sa strane korisničkog sučelja prikazati promjene u izgledu web sučelja tijekom tih aktivnosti odnosno izmjena akcija kod kupovine i narudžbe proizvoda.

Kupnja proizvoda

Slika 2.9.1 prikazuje dijagram stanja korisničke košarice pri kupnji novih proizvoda. Zaobljeni pravokutnici na dijagramu predstavljaju stanja objekta košarice, a prijelazi između stanja su uvijek vođeni događajima odnosno akcijama korisnika: klik mišem ili izbor dodavanja novog artikla (naziv događaja na dijagramu je 'novi_artikl'), klik na artikl iz košarice ('artikl_iz_kosarice') kao izbor micanja artikla iz košarice i klik ili izbor potvrde kupnje ('potvrda_kupnje'). Pri tome se već na prijelazima događaju određene akcije: povećanje brojača artikala u košarici i promjena (uvećanje ili umanjenje) ukupne cijene košarice, a što se može događati pod određenim uvjetima: artikl je dostupan i korisnik je učinio potvrdu micanja iz košarice.

Primjeri akcija koje se događaju unutar stanja su: umetanje artikla u košaricu, prikaz popisa artikala, promjena cijene s obzirom na cijenu dostave i s obzirom na ostvarivanje popusta. U

ovom primjeru nakon određene cijene košarice dostava postaje besplatna, a cijena košarice treba premašiti određeni iznos da bi se narudžba košarice mogla dovršiti. Zato je nakon potvrde kupnje stavljen čvor odluke kojim se prikazuje hoće li se kupnja prihvatiti ili ne.

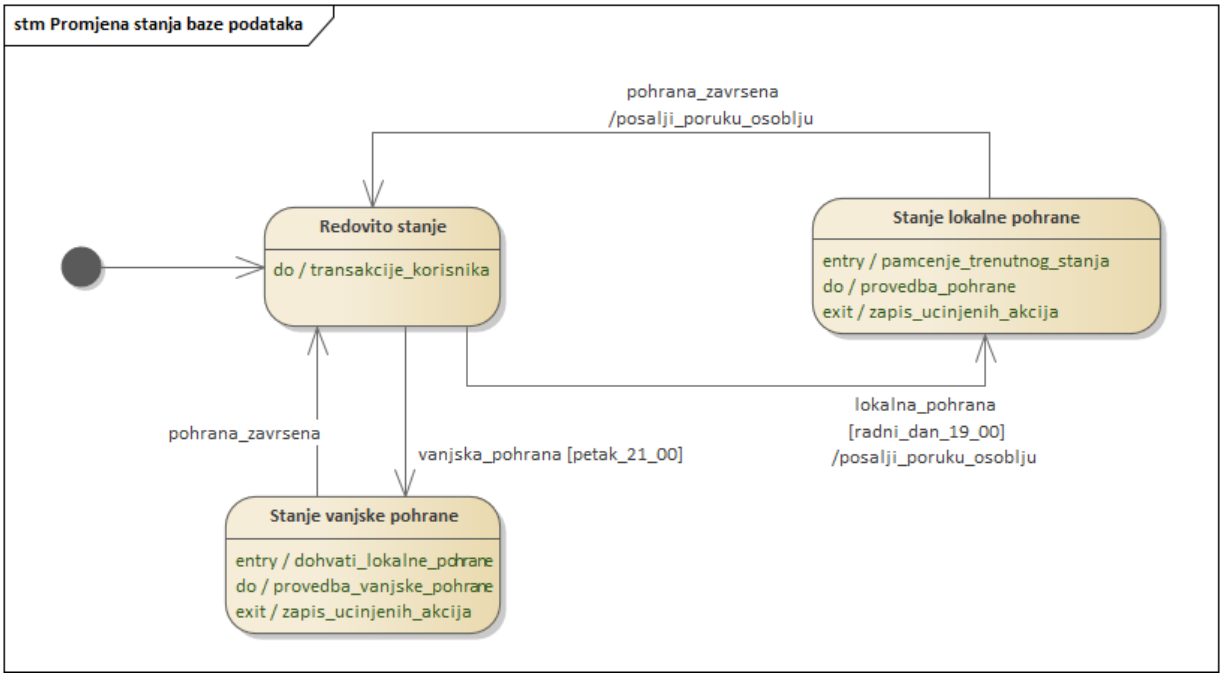


Slika 2.9.1 Dijagram stanja košarice pri kupnji proizvoda

Dijagram sadrži nekoliko **pseudostanja** koja ne predstavljanju stanja objekta u strogom smislu već se njima dodatno oblikuje dijagram i tijek promjene stanja u njemu. To su prije svega **početno i završno stanje** dijagrama kojima dijagram započinje i završava. Prikazuju se notacijom zacrnjenih kružića gdje je takav kružić za završno stanje dodatno zaokružen. Može biti jedno početno i više završnih stanja dijagrama. Od češće korištenih pseudostanja tu je upotrijebljen i prikazan **čvor odluke** (engl. *decision*) s notacijom romba kojim se ostvaruje uvjetno grananje unutar dijagrama.

Baza podataka

Slika 2.9.2 prikazuje dijagram stanja baze podataka tijekom radnog tjedna. Baza podataka prolazi kroz tri različita stanja: redovito stanje tijekom poslovanja tvrtke za prodaju auto-dijelova, stanje za vrijeme trajanja lokalne pohrane i stanje za vrijeme trajanja vanjske pohrane. Pri tome se, osim provođenja transakcija korisnika i provedbe pohrane, odvijaju određene akcije kao što su slanje poruka osoblju, trgovcima i administratoru, pamćenje trenutnog stanja podataka i transakcija i stvaranje zapisa učinjenih akcija.



Slika 2.9.2 Dijagram stanja baze podataka tijekom radnog tjedna

Od pseudostanja upotrijebljeno je početno stanje bez završnih stanja s obzirom na narav sadržaja koji dijagram prikazuje. Baza podataka je neprestano uvijek u nekom od aktivnih stanja, bez prekida rada.

Stanja u dijagramu stanja mogu biti složena gdje pojedina stanja mogu imati podstanja kao primjer jednostavnijih složenih stanja ili cijeli ugniježđeni stroj stanja kao još napredniju mogućnost prikaza.

Od ostalih pseudostanja postoji i **čvor spajanja** (engl. *junction*) kao jedno od mogućih pseudostanja s notacijom podebljane crne točke i značenjem spajanja različitih grana nakon uvjetnih grananja. Također, postoje i pseudostanja kojima se označavaju usporedna ili paralelna stanja su **čvor račvanja** (engl. *fork*) i **čvor skupljanja** (engl. *join*) s istovjetnom notacijom kao kod dijagrama aktivnosti.

.

Česte pogreške

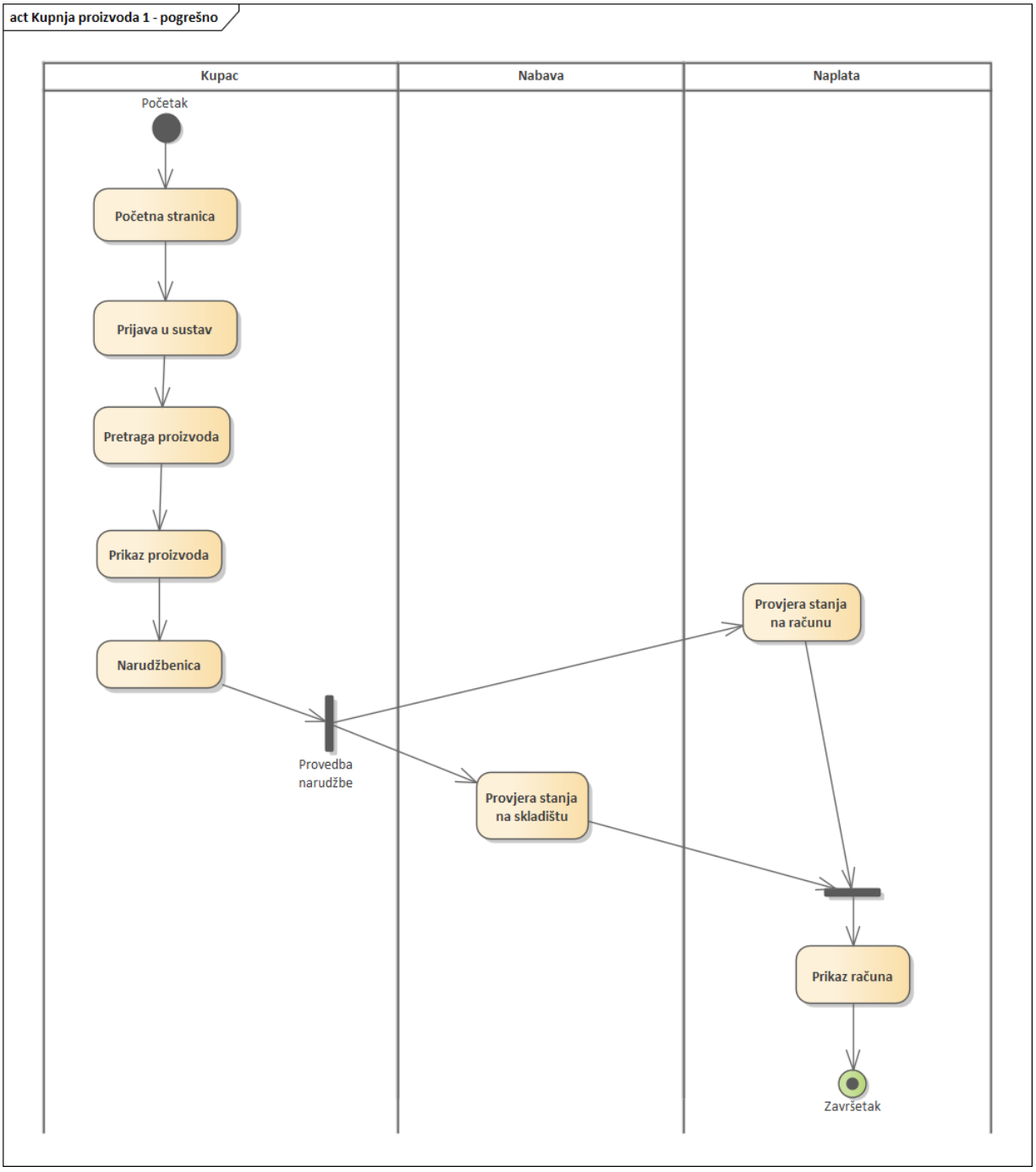
Dijagram stanja notacijski u pojedinim elementima ima sličnosti s dijagramom aktivnosti, ali je semantički i u namjeni bitno različit od dijagrama aktivnosti. Dijagram aktivnosti prikazuje upravljački tok aktivnosti odnosno akcija pojedinih objekata ili sudionika sustava koje se u pravilu izmjenjuju bez izričitih poticaja u obliku signala ili događaja dok dijagram stanja prikazuje izmjene stanja nekog objekta u sustavu. Kao što je već naglašeno, u dijagramu stanja promjena stanja objekta se izričito događa vođeno događajima kao poticajima za promjenu stanja.

Zajednička im je notacija nekih upravljačkih čvorova i pseudostanja zbog čega nerijetko dolazi do zamjene notacije i semantike dijagrama aktivnosti s notacijom i semantikom dijagrama stanja: dijagram aktivnosti notacijom zaobljenih pravokutnika prikazuje akcije ili aktivnosti, a dijagram stanja sličnom notacijom prikazuje stanja nekog objekta. Akcije se u dijagramu stanja nalaze navedene na prijelazima između stanja ili unutar stanja. Na taj način se ta dva dijagrama međusobno nadopunjuju gdje dijagram aktivnosti može preciznije modelirati slijed događaja i aktivnosti koji upravljaju prijelazima u dijagramu stanja.

Konkretnije su u ovdje danim primjerima dijagramom aktivnosti detaljnije prikazani slučajevi uporabe kupnje i narudžbe proizvoda te proces sigurnosne pohrane baze podataka, a dijagramom stanja su prikazane promjene stanja košarice pri kupovini proizvoda i promjena stanja baze podataka tijekom procesa sigurnosne pohrane podataka. Moglo bi se reći da dijagram aktivnosti predstavlja model upravljan podacima, a dijagram stanja model upravljan događajima.

Slika 2.9.3 prikazuje primjer dijagrama s pomiješanom ili nejasnom semantikom te dvije vrste dijagrama. Započeta je izrada dijagrama aktivnosti za proces kupnje proizvoda, ali pojedini elementi koji bi trebali predstavljati akcije koje se obavljaju tijekom te aktivnosti imaju djelomično ili naglašeno semantiku stanja što otežava razumijevanje i dovodi u pitanje smisao izrađenog dijagrama.

Primjerice, akcija 'Početna stranica' ne predstavlja nikakvu posebnu aktivnost već stanje korisničkog sučelja u nekom trenutku. Isto tako, akcija 'Prikaz proizvoda' zasigurno ne upućuje da se radi o nekoj akciji korisnika već također o stanju korisničkog sučelja nakon akcije pretrage proizvoda. Moglo bi se pretpostaviti da korisnik u tom vremenu zapravo pregledava detalje proizvoda te bi se to, ako je istina, trebalo jasnije navesti u navodu unutar čvora akcije. Nadalje, akcija 'Narudžbenica' eventualno upućuje da bi kupac u tom vremenu pokreće narudžbu, ali je takvim nazivom akcije previše prostora ostavljeno nejasno, tj. za nagađanje u pogledu tumačenja dijagrama. Slično svemu prethodno rečenome moglo bi se komentirati i za završnu akciju na dijagramu 'Prikaz računa' što samo po sebi ne potvrđuje da je i provedena naplata.



Slika 2.9.3 Primjer miješanja semantike dijagrama aktivnosti i dijagrama stanja

Semantika pojedinih čvorova akcije u primjeru prikazanog dijagrama aktivnosti je isprepletena sa semantikom stanja te je na dijagramu potrebno jasno uzeti u obzir da se radi o semantici akcija ili aktivnosti. Dodatnu težinu pri tome može dati činjenica da su čvorovi podijeljeni particijama po pojedinim sudionicima gdje bi trebalo biti jasno tko što obavlja u prikazanom slučaju uporabe sustava.

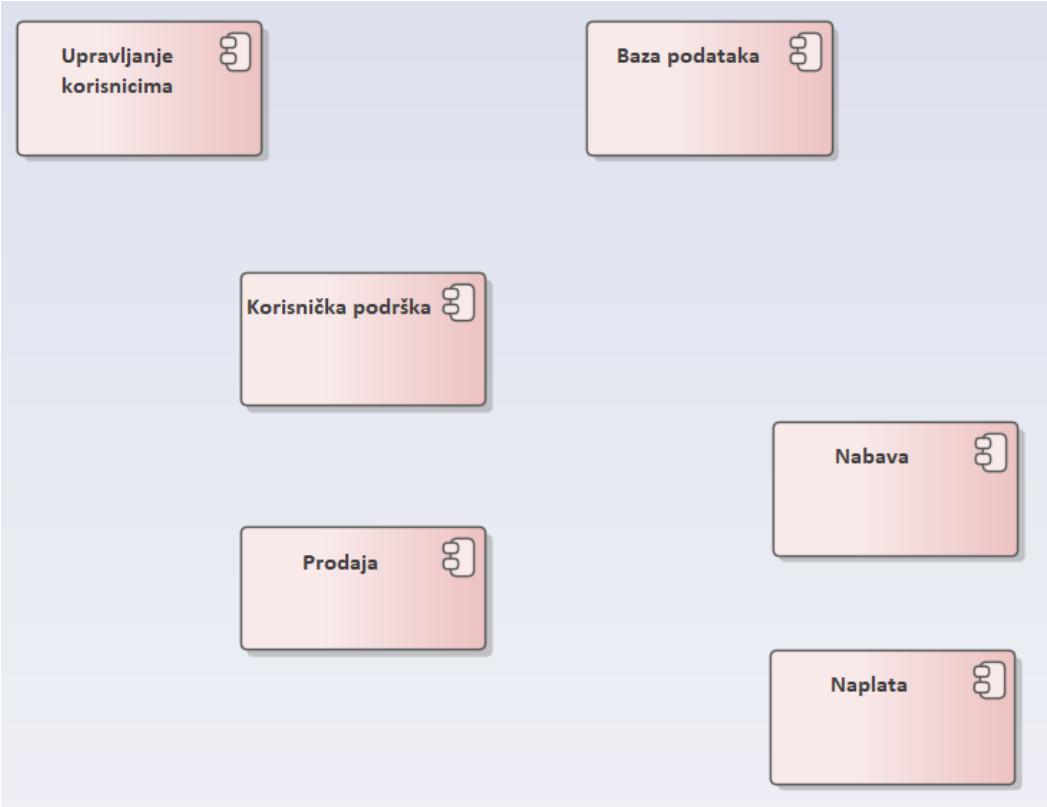
2.10. Dijagram komponenti

Dijagram komponenti (engl. *component diagram*) pripada skupini strukturnih UML-dijagrama koji neposredno prikazuje unutarnju strukturu proizvoda i artefakte nastale tijekom procesa projektiranja i razvoja.

Komponenta kao logička cjelina

Cjelokupni sustav prodaje auto-dijelova bi se mogao opisati na razini komponenti koje opisuju temeljne dijelove sustava (odnosno podsustave) bez posebnih navoda naziva fizičkih datoteka pa će se najprije u ovom slučaju tako graditi i dijagram komponenti sustava. Prikazat će se komponente kao **logički zaokružene cjeline** s jasnim odgovornostima unutar sustava koje mogu činiti sastavne dijelove pojedinih paketa i u sebi sadržavati jednu ili više datoteka ostvarenja sustava.

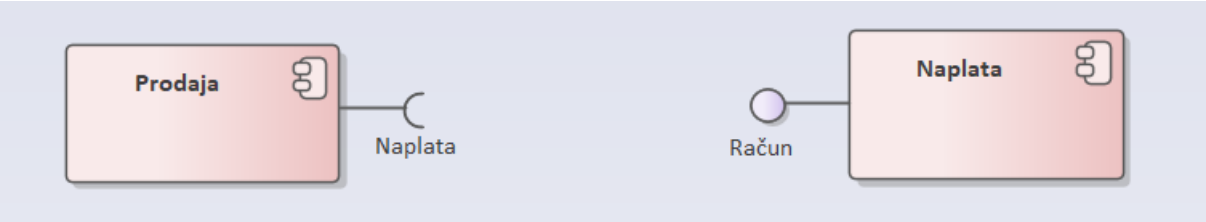
Sustav se temeljno sastoji od dvaju glavnih dijelova, 'Prodaja' i 'Upravljanje korisnicima', što ih čini prvim zasebnim komponentama opisa. Podsustav 'Prodaja' je blisko vezan uz podsustav 'Korisnička podrška' koji, kao i podsustav 'Upravljanje korisnicima', dohvaćaju podatke iz komponente 'Baza podataka'. Osim njih, tu se još važne komponente podsustava 'Nabava' i 'Naplata', Slika 2.10.1.



Slika 2.10.1 Prikaz komponenti unutar dijagrama komponenti

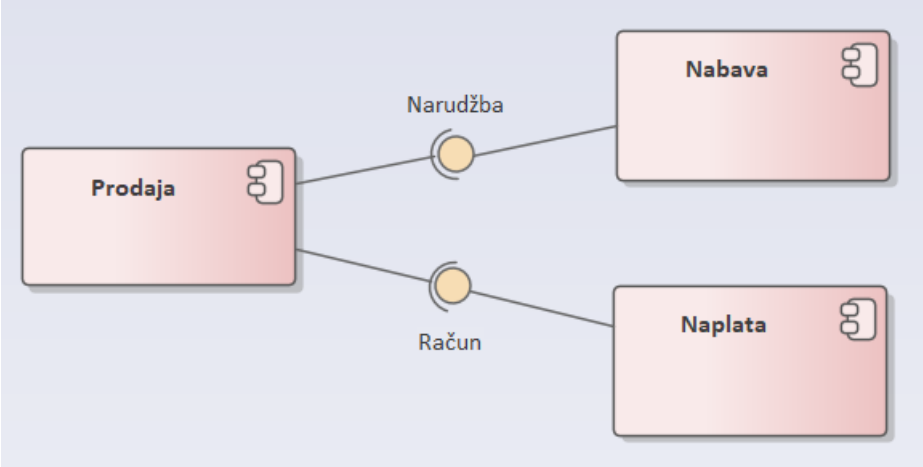
Svaka komponenta podrazumijeva posjedovanje **sučelja** (engl. *interface*) koja mogu biti dvije vrste: ono koje je realizirano ili „ponuđeno“ (engl. *provided*) sučelje i ono koje koristi ili „zahtijevano“ (engl. *required*) sučelje. Primjerice, komponenta 'Prodaja' potrebuje uslugu naplate te se to prikazuje zahtijevanim sučeljem 'Naplata'. U isto vrijeme komponenta 'Naplata' nudi uslugu naplate računa te se to prikazuje ponuđenim sučeljem 'Račun', Slika 2.10.3.

Ponuđeno sučelje se prikazuje kružićem ili „lopticom“ (engl. *ball*) istaknutim od strane komponente koja ga nudi (ili engl. *lollipop*), a zahtijevano sučelje istaknutim polukrugom (ili engl. *socket*) od strane komponente koja ga zahtijeva.



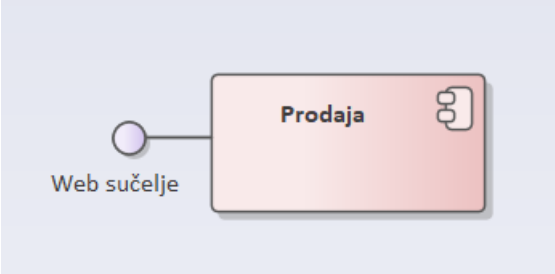
Slika 2.10.2 Primjer ponuđenog i zahtijevanog sučelja komponenti

Ta sučelja se izravno povezuju u prikazu na način da se s obzirom na komplementarnu grafičku notaciju „uglavljuju“ jedno u drugo (ili engl. *assembly*) te s obzirom na to da ponuđeno sučelje komponente 'Naplata' ispunjava zahtjeve sučelja za provedbu naplate komponente 'Prodaja', Slika 2.10.3. Osim toga, u ovom primjeru radi se o tome da komponenta 'Prodaja' zahtijeva i sučelje kojim se želi ostvariti narudžbu, a komponenta 'Nabava' taj zahtjev ispunjava svojim ponuđenim sučeljem. Ponuđenom sučelju je postavljen naziv 'Narudžba' prilikom same definicije.



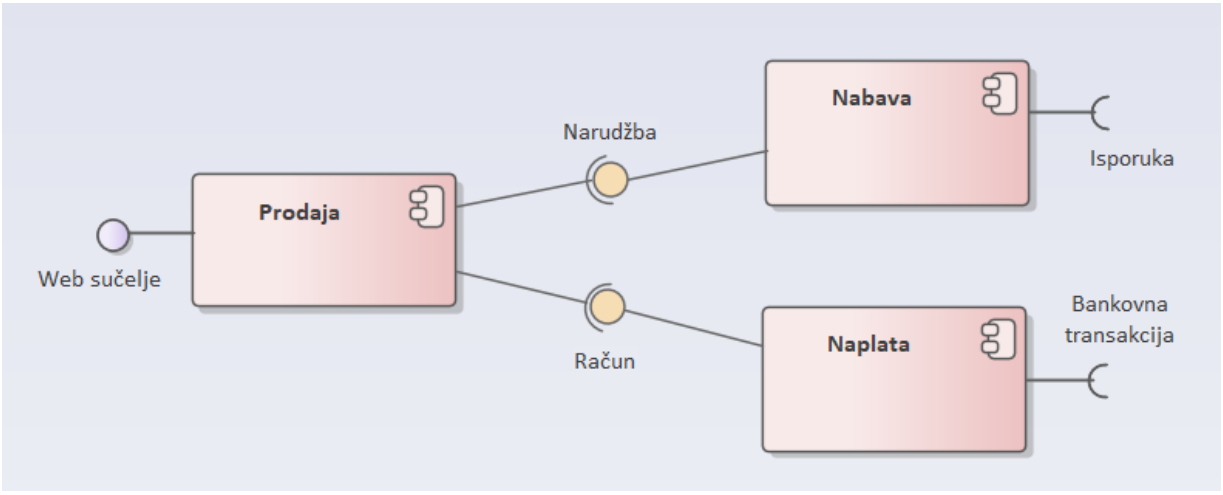
Slika 2.10.3 Ponuđena i zahtijevana sučelja komponenti

Na jednom dijagramu nije nužno imati sva sučelja „uparena“, tj. ne mora svako ponuđeno sučelje biti povezano s nekim zahtijevanim sučeljem niti obratno. U svakom slučaju potrebno je naznačiti da neko sučelje postoji a može ga koristiti i neka komponenti izvan granica sustava koji se nekim dijagramom prikazuje. Time se označava i vanjsko sučelje sustava koji se prikazuje. Npr., komponentu 'Prodaja' koriste javni ili registrirani korisnici weba putem web sučelja koje komponenta nudi i nije nužno na dijagramu prikazati ništa osim toga da navedeno sučelje postoji i da ga mogu koristiti web korisnici, Slika 2.10.4.



Slika 2.10.4 Ponuđeno javno sučelje komponente

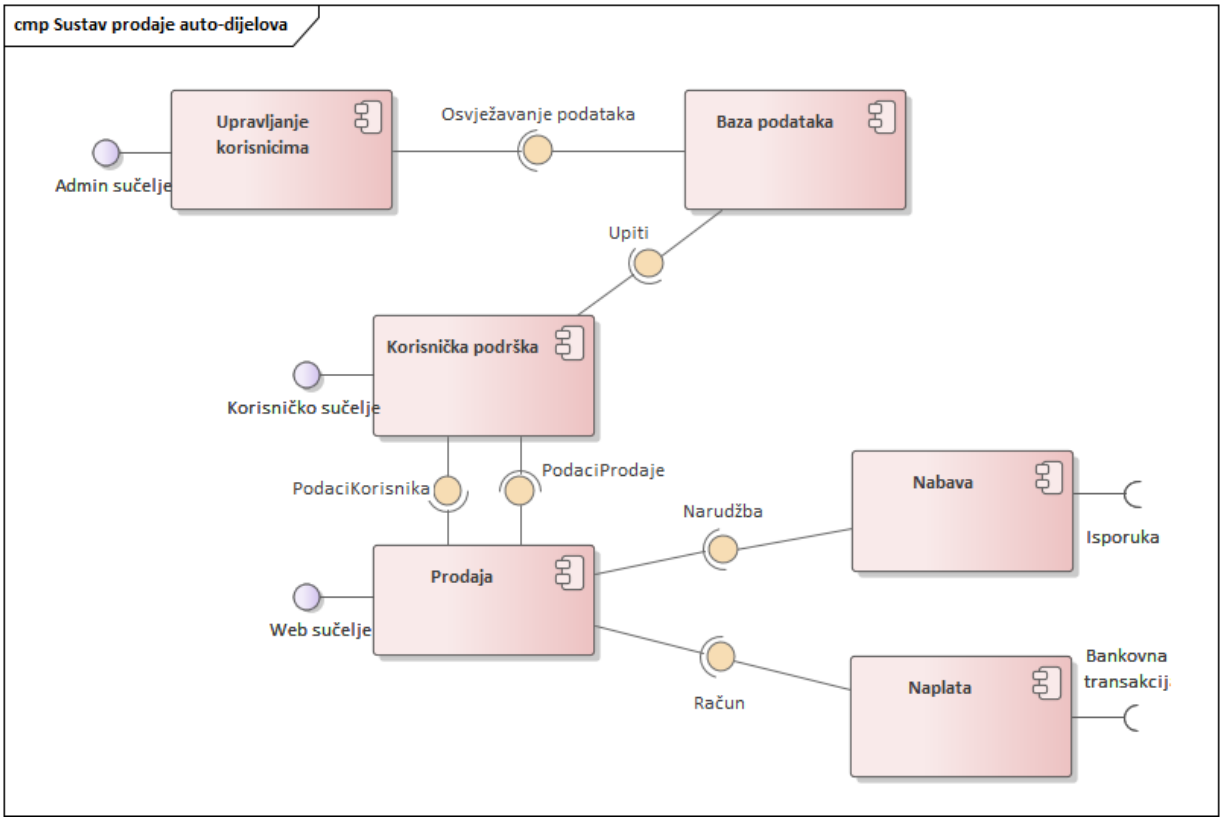
Isto tako, komponente 'Nabava' i 'Naplata' za ispunjavanje svojih funkcionalnosti zahtijevaju komunikaciju s vanjskim klijentima. Primjerice, komponenta 'Nabava' treba uslugu isporuke naručenih artikala od vanjskog dobavljača te se to prikazuje posebnim zahtijevanim sučeljem 'Isporuka', a komponenta 'Naplata' uslugu naplate očekuje putem bankovne transakcije koju provodi vanjska usluga te se to u ovom slučaju prikazuje posebnim zahtijevanim sučeljem 'Bankovna transakcija', Slika 2.10.5.



Slika 2.10.5 Prikaz različitih vrsta sučelja komponenti

Općenito, za cjelokupni sustav prodaje auto-dijelova može se nadalje reći da je komponenta 'Prodaja' blisko vezana uz komponentu 'Korisnička podrška' koja se, kao i komponenta 'Upravljanje korisnicima', veže na komponentu 'Baza podataka'. Između komponenti 'Prodaja'

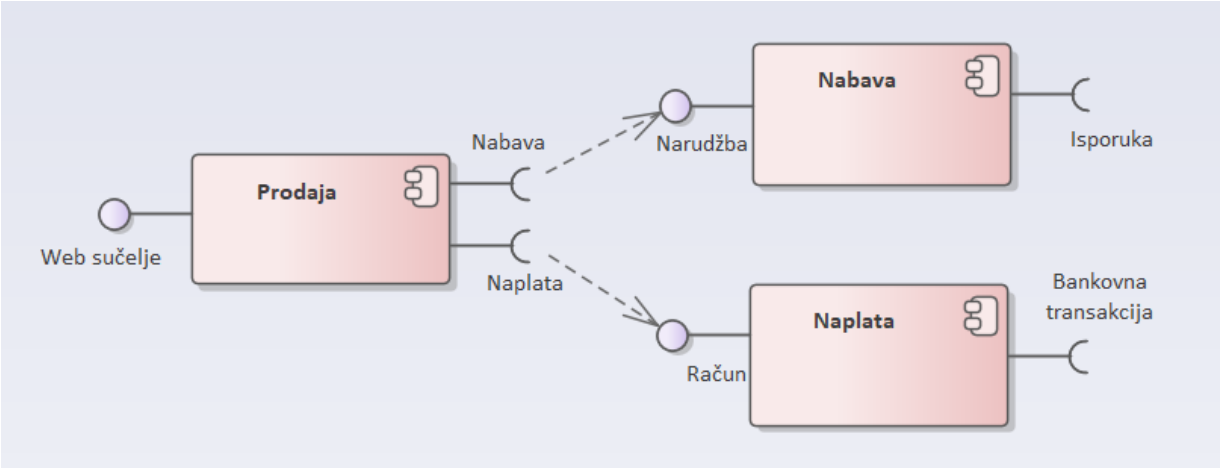
i 'Korisnička podrška' definiraju se sučelja: sučelje 'PodaciProdaje' kojim korisnička podrška provjerava podatke o prodaji za pojedine korisnike i sučelje 'PodaciKorisnika' kojim funkcionalnost prodaje ima na raspolaganju informacije o korisniku sa strane korisničke podrške, Slika 2.10.6.



Slika 2.10.6 Dijagram komponenti sustava prodaje auto-dijelova

Odnos ovisnosti tih dijelova sustava o podsustavu 'Baza podataka' oblikovan odnosima njihovih sučelja govori da je sadržaj baze podataka temelj informacijskog sustava za cijelo poslovanje. Opcionalno bi se i neki drugi dijelovi mogli prikazati povezani vezom ovisnosti prema komponenti 'Baza podataka'. Posebice bi se komponenta 'Prodaja' mogla povezati s komponentom 'Baza podataka' izravno, a ne posredno preko komponente 'Korisnička podrška'. Ovdje se, i zbog zadržavanja jednostavnosti prikaza, komponenta 'Baza podataka' prikazuje više kao potpora poslovanju s korisnicima a ne isključivo kao sustav evidencije računa i povijesti prodaje. Slika 2.10.6 prema tome prikazuje jedno od mogućih rješenja dijagrama komponenti za sustav prodaje auto-dijelova na razini prikaza ključnih podsustava kao zasebnih komponenti.

Komponente se mogu povezati na dva načina: izravnim povezivanjem sučelja kako je do sada bilo prikazano ili vezom ovisnosti od zahtijevanog do ponuđenog sučelja. Slika 2.10.7 daje primjer uporabe veze ovisnosti za prikaz povezanosti komponenti.



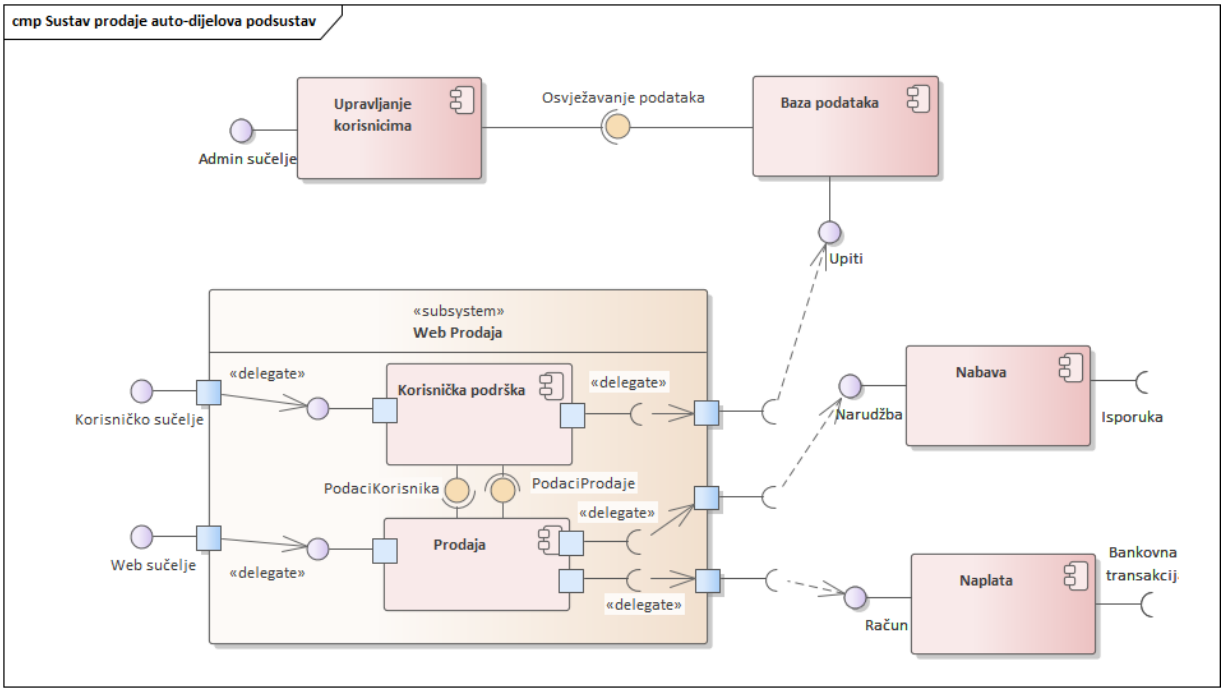
Slika 2.10.7 Prikaz odnosa komponenti uporabom sučelja

U nastavku će se razmotriti dodatne mogućnosti prikaza ostvarenja ovog sustava na dijagramu komponenti.

Hijerarhija komponenti

S obzirom na to da su funkcionalnosti komponenti 'Prodaja' i 'Korisnička podrška' bliže povezane moguće je i uputno to prikazati i kao zaseban podsustav.

Slika 2.10.8 prikazuje cijeli dijagram komponenti s detaljnim prikazom podsustava web prodaje.



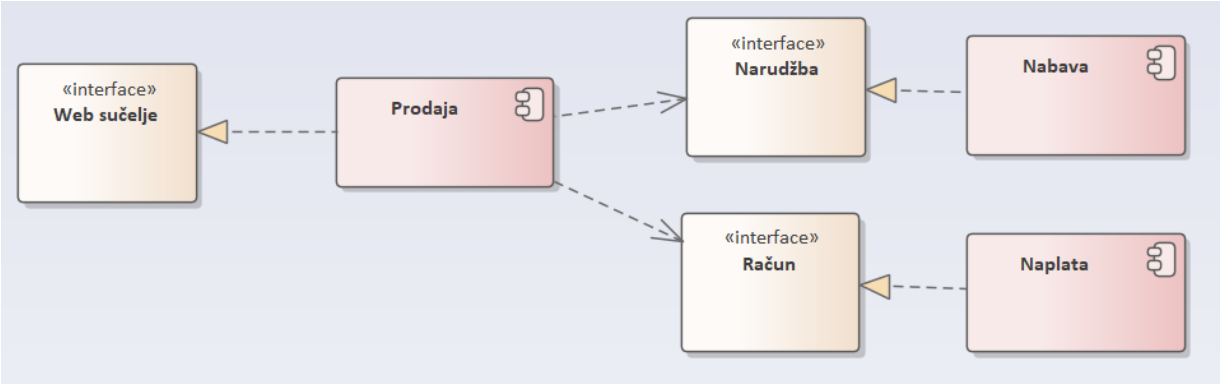
Slika 2.10.8 Dijagram komponenti s prikazom podsustava 'Web Prodaja'

Veza vrste *delegate* predstavlja „zastupničku“ vezu kojom se naznačuje koje unutarnje ponuđeno sučelje se bira („delegira“) za ostvarenje vanjski naznačenog sučelja. Isto tako, ako unutarnje zahtijevano sučelje želi pristupiti nekom vanjskom ostvarenom sučelju onda se tom vrstom veze naznačuje preko kojeg vanjski zahtijevanog sučelja će takvu vezu ostvariti. Ovakva veza zastupanja automatski na sebi ima naznačen opis, odnosno stereotip <<*delegate*>>.

Proširena notacija sučelja

Osim dosadašnjih načina prikaza sučelja komponenti korištenjem sažetijih elemenata sučelja i veza između njih, moguće je definirati i prikazati sučelja u izričitijem ili proširenom obliku, slično kao na dijagramu razreda gdje se posebno prikazuju entiteti sučelja. U takvim slučajevima koristi se veza realizacije čime se prikazuje da neka komponenta ostvaruje navedeno sučelje. Komponenta koja koristi neko sučelje je s njime onda povezana vezom ovisnosti.

Primjer odnosa komponenti prodaje, nabave i naplate može se prikazati uz prošireni prikaz postojećih sučelja, Slika 2.10.9.



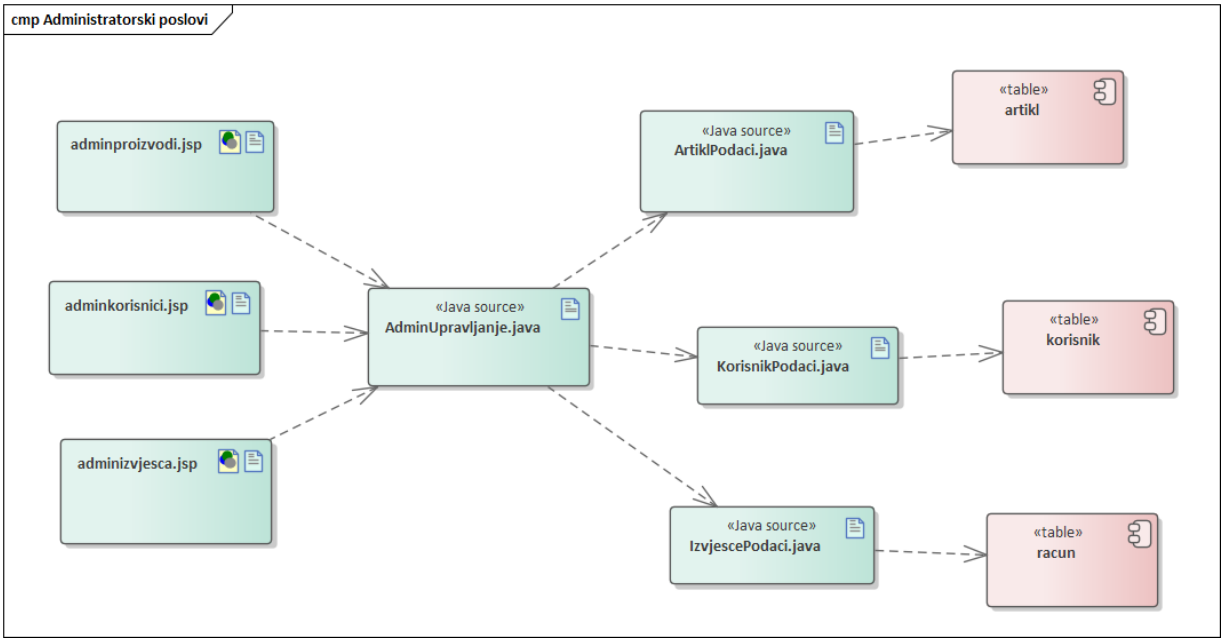
Slika 2.10.9 Primjer prikaza s proširenom notacijom sučelja

Ovisno o tome koristi li se u nekom odnosu veza ovisnosti ili realizacije između komponente vidljivo je tko zapravo nudi ili realizira, a tko koristi ili zahtijeva neko sučelje.

Komponenta kao fizički element

Ako se dijagram komponenti radi na razini fizičke komponente, ili „artefakta“ (engl. *artifact*), često se prije svega misli na datoteke različitih unutarnjih struktura.

Primjer dijagrama komponenti na razini „artefakata“ sustava bio bi prikaz ostvarenja za obavljanje administratorskih poslova a u pogledu ostvarenih datoteka izvornog koda, Slika 2.10.10.



Slika 2.10.10 Dijagram komponenti administratorskih poslova

Administrator sustava obavlja tri vrste posla: održavanje podataka o proizvodima ili artiklima prodaje, osvježavanje podataka o registriranim korisnicima ili kupcima i stvaranje izvješća o prodaji. Datoteke vrste JSP, 'adminproizvodi.jsp', 'adminkorisnici.jsp' i 'adminizvjesca.jsp' pripremaju izgled administratorskih sučelja za svaku od navedenih zadaća (i predstavljaju pogled ili *view* dio obrasca MVC). Središnja datoteka 'AdminUpravljanje.java' upravlja svim zadaćama uz pomoć datoteka za dohvat podataka iz baze podataka (te predstavlja *controller* dio obrasca MVC). Te pomoćne datoteke 'ArtiklPodaci.java', 'KorisnikPodaci.java' i 'IzvjescePodaci.java' izravno komuniciraju i dohvaćaju podatke iz baze podataka, prije svega iz tablica baze podataka 'artikl', 'korisnik' i 'račun' (što predstavlja *model* dio obrasca MVC).

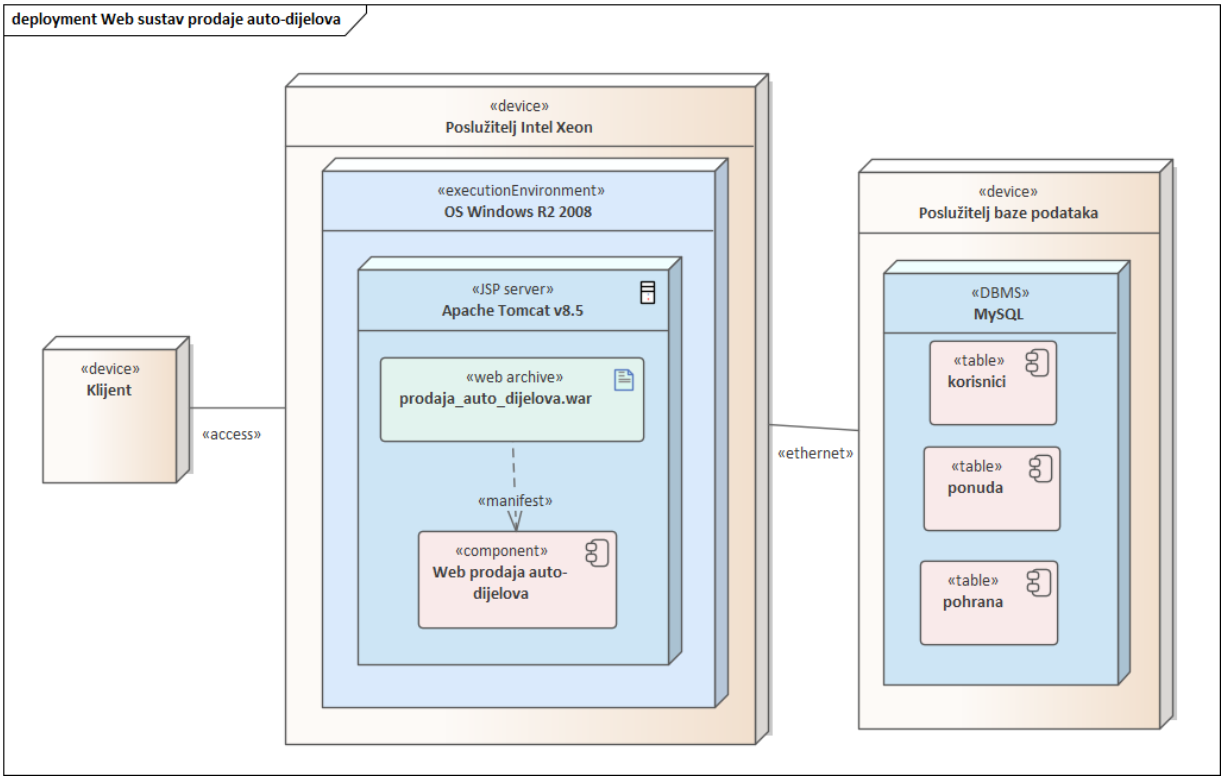
Definirani su stereotipovi svih komponenti koji se prikazuju tekstualno, u svom osnovnom obliku, iznad naziva komponente, osim za JSP datoteke gdje se ugrađeni stereotipovi 'JSP page' prikazuju u ikoniziranom obliku desno od naziva komponente.

2.11. Dijagram razmještaja

Dijagram razmještaja ili dijagram rasporeda (engl. *deployment diagram*) osim logičkih prikazuje i razmještaj fizičkih komponenti sustava: sklopovske podloge i okoline izvršavanja. Unutar sklopovskih komponenti nalaze se artefakti koji predstavljaju konkretnu realizaciju logičkih komponenti prikazanih na dijagramu komponenti.

Osnovni fizički element je **čvor** (engl. *node*) koji se nudi u dva podrazumijevana oblika, kao tipičnih specijalizacija elementa te vrste: **uređaj** (engl. *device*) i **okolina izvršavanja** (engl. *execution environment*) što se uobičajeno označava s istoimenim stereotipovima. Temelj svakog oblikovanja dijagrama razmještaja je postavljanje čvorova unutar kojih će se postavljati artefakti za čiji se opis podrazumijevano koristi istoimeni stereotip. Osim generičkih stereotipova <<device>>, <<execution environment>> i <<artifact>> mogu se koristiti i specifičniji stereotipovi za opis pojedinih čvorova unutar kojih se nalaze pojedini artefakti. Primjerice, može se koristiti stereotip <<server>> koji označava poslužitelj ili <<OS>> kao oznaku operacijskog sustava kao okoline izvršavanja artefakata, ili još specifičnije navesti vrstu poslužitelja kao primjerice <<application server>> ili <<web server>>. Artefakti se mogu pobliže opisati stereotipovima kao i na dijagramu komponenti, primjerice <<executable>> ili <<library>>.

Primjer dijagrama razmještaja prikazuje fizičku strukturu sustava prodaje auto-dijelova s više različitih čvorova, Slika 2.11.1. Tako se, primjerice, sustav prodaje auto-dijelova nalazi ostvaren na aplikacijskom poslužitelju 'Intel Xeon' s operacijskim sustavom 'Windows R2 2008' te se pokreće na web poslužitelju 'Apache Tomcat v.8.5'. Pojedini uređaji, klijentsko računalo, aplikacijski i poslužitelj baze podataka, su označeni stereotipovima <<device>>. Okoline izvršavanja su, osim stereotipom <<executionEnvironment>> za operacijski sustav aplikacijskog poslužitelja, označeni nešto posebnijim stereotipovima <<JSP server>> na aplikacijskom i <<DBMS>> na poslužitelju baze podataka.



Slika 2.11.1 Primjer dijagrama razmještaja za sustav prodaje auto-dijelova

Čvorovi su međusobno povezani vezama pridruživanja koje se također pobliže opisuju različitim stereotipovima, npr. da se naznači narav odnosa između čvorova. Najčešće se tim odnosima naznačuje korišteni protokol komunikacije između čvorova, a moguće je naznačiti i veze ovisnosti. U ovom primjeru, poslužitelj komunicira s klijentskim računalima i poslužiteljem baze podataka.

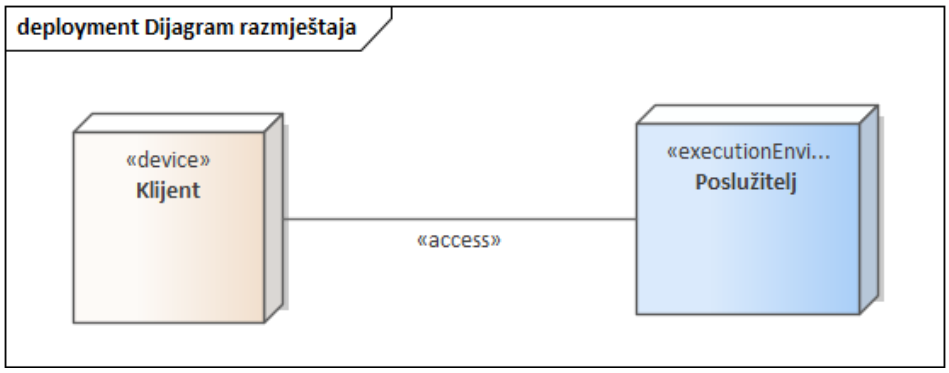
Unutar čvorova prikazuju se artefakti implementacije, odnosno dijelovi implementacije programske potpore. Npr. datoteka web arhive 'prodaja_auto_dijelova.war' kao artefakt ili gotova komponenta programske potpore u ovom konkretnom slučaju predstavlja, očituje ili „manifestira“ logičku komponentu cjelokupnog sustava web prodaje auto-dijelova te se to označava posebnom vrstom veze s odgovarajućim stereotipom. Treba primijetiti da su komponente također označene odgovarajućim stereotipovima koje ih pobliže ili detaljnije opisuju, npr. web arhiva 'prodaja_auto_dijelova.war' je postavljena kao komponenta vrste artefakta a komponenta 'Web prodaje auto-dijelova' kao opća komponenta. To može biti važno da se i vizualno naglasi razlika između njih u pogledu razina apstrakcije na kojoj se nalaze.

Baza podataka je sljedeći logički i fizički bitni element i komponenta sustava prodaje auto-dijelova. Na istovjetan način kako se to učinilo za prikaz aplikacijskog poslužitelja i poslužitelj baze podataka kao drugi temeljni sklopovski element sustava je postavljen u prostor dijagrama razmještaja s naznakama vrste sustava upravljanja bazom podataka i elemenata baze podataka. Logičke komponente baze podataka, relacije ili „tablice“, se ne stvaraju nužno kao artefakti ili izdvojene datoteke, ali ih je u svakom slučaju uputno označiti kao komponente s odgovarajućim stereotipom.

Klijentski uređaji su predstavljeni uopćeno jednostavnim čvorom koji može predstavljati različite vrste računala koje pristupaju sustavu.

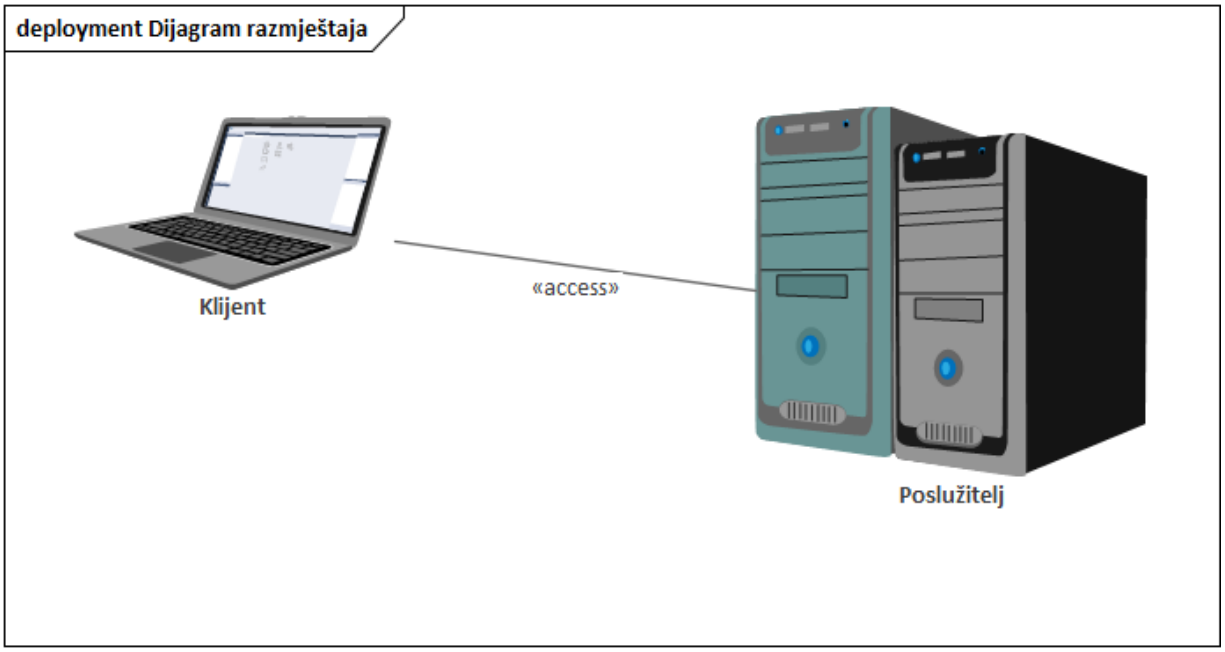
Grafička notacija čvorova

Grafički prikaz elemenata dijagrama razmještaja definiran je na način da su čvorovi prikazani kao pravokutnici s 3-D perspektivom, odnosno kao kvadri, Slika 2.11.2.



Slika 2.11.2 Primjer jednostavnog dijagrama razmještaja

S obzirom na to da čvorove kao sklopovske komponente lako možemo zamisliti u njihovom fizičkom vanjskom izgledu korisno je imati i takvu mogućnost prikaza što ovisi i o mogućnostima alata za modeliranje, Slika 2.11.3.



Slika 2.11.3 Primjer grafički uljepšanog dijagrama razmještaja

3. Modelno usmjereno programsko inženjerstvo

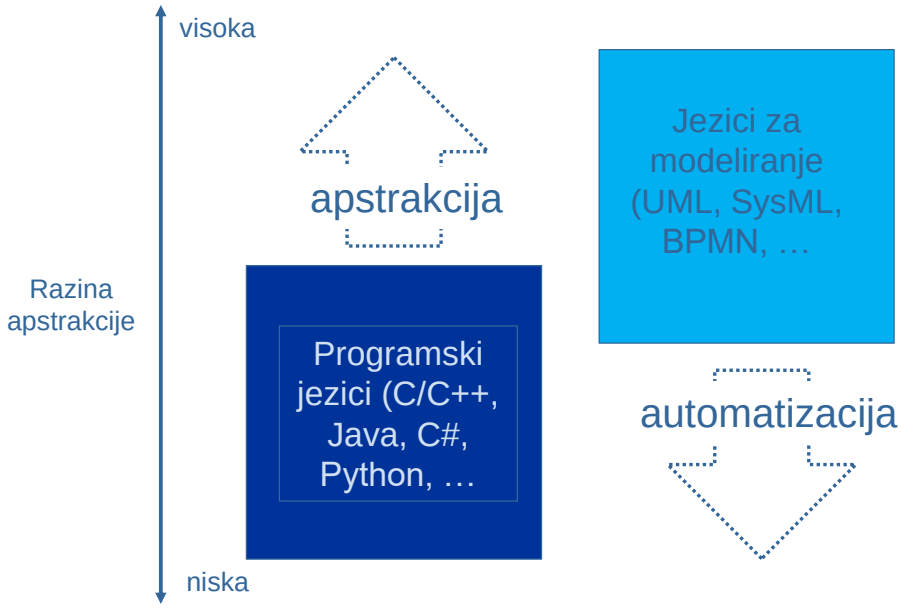
U prethodnom poglavlju se na odabranom primjeru predstavio pristup modeliranja programske potpore za sustav prodaje auto-dijelova jezikom UML. U prvom dijelu više se radilo o modeliranju u sklopu inženjerstva zahtjeva gdje su se oblikovali UML-dijagrami pretežno ponašajnog tipa sa svrhom da se opiše buduća programska potpora sve do točke izrade dijagrama razreda na konceptualnoj razini. Tu se pretpostavila mogućnost alata da na temelju danog modela domene automatizira stvaranje programskog koda generiranjem iz dijagrama razreda. Nastavno na to se pristupilo implementaciji programske potpore te se u drugom dijelu poglavlja radilo više o dokumentiranju gotovog proizvoda. Na temelju programskog koda se pritom generirao dijagram razreda na implementacijskoj razini koji je dalje ručno dorađen da bi se nakon toga oblikovali i drugi UML-dijagrami pretežno strukturnog tipa čime se opis konkretne implementacije prikazao na višoj razini apstrakcije radi lakšeg početnog razumijevanja iste, čime se dokumentiranje razvijene programske potpore zaključilo.
Dva spomenuta mehanizma, **automatizacija** i **apstrakcija** temelj su ideje modelno usmjerenog programskog inženjerstva.
U ovom poglavlju će se ukratko predstaviti i razmotriti određeni pojmovi vezani uz modelno usmjereno programsko inženjerstvo. Najprije će se probati dati definicija tog pristupa razvoju programske potpore, predstaviti vrste modela iz praktične primjene te se osvrnuti na mogućnosti podrške takvom pristupu oblikovanja programske potpore pomoću CASE alata.

3.1. Uvod i definicija

Prema jednoj mogućoj definiciji, **modelno usmjereni razvoj** (engl. *model-driven development*) je pristup razvoju programske potpore gdje su usredotočenost i glavni proizvodni izlazi (artefakti) stavljeni na modele, a ne na programe [15]. Isto tako, može se reći da je to metodologija programskog inženjerstva koja se umjesto na dokumente usredotočuje na modele domene razvoja kao glavnog sredstva razmjene informacija, kao jedna od definicija koju dijeli s modelno temeljenim inženjerstvom sustava (engl. *MBSE – model-based systems engineering*), [35].

S obzirom na uporabu ideje modela, programska potpora ima jedinstveno svojstvo da omogućuje evoluciju apstraktnih modela u potpunu uspostavljenu implementaciju bez mijenjanja fizičkih sredstava nad kojima se radi, alata ili metoda. Također se vodi mogućnošću dobivanja apstraktnog pogleda izravno, na automatizirani način, iz datoteka implementacije. Početni model može biti neprestano rafiniran ili evoluiran sve dok se aplikacija u potpunosti ne specificira čime bi model postao sustav koji se modelira. U konačnici ideja je onda u pretpostavci da je moguća i „savršena” točnost modela programske potpore, s obzirom na to da model i sustav predstavljaju istu stvarnost. U stvarnosti to se nije dostiglo, već je predmet rasprava i razvoja metoda i alata kojima bi se tom cilju uspjelo približiti.

Kako je spomenuto u uvodu poglavlja, ideja se temelji na dva različita pristupa razvoju, a to su: apstrakcija i automatizacija, Slika 3.1.1.



Slika 3.1.1 Osnovni koncepti modelno usmjerenog razvoja (prema [15])

Pri tome mogu se razlikovati dvije skupine jezika. Prvi su klasični programski jezici za implementaciju funkcionalnosti pisanjem programskog koda s velikim skupom mogućnosti za odabir, a drugi su jezici za modeliranje, gdje je UML najpoznatiji od njih.

Razvojem programskog koda razvija se i potreba za apstrakcijom, odnosno prikazom istog na višoj razini apstrakcije, kao mogućnosti opisa radi boljeg razumijevanja vanjskom promatraču. Stvaranje modela programske potpore kao početne točke razvoja pretpostavlja i mogućnost automatizacije gotovog rješenja iz oblikovanih modela.

Pojam modelno usmjerene arhitekture (engl. *MDA – model-driven architecture*) predstavlja upravo način opisa i strukturiranja specifikacije sustava programske potpore uporabom modela. Ideja tog koncepta je odvajanje poslovne i aplikacijske logike od tehnološke platforme.

3.2. Različiti pogledi i vrste modela

O pojmu „modela“ može se različito razmišljati i ovisno o tome raščlanjivati prema različitim kriterijima, gledištima i razinama apstrakcije. U ovom poglavlju ukratko se govori o nekim pojmovima koji se uvode s pristupom modelno usmjerenog inženjerstva i arhitekture sustava.

Sommerville u svojoj knjizi [7] spominje nekoliko vrsta modela s obzirom na različite točke gledišta prilikom modeliranja sustava te ih povezuje s pojedinim vrstama UML-dijagrama. To su vrste modela: kontekstni, interakcijski, strukturni i ponašajni, što ujedno opisuje namjenu i podjelu dotičnih vrsta UML-dijagrama.

S obzirom na vanjsku točku gledišta govori o kontekstnom modelu (engl. *context model*). Pod time se misli da operativni kontekst ili okolinu u kojoj sustav funkcionira. Pri tome je važno

povući ili odrediti granice sustava na što utjecaj imaju postojeći društveni i organizacijski uvjeti. Jedan mogući pogled na to je arhitekturni kontekst koji pokazuje sustav i njegov odnos sa susjednim sustavima. Granice određuju što se nalazi unutar a što izvan sustava te imaju dubok utjecaj na zahtjeve sustava. Drugi mogući pogled na to je kontekst poslovnog procesa koji otkriva kako se sustav razvio unutar šireg konteksta poslovnog procesa. Za prikaz toga pogodan je UML-dijagram aktivnosti koji prikazuje kontekst poslovnog procesa.

Model međudjelovanja ili interakcije (engl. *interaction model*) je ključan u prepoznavanju i definiciji korisničkih zahtjeva prema sustavu, a međudjelovanje prema drugim sustavima stavlja pozornost na pitanja komunikacije između sustava, dok se na to nadovezuju pitanja performansi i pouzdanosti sustava. Uz ove vrste modela vežu se dijagrami slučajeva uporabe i dijagrami slijeda.

Strukturni model (engl. *structural model*) prikazuje organizaciju sustava u pogledu sastavnica ili komponenti koje čine sustav i njihovih odnosa. Ovi modeli su prema tome namijenjeni analizi i oblikovanju arhitekture sustava jer prikazuju strukturu i organizaciju sustava kao ishoda oblikovanja sustava.

Ponašajni model (engl. *behavioral model*) prikazuje dinamiku sustava prilikom njegovog rada, prikazujući što se pri tome sve događa ili što bi se očekivano trebalo događati na poticaje iz okoline sustava. Tu se po naravi prikaza dinamike sustava razlikuju modeli vođeni ili upravljani podacima i modeli upravljani događajima. Tome odgovara i razlika između UML-dijagrama aktivnosti i UML-dijagrama stanja koji se vežu uz modele tako različitih naravi. Kod prvih se opet misli na poslovne sustave koji su primarno upravljani podacima koji ulaze u sustav s relativno malo obrade vanjskih poticaja te su korisni kod analize zahtjeva u smislu analize cijelog ciklusa obrade unutar sustava. Drugačiji ponašajni modeli su oni vođeni događajima koji pokazuju kako sustav reagira na unutarnje ili vanjske poticaje. Dobar primjer za to su sustavi za rad u stvarnom vremenu koji imaju minimalno obrade podataka te se temelje na pretpostavci konačnog broja stanja koja se izmjenjuju a u kojem se sustav može naći u bilo kojem trenutku.

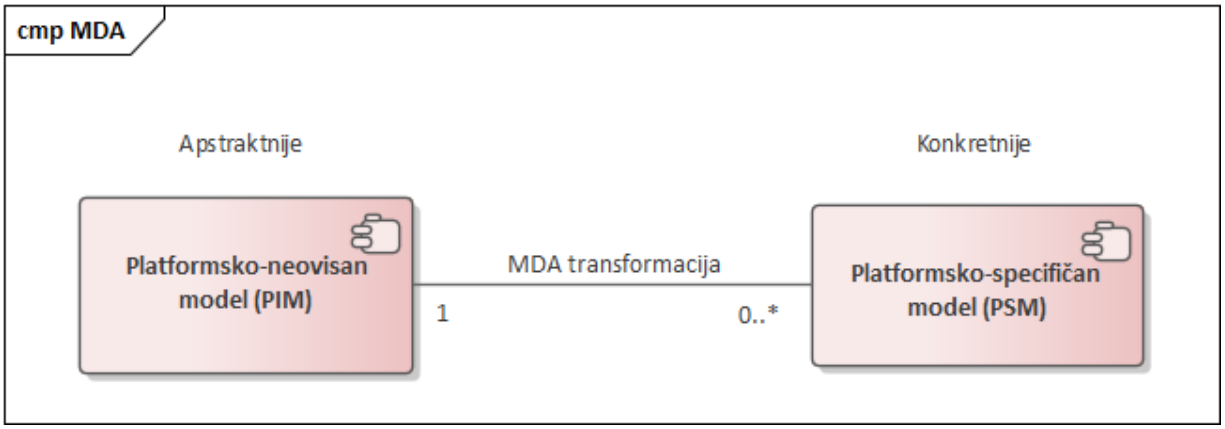
Slično tome, Tockey u svojem radu [11] govori o pojmu tzv. **semantičkog modela** (engl. *semantic model*) kao pojma oko kojega se veže ideja potpune automatizacije implementacije od najviše prema nižim razinama detalja. Pri tome razlikuje semantički model na razini poslovnih pravila od kojeg se kreće do semantičkog modela na razini tehnologije između kojih se primjenjuje tehnologija automatizacije, tj. preslikavanja i prevođenja od apstraktnog modela do implementacije programskim kodom.

Unificirani proces (engl. *UP - Unified Process*) kao metodologija razvoja programske potpore se temelji na iterativnom i inkrementalnom razvoju, uporabi slučajeva uporabe i usmjerenju na arhitekturu sustava. Metodologija opisuje generičke procese razvoja i predstavlja radni okvir koji je proširiv i podesiv prema konkretnoj razvojnoj okolini. Najpoznatija izvedba je **RUP** (*Rational Unified Process*), razvijena od tvrtke IBM. Uz jezgrene aktivnosti toga procesa: zahtjeva, analize, oblikovanja, implementacije i ispitivanja, vežu se pojedini modeli: model slučajeva uporabe, model analize, model oblikovanja, model razmještaja, model implementacije i model ispitivanja. Modeli su opisani uporabom UML-dijagrama koji se elaboriraju i u ovom priručniku i koji predstavljaju integralni dio unificiranog procesa.

U pogledu modelno usmjerene arhitekture postoje tri razine modela kao opisa sustava određena stupnjem implementacije sustava. To su:

- računalno-neovisan model (engl. *Computational Independent Model - CIM*) koji predstavlja domenski model ili apstrakciju domene sustava gdje nema pretpostavki automatizacije prevođenja u modele niže razine te predstavlja konceptualni pogled na sustav,
- platformski neovisan model (engl. *Platform Independent Model - PIM*) koji modelira rad sustava neovisno o platformi implementacije i u pravilu prikazuje statičku strukturu sustava s opisom načina reakcije na unutarnje ili vanjske poticaje, npr. na algoritamskoj razini,
- platformsko specifičan model (engl. *Platform Specific Model - PSM*) koji cilja specifičnu tehnološku okolinu gdje može postojati više slojeva opisa platformsko-specifičnih detalja.

Jedan platformski-neovisan model (PIM) moguće je prevesti u više odvojenih platformsko-specifičnih modela (PSM) gdje modelno usmjerena arhitektura (MDA) definira skup transformacija koje preslikavaju platformsko-neovisne modele u platformsko-specifične modele, Slika 3.2.1.



Slika 3.2.1 Uloga modelno usmjerene arhitekture (prema [17])

Cilj modelno usmjerene arhitekture je prema tome osigurati prenosivost između apstraktne arhitekture sustava i konkretne platformske arhitekture. Uporaba modela općenito ima svrhu pomagala za raspravu o postojećim i predloženim sustavima, i detaljnog opisa sustava kao polazišta za automatsku implementaciju sustava. U pogledu točnosti i potpunosti model koji se upotrebljava kao pomagalo za raspravu o sustavu u pravilu nije do kraja potpun i ne mora nužno biti točan u svim detaljima. Model kao oblik dokumentacije sustava treba biti točan, ali ne nužno i potpun, dok model koji se koristi kao temelj automatskog generiranja sustava treba biti i točan i potpun.

3.3. Standardi i alati za modeliranje

OMG (*Object Management Group*), udruženje je ustanova za razvoj standarda integracije velikih sustava te je odgovorno za definiranje jezika UML. Udruženje pokriva niz tehnologija i stoji iza niza manje ili više poznatih ISO standarda.

Osim jezika UML, u pogledu modelno usmjerenog inženjerstva dobro je spomenuti i druge standarde udruženja OMG:

- BPMN (*Business Process Modeling Notation*) – grafička notacija za modeliranje poslovnih procesa s elementima dijagrama toka, slično UML-dijagramima aktivnosti. Za razliku od standarda UML, za koji bi se moglo reći da je objektno-orijentiran, za standard BPMN bi se moglo reći da je procesno orijentiran i da je razvijen s ciljem olakšanog razumijevanja toka i procesa poslovnih aktivnosti. Pri tome definira vlastitu notaciju elemenata i simbola.
- SysML – standard koji se bavi inženjerstvom sustava te se veže uz pojam modelno usmjerenog inženjerstva sustava (MBSE). Standard se time smatra jezikom modeliranja arhitekture sustava gdje se u odnosu na UML i modelno usmjereni razvoj programske potpore modeliranje razreda i objekata zamjenjuje s pojmom modeliranja blokova. Pojam **bloka** pri tome pokriva čitav niz različitih elementa: programsku potporu, sklopovlje, podatke, procese. Po pitanju vrsta dijagrama koje nudi ovaj jezik, on se djelomično poklapa s jezikom UML, uz prošireni niz dijagrama gdje je konačan skup dijagrama nešto manji u odnosu na jezik UML.

CASE (engl. *Software Aided Software Engineering*) alati su programski proizvodi koji podupiru proces programskog inženjerstva, a posebice aktivnosti specifikacije, oblikovanja (modeliranja), implementacije i evolucije programske potpore. Ovisno o namjeni mogu se koristiti u svakoj fazi životnog ciklusa programskog proizvoda i cilj im je povećanje kvalitete konačnog proizvoda kroz normiranje načina prikaza i dijeljenja informacija uz smanjenje vremena i napora potrebnog za izradu programske potpore. Način na koji se to nastoji postići je omogućavanje automatizacije dijela procesnih aktivnosti te povećanje ponovne iskoristivosti (engl. *reusability*) modela i komponenti.

U web prostoru postoje mnogi alati programske potpore koji su namijenjeni modeliranju UML-om. Neki od njih, kao SparxSystems Enterprise Architect ili oni iz skupine IBM Rational alata su komercijalni alati koji šire pokrivaju poslovne procese oblikovanja sustava, a neki su dostupniji i pretežno predstavljaju okolinu na izradu UML-dijagrama ili su dio uspostavljenih razvojnih okolina.

Dobar dio alata je dostupan besplatno, pod otvorenim licencama ili imaju posebna nekomercijalna izdanja, a neki od njih su alati otvorenog koda (engl. *open-source*). Primjer dostupnih alata studentima za preuzimanje su ArgoUML, Astah, PlantUML, StarUML, UML Designer i Visual Paradigm, dok su neki dostupni unutar web preglednika kao Creately i LucidChart.

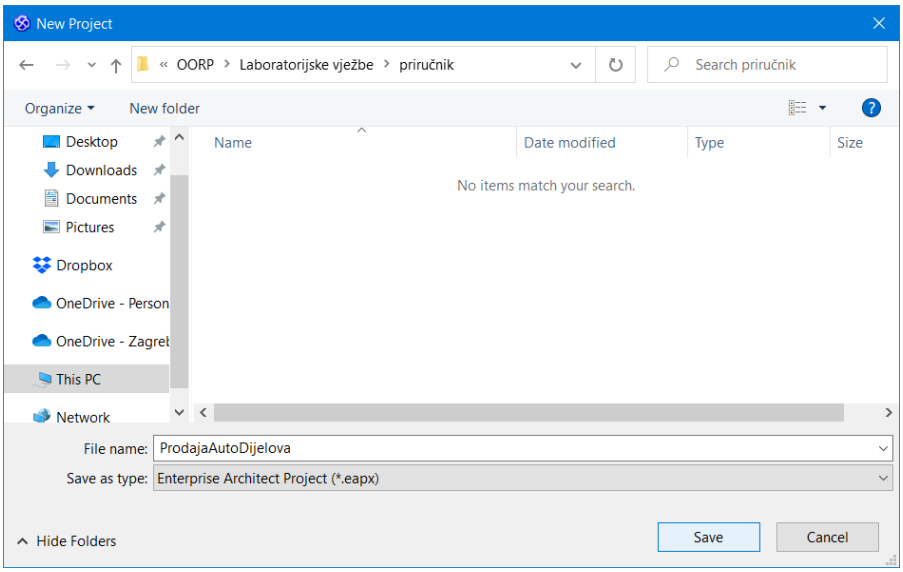
Pojedini alati nude podršku za više različitih standarda ili notacija modeliranja, prvenstveno oni komercijalni s podrškom za modelno usmjerenu arhitekturu. Osim standarda povezanih s modelno usmjerenim pristupom dobar dio alata predstavlja grafičko okruženje za izradu raznovrsnih grafičkih reprezentacija ili dijagrama. Primjerice, Microsoft Visio je aplikacija koja pokriva široki spektar grafičke notacije, s dodatkom za modeliranje UML-om.

4. Uporaba alata *Enterprise Architect*

Enterprise Architect kao alat, osim namjene modeliranja UML-om, nudi bogati niz mogućnosti: usmjerenost inženjerstvu sustava i procesa, potporu za više važećih standarda i notacija izrade modela i dijagrama, inženjerstvo koda za veći broj programskih jezika, integraciju s pojedinim razvojnim okruženjima. Konkretno, *Enterprise Architect* podržava 13 vrsta UML-dijagrama, uz niz drugih standarda, uključujući SysML i BPMN. Tehnologijom MDG (*Model Driven Generation*) proširuje resurse modeliranja prema specifičnim domenama i notacijama.
U pogledu pojma modela vezano uz uporabu notacije UML, referira se na semantiku: modela slučajeva uporabe i međudjelovanja, te dinamičkog, logičkog, komponentnog i fizičkog modela. U pogledu modelno usmjerene arhitekture nudi mogućnost transformacije modela za nekoliko ciljnih tehnologija.
U nastavku poglavlja će se na primjerima rađenim za elaborirani projekt programske potpore prikazati pojedine funkcionalnosti sučelja alata, od stvaranja projekta preko karakteristika notacije za izdvojene vrste UML-dijagrama do inženjerstva programskog koda. Ovisno o korištenom izdanju alata izgled pojedinih elemenata sučelja može se razlikovati.
Za svaku vrstu dijagrama komentiran je sadržaj alatnih traka. U pravilu su izdvojene po dvije alatne trake karakteristične za pojedinu vrstu dijagrama te je usto prikazana i treća alatna traka koja sadrži općenito elemente dijagrama koji su zajednički za svake vrste dijagrama. Ovisno o vrsti dijagrama pojedine stavke u toj alatnoj traci mogu imati manju ili veću važnost.

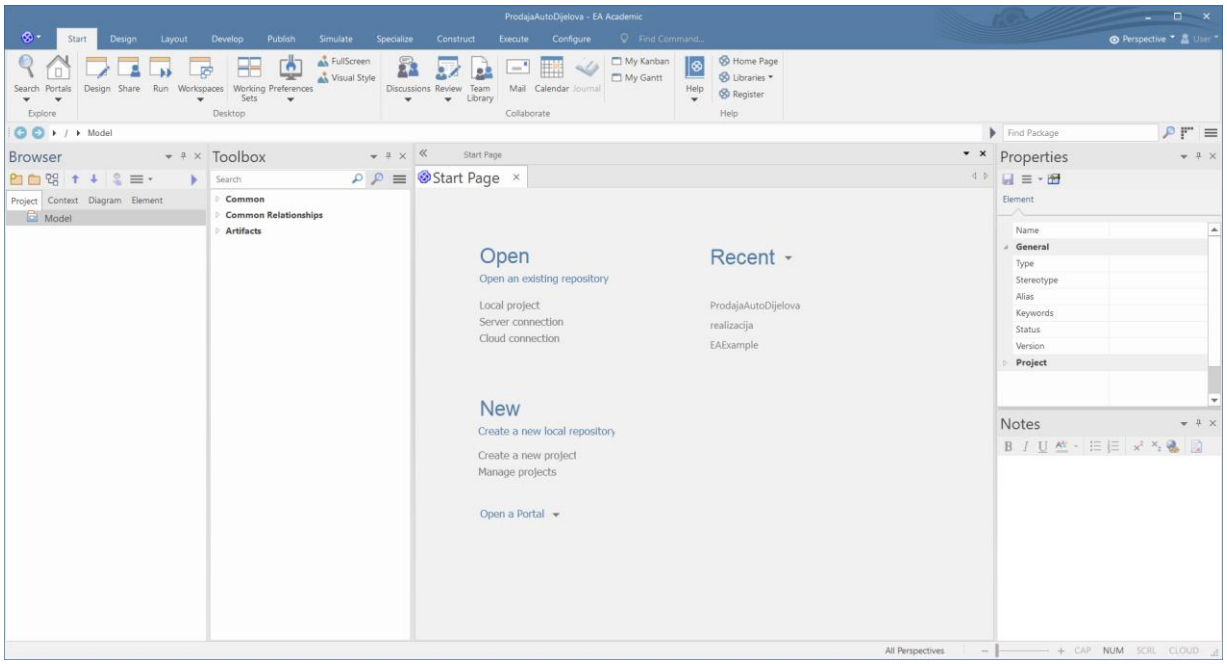
4.1. Stvaranje novog projekta

Prilikom stvaranja novog projekta u pravilu se otvara odgovarajući dijalog (Slika 4.1.1) gdje se zadaje naziv datoteke projekta.



Slika 4.1.1 Otvaranje novog projekta alatom Enterprise Architect

Početni projektni prostor se hijerarhijski uređuje uporabom projektnog preglednika (*Browser*) (Slika 4.1.2, na lijevoj strani slike).

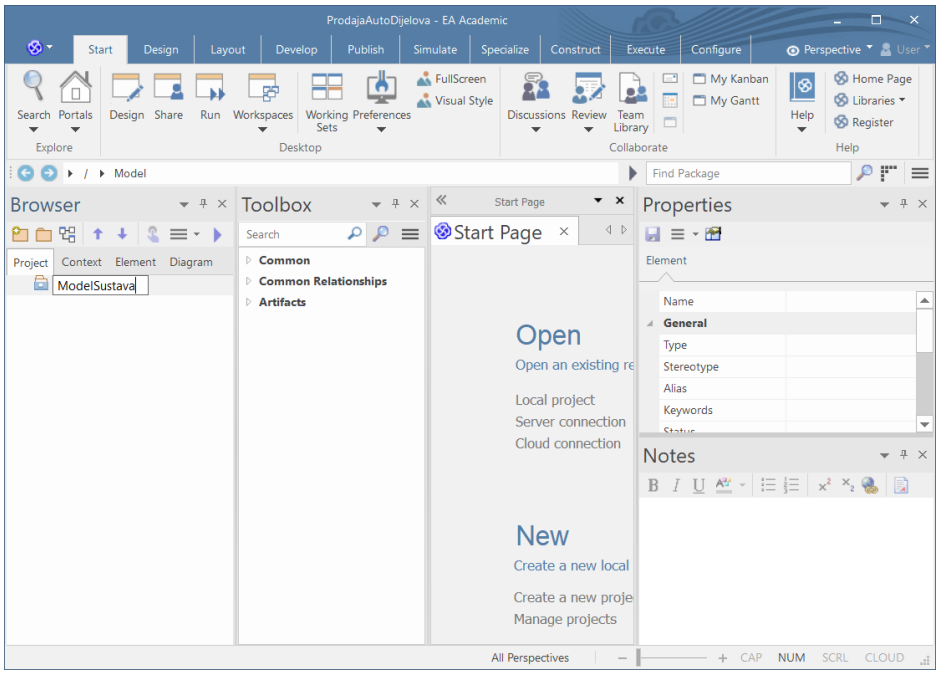


Slika 4.1.2 Početni izgled sučelja

Podrazumijevano se stvara korijenski element (naziv 'Model') koji se može iskoristiti za pohranu različitih elemenata projekta u željenoj hijerarhijskoj organizaciji.

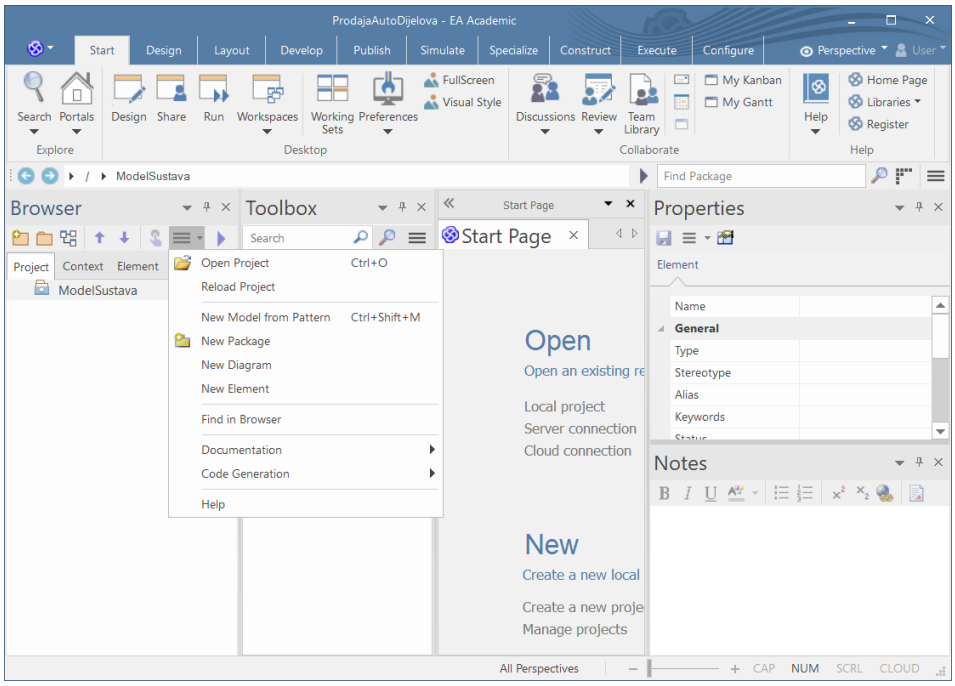
Hijerarhijska organizacija sadržaja

Unutar okvira projektnog preglednika može se iskoristiti postojeći ili stvoriti novi korijenski element s proizvoljnim nazivom (Slika 4.1.3).



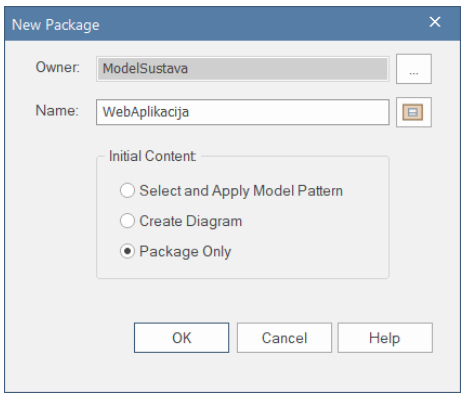
Slika 4.1.3 Postavljanje naziva modela u projektnom pregledniku

Unutar definiranog korijenskog elementa moguće je dodavati nove modele (*Model*), pakete (*Package*), dijagrame (*Diagram*) i različite alatne i elemente dijagrama (*Element*) te je moguće uspostaviti vlastitu strukturu i hijerarhiju sadržaja unutar projekta. Osim toga, pomoćnik *Model Wizard* već nudi pretpostavljene uzorke ili predloške projektne strukture. Navedene funkcionalnosti se ostvaruju alatnom trakom koja se nalazi pri vrhu okvira projektnog preglednika (Slika 4.1.4).



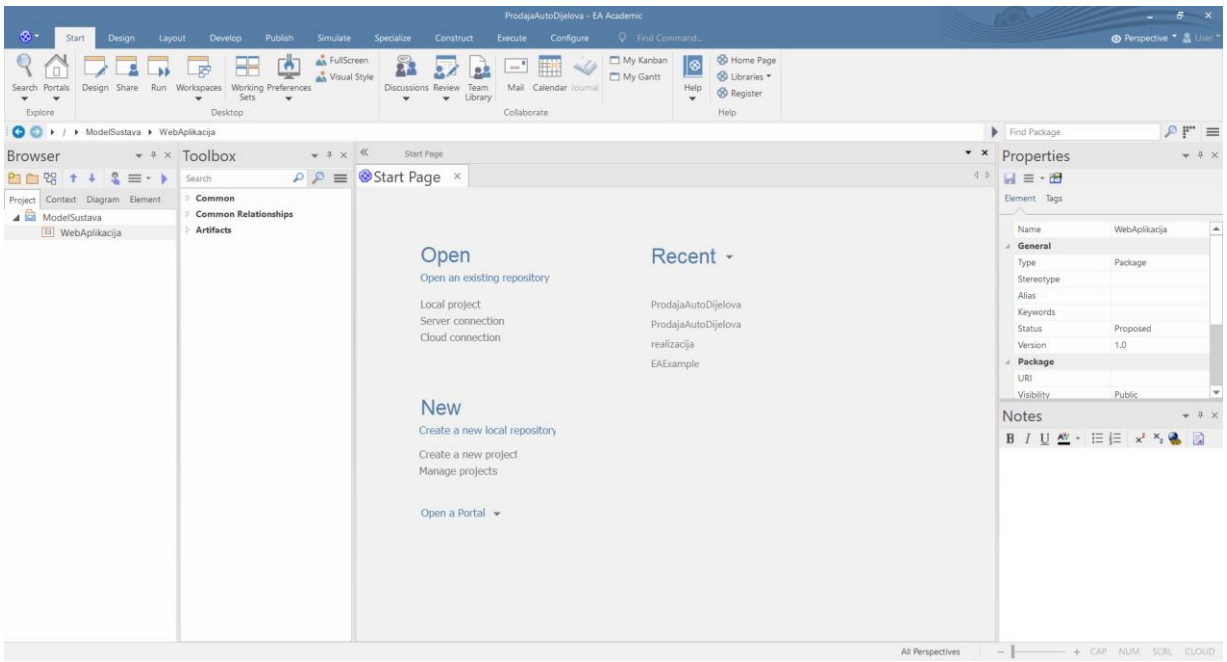
Slika 4.1.4 Upravljanje elementima hijerarhije modela projekta uporabom alatne trake

Neki osnovni pristup pretpostavlja upotrebu paketa unutar kojih se sprema sadržaj projekta. Pri definiciji novog paketa definira se prostor unutar kojeg će se dodavati novi paketi i različite vrste UML dijagrama (Slika 4.1.5). Opcija *Package Only* podrazumijeva stvaranje praznog paketa, a može se odmah zadati stvaranje dijagrama (opcija *Create Diagram*) ili primijeniti neki od ponuđenih predložaka ili uzoraka modela (opcija *Select and Apply Model Pattern*).



Slika 4.1.5 Definicija novog paketa

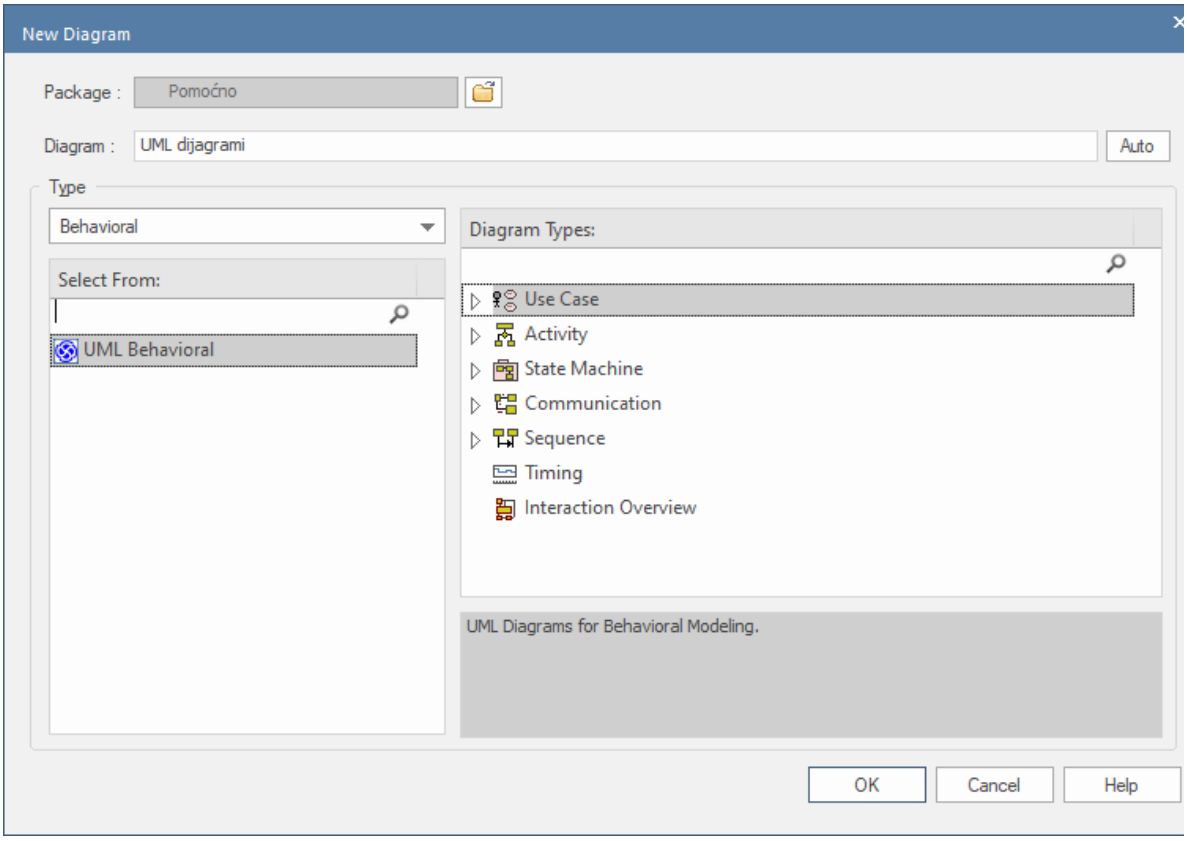
Naposljetku, Slika 4.1.6 prikazuje početni izgled sučelja.



*Slika 4.1.6 **Početno sučelje alata** Enterprise Architect s otvorenim novim paketom*

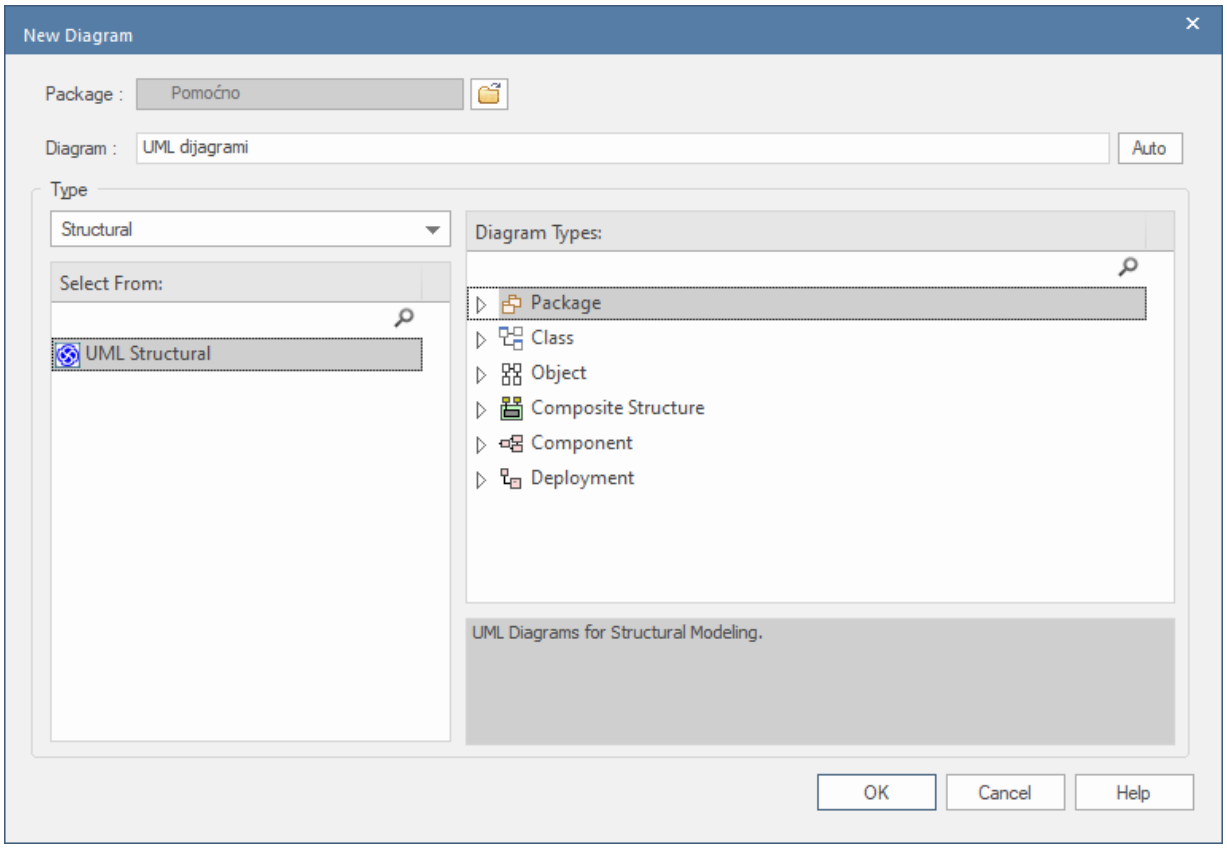
Umetanje novog dijagrama

Unutar alata dostupni su za modeliranje svi UML dijagrami predstavljeni u poglavlju 2: ponašajni dijagrami, Slika 4.1.7, i strukturni dijagrami, Slika 4.1.8.



Slika 4.1.7 Izbor ponašajnih vrsta UML dijagrama unutar alata Enterprise Architect.

Za svaku vrstu dijagrama su u dodatku prikazani izbor elemenata odnosno sadržaji alatnih okvira po pojedinim vrstama dijagrama kako ih nudi alat *Enterprise Architect* što bi se u pravilu trebalo poklapati s onim što nude i ostali alati za UML modeliranje.



Slika 4.1.8 Izbor strukturnih vrsta UML dijagrama unutar alata Enterprise Architect.

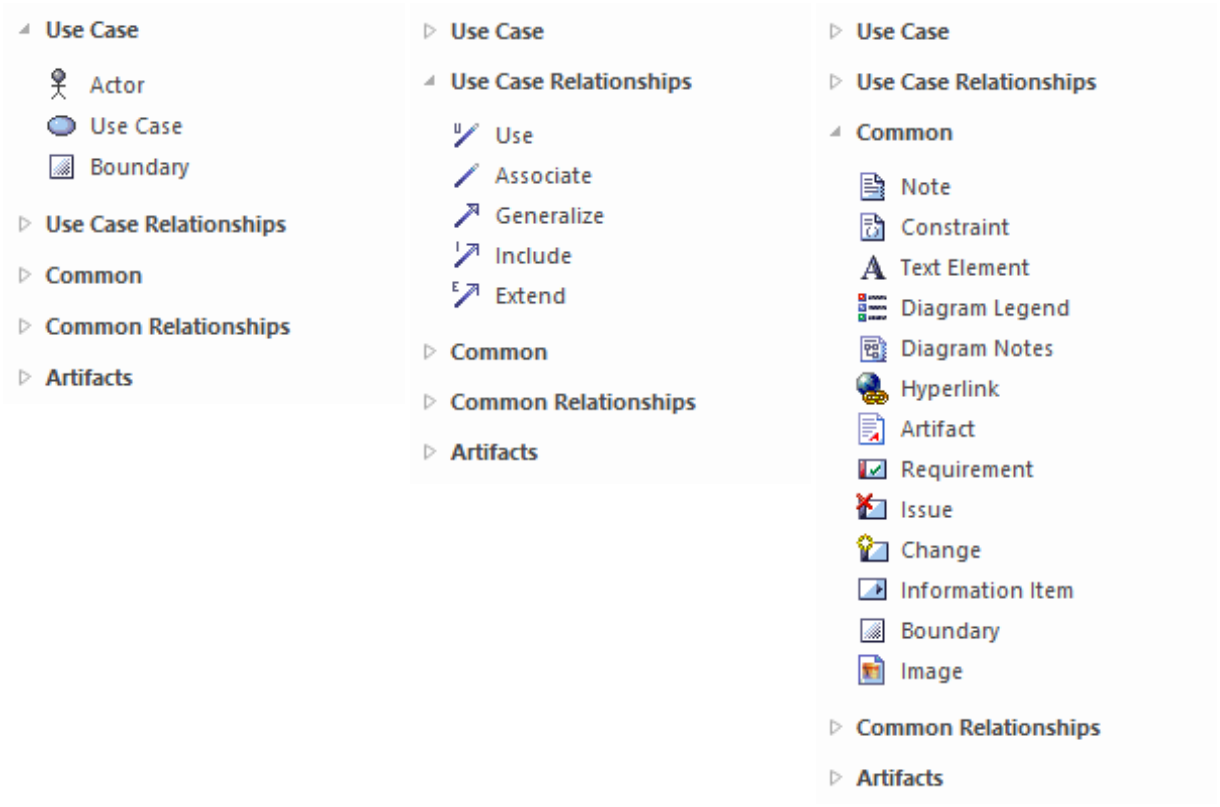
U sljedećem poglavlju se ukratko prikazuju alatni okviri uz specifične primjere izrade po vrstama dijagrama koji su predstavljeni u tekstu ovog priručnika.

4.2. Alatni okviri i primjeri uporabe

U ovom poglavlju prikazat će pregled alatnih okvira i funkcionalnosti po svim vrstama UML-dijagrama koji su uključeni u modeliranje i dokumentiranje u primjeru ovog teksta.

Dijagram slučajeva uporabe

Unutar alatne kutije (engl. *Toolbox*) pokazuju se alatne stavke vezane uz izradu dijagrama slučajeva uporabe. Slika 4.2.1 prikazuje izbor najčešće korištenih elemenata prikazanih unutar alatnih okvira *Enterprise Architecta* za dijagram slučajeva uporabe.



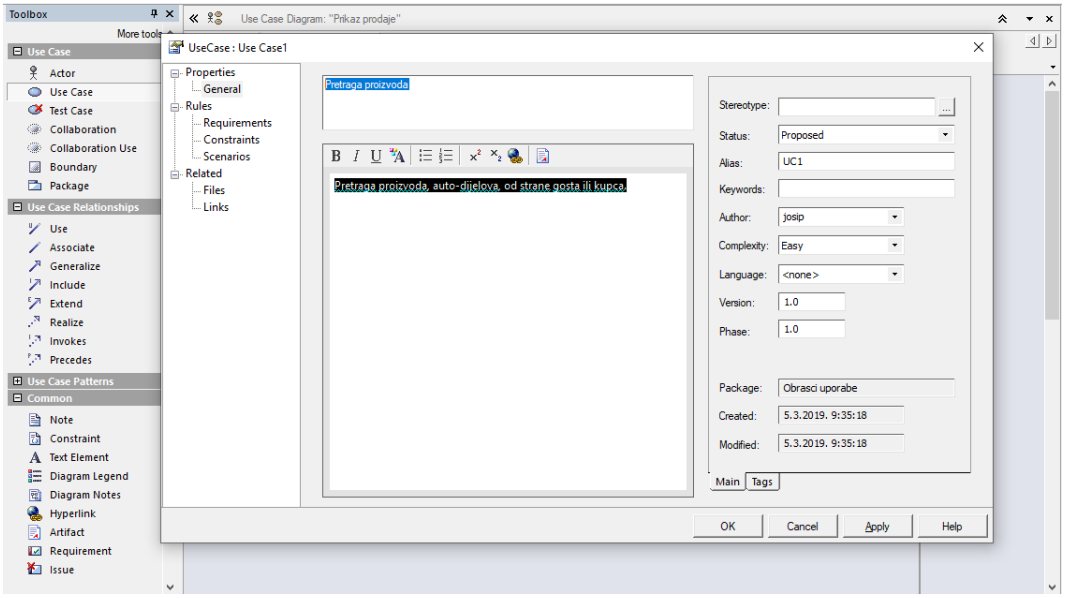
(a) *(b)* *(c)*

Slika 4.2.1 Alatni okviri modela slučajeva uporabe: (a) elementi, (b) odnosi i (c) opći dodaci.

Alatni okvir *Use Case* sadrži tri temeljna elementa dijagrama: aktora, slučaj uporabe i granicu sustava, dok alatni okvir *Use Case Relationships* sadrži četiri najčešće korištena odnosa dijagrama: pridruživanje, poopćenje, obuhvaćanje i proširenje. Od elemenata vrste *Common* dobro je spomenuti mogućnost pisanja bilješke (*Note*).

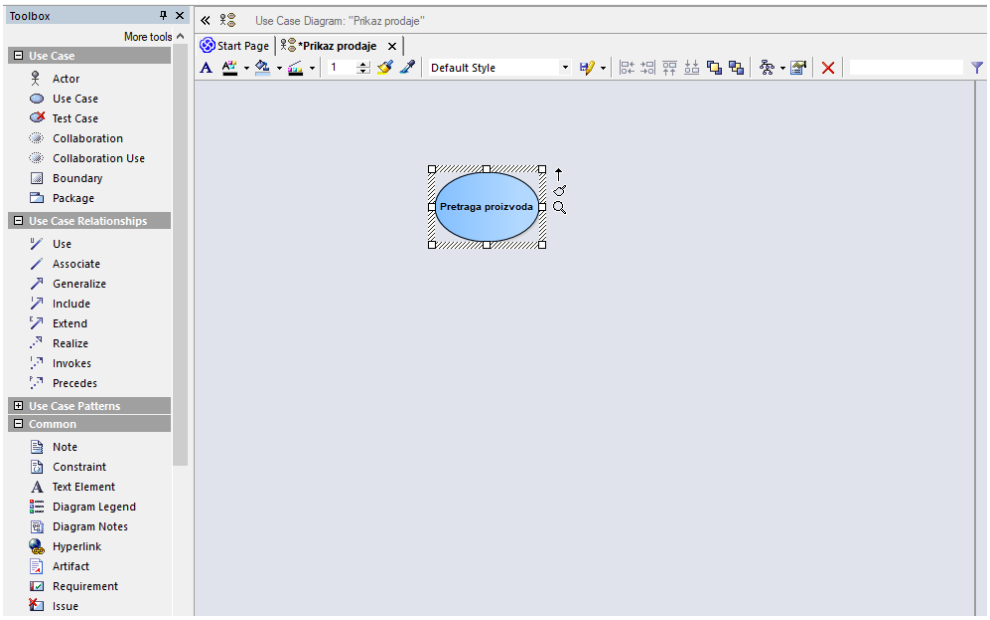
Umetanje elemenata

Slučajevi uporabe se definiraju i postavljaju u dijagram upotrebom alatnog okvira *Use Case* (Slika 4.2.1a) klikom na stavku *Use Case*. U dijaloškom okviru koji se otvara (Slika 4.2.2) potrebno je dati naziv i kratki opis novom slučaju uporabe. Usto, moguće je postaviti i mnoštvo drugih informacija, kao npr. upisati skraćeni naziv u polju *Alias*: 'UC1', kao početak pobrojavanja dijagrama radi lakšeg pozivanja na njih u kasnijem razvoju dokumentacije i programskog ostvarenja.



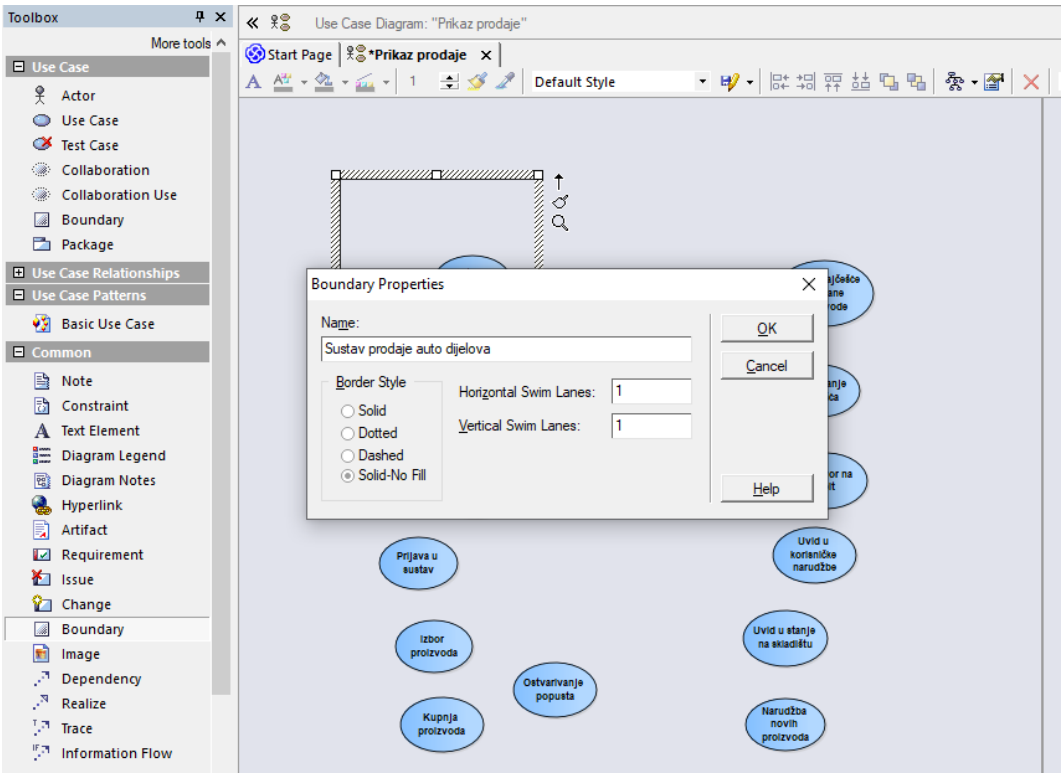
Slika 4.2.2 Definicija novog slučaja uporabe

Nakon potvrde dodavanja novog slučaja uporabe prikaz istog u ovalnom obliku se pojavljuje u dijagramu sa zadanim nazivom, Slika 4.2.3.



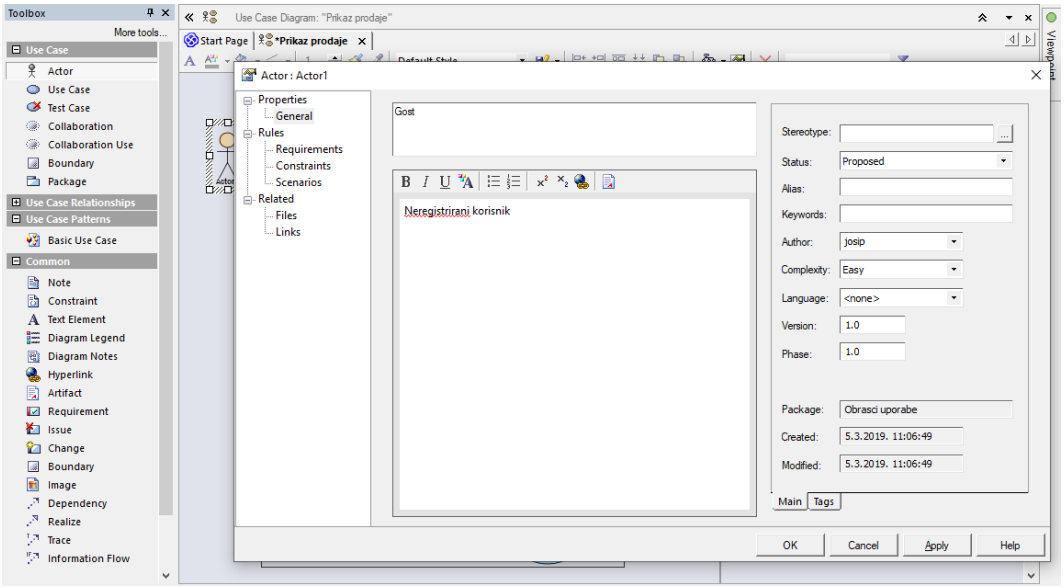
Slika 4.2.3 Izgled novog slučaja uporabe

Granica sustava se umeće izborom stavke *Boundary* iz alatnog okvira *Use Case*, Slika 4.2.4.



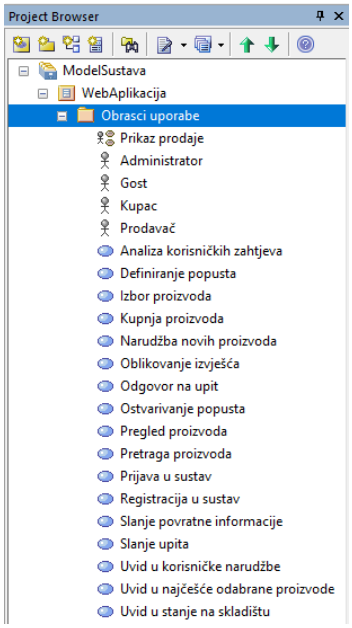
Slika 4.2.4 Definicija granice sustava modeliranog dijagramom slučajeva uporabe

Aktori se postavljaju u dijagram izborom stavke *Actor* iz alatnog okvira *Use Case* (Slika 4.2.1a) te im je potrebno definirati naziv i kratki opis (Slika 4.2.5).



Slika 4.2.5 Definicija novog aktora

U okviru projektnog preglednika moguće je u svakom trenutku pod stavkom odgovarajućeg paketa imati pregled elemenata: aktora i slučajeva uporabe koji čine pojedini dijagram, Slika 4.2.6.

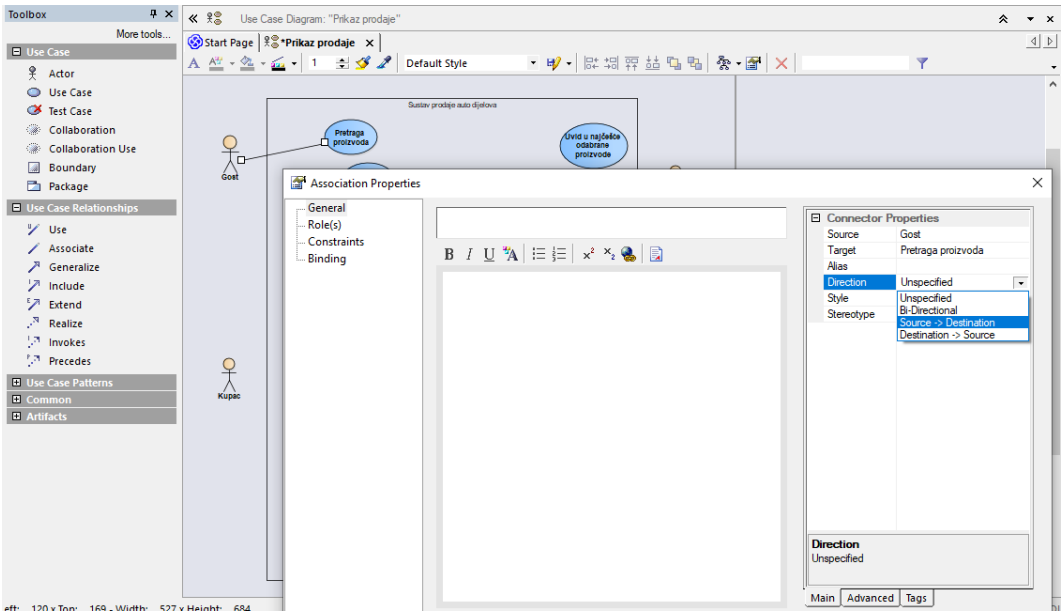


Slika 4.2.6 Projektni preglednik

Nakon umetanja elemenata, njihovi odnosi se definiraju umetanjem različitih vrsta veza.

Umetanje odnosa između elemenata

Veze pridruživanja se postavljaju u dijagram izborom stavke *Associate* iz alatnog okvira *Use Case Relationships* (Slika 4.2.1b) te je u sučelju dijagrama potrebno povući vezu između elemenata između kojih se želi postaviti veza pridruživanja. Potom se dvostrukim klikom ili izborom *Properties* iz izbornika može dodatno opisati postavljena veza pridruživanja. Otvara se dijaloški okvir te je moguće definirati naziv, kratki opis i usmjerenje veze. Slika 4.2.7 prikazuje definiciju veze pridruživanja između aktora 'Gost' (polje *Source* u desnom okviru) i slučaja uporabe 'Pretraga proizvoda' (polje *Target* u desnom okviru) s usmjerenjem od aktora prema slučaju uporabe (stavka *Source* → *Destination* u polju *Direction* u okviru *Connection Properties*). Time se želi reći da aktor 'Gost' koristi slučaj uporabe 'Pretraga proizvoda' unutar predviđenog sustava, tj. da će predviđeni programski proizvod korisniku gostu (koji nije registriran ni prijavljen u sustav) omogućiti pretragu proizvoda (auto-dijelova) koji će se ponuditi unutar sučelja aplikacije.

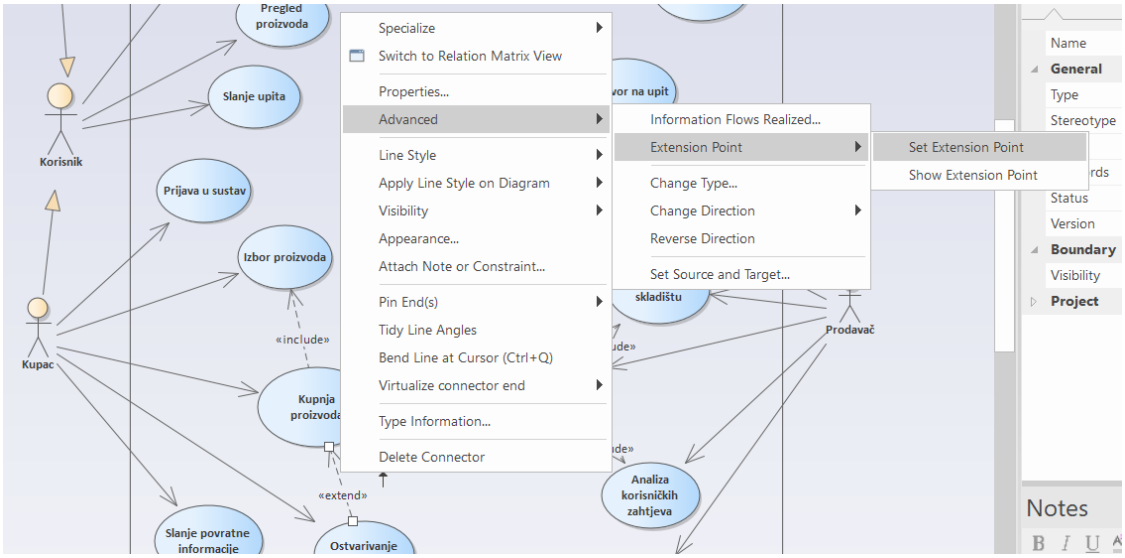


Slika 4.2.7 Definicija veze pridruživanja

Veza obuhvaćanja definira se unutar dijagrama izborom stavke *Include* iz alatnog okvira *Use Case Relationships* (Slika 4.2.1b) nakon čega se povlači u smjeru svog usmjerenja. Slika 2.3.8 i Slika 2.3.9 su primjeri prikaza veze obuhvaćanja.

Veza proširenja definira se unutar dijagrama izborom stavke *Extend* iz alatnog okvira *Use Case Relationships* (Slika 4.2.1b) nakon čega se povlači u smjeru od proširenog prema osnovnom slučaju uporabe.

Desnim klikom na liniju veze proširenja i izborom *Advanced → Extension Point → Set Extension Point* definira se točka proširenja, Slika 4.2.8.

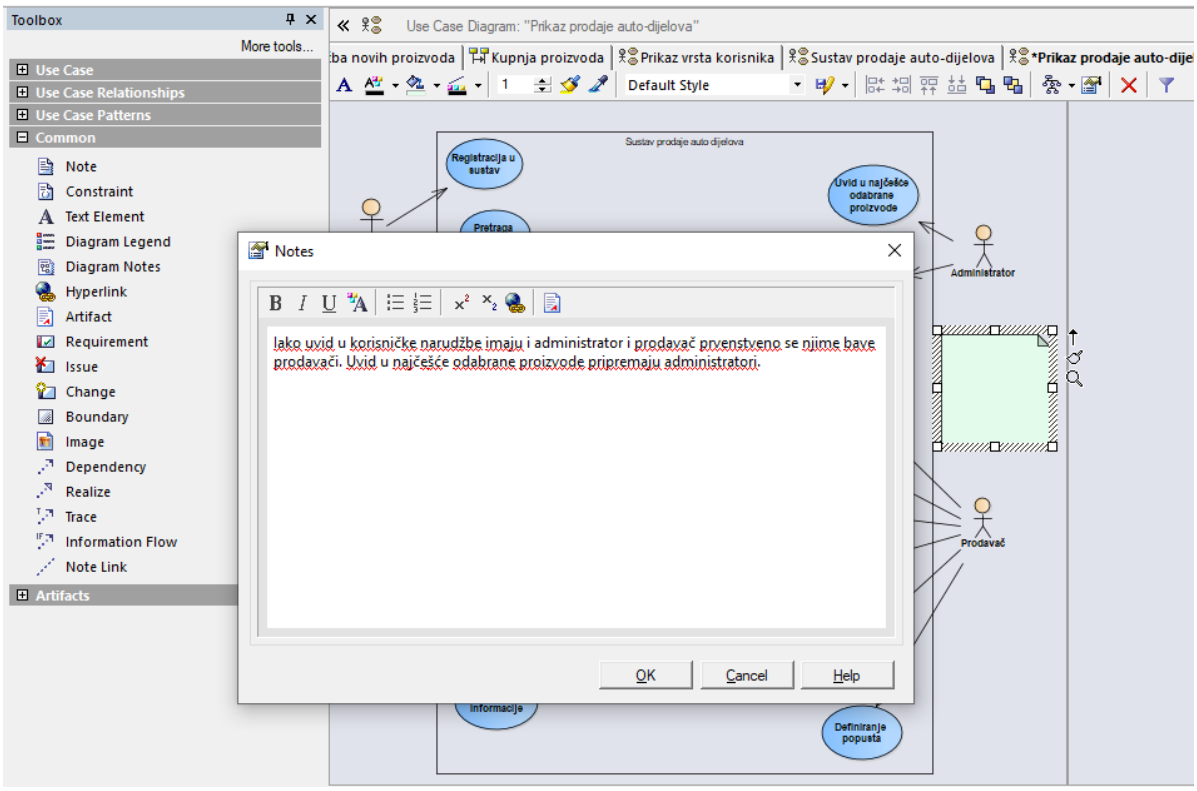


Slika 4.2.8 Definiranje točke proširenja za vezu proširenja na dijagramu slučajeva uporabe

Umetanje odnosa poopćenja se čini na sličan način povlačenjem veze poopćenja od specifičnijeg prema općenitijem elementu.
Jednom definirani aktor postaje zasebna i jedinstvena stavka u prikazu projektnog preglednika pa bi npr. definicija veza poopćenja između aktora koji se prikazuju na više dijagrama za posljedicu imala njihov prikaz i na drugim dijagramima gdje bi to dovelo do nepreglednosti zbog prikaza novih linija veza preko cijelog dijagrama. Zato je neželjeni prikaz veza moguće je „sakriti“ (ukloniti iz prikaza).
Primjer za to su dijagram hijerarhijskog prikaza vrsta aktora (Slika 2.3.5) i dijagram prikaza svih aktora s slučajevima uporabe (Slika 2.3.4). Na ovom drugom su veze poopćenja koje postoje između aktora skrivene jer nisu nužno potrebne s obzirom na postojanje prvospomenutog dijagrama i zbog preglednosti dijagrama.

Umetanje bilješke ili komentara

Bilješka se upisuje kao posebna vrsta elementa postavlja izborom stavke *Note* iz alatnog okvira *Common* (Slika 4.2.1c) gdje se tekst komentara stavlja u za to posebno otvoreni prozor, odnosno dijalog (Slika 4.2.9).

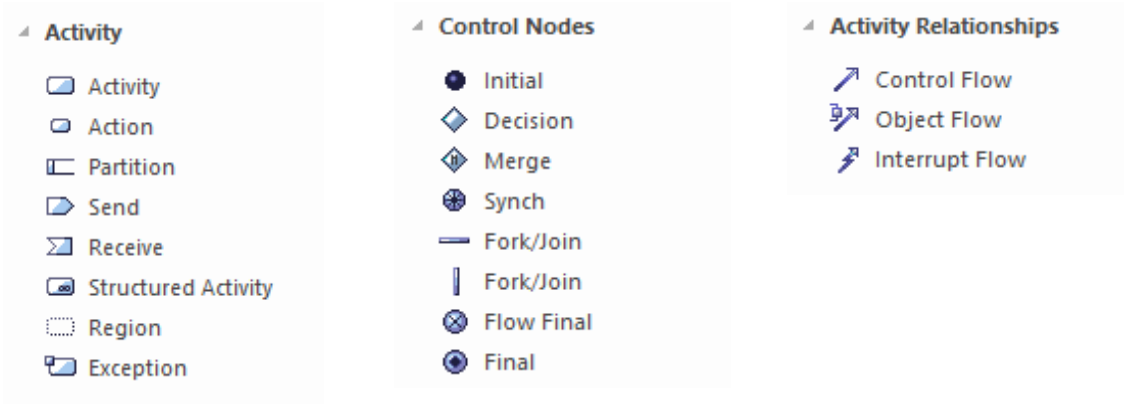


Slika 4.2.9 Umetanje bilješke na dijagram slučajeva uporabe

Slika 2.3.4 prikazuje rezultat umetanja komentara, a Slika 2.3.7 primjer umetanja dodatnog komentara.

Dijagram aktivnosti

Pri otvaranju novog dijagrama aktivnosti alatni okvir *Toolbox* zadobiva oblik sa stavkama za postavljanje elemenata i njihovih odnosa na dijagramu aktivnosti, Slika 4.2.10.



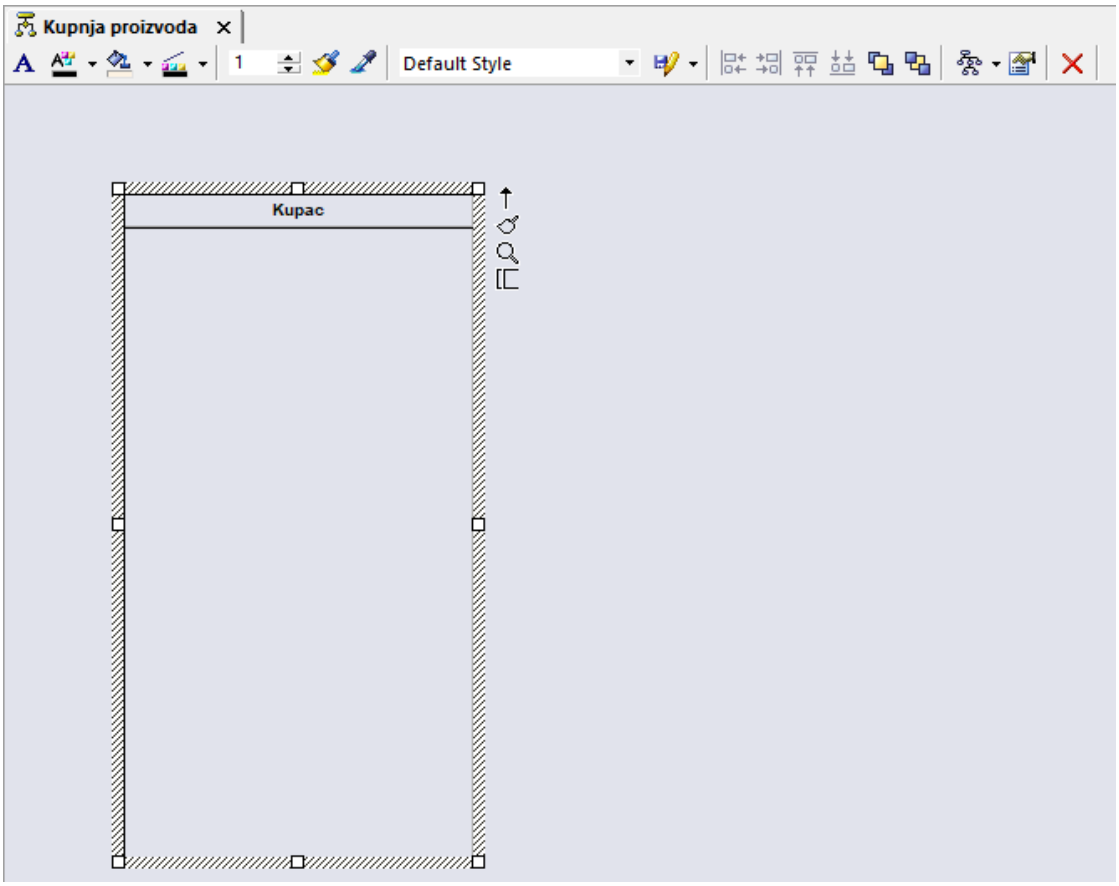
(a) *(b)* *(c)*

Slika 4.2.10 Alatni okviri dijagrama aktivnosti

Ponuđene vrste odnosa između aktivnosti na dijagramu predstavljaju **prijelaze** ili **tok** između akcija koji se mogu ali i ne moraju događati pod određenim uvjetima. Na dijagramu aktivnosti može se prikazati upravljački i podatkovni tok, ali ovdje ćemo se zadržati na ovom prvom, upravljačkom toku (engl. *control flow*) pa se govori o takvoj vrsti prijelaza između akcija.

Postavljanje particija

Nova particija se postavlja u dijagram izborom stavke *Partition* iz alatnog okvira *Activity* (Slika 4.2.10a) i postavljanjem odgovarajućeg naziva particije, Slika 4.2.11.

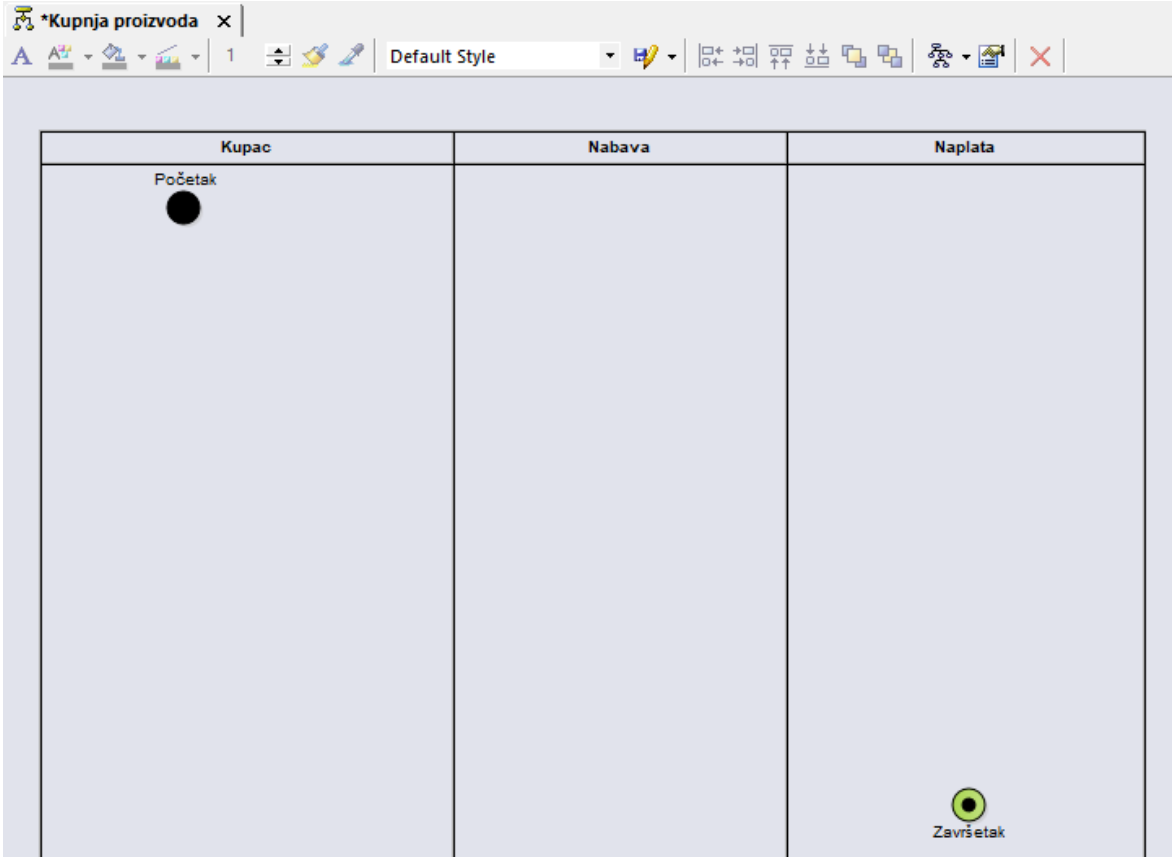


Slika 4.2.11 Postavljenja particije (plivačke staze) na dijagram aktivnosti

Particije se mogu položiti i horizontalno klikom na najdonju ikonicu u gornjem desnom kutu za particiju koja je u fokusu (vidi se za primjer particije na slici) te ih jednostavnim povlačenjem po zaslonu slagati jednu uz drugu.

Umetanje čvorova i prijelaza

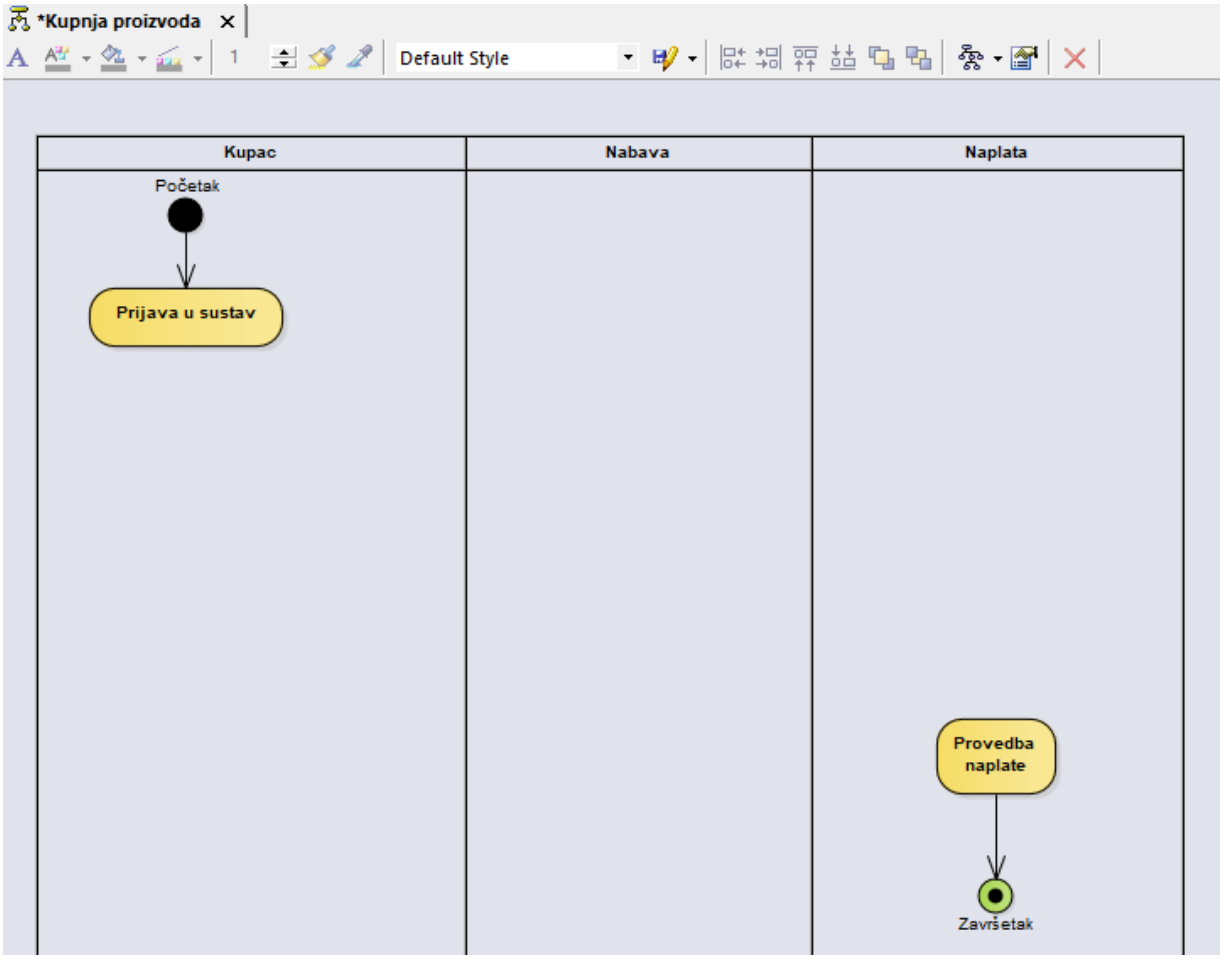
Dijagram aktivnosti započinje s početnim čvorom te završava s nekim od završnih čvorova, Slika 4.2.12. Ti čvorovi se na dijagramu dobivaju izborom stavki *Initial* i *Final* iz alatnog okvira *Control Nodes*, Slika 4.2.10b.



Slika 4.2.12 Postavljanje početnog i završnog čvora na dijagramu aktivnosti

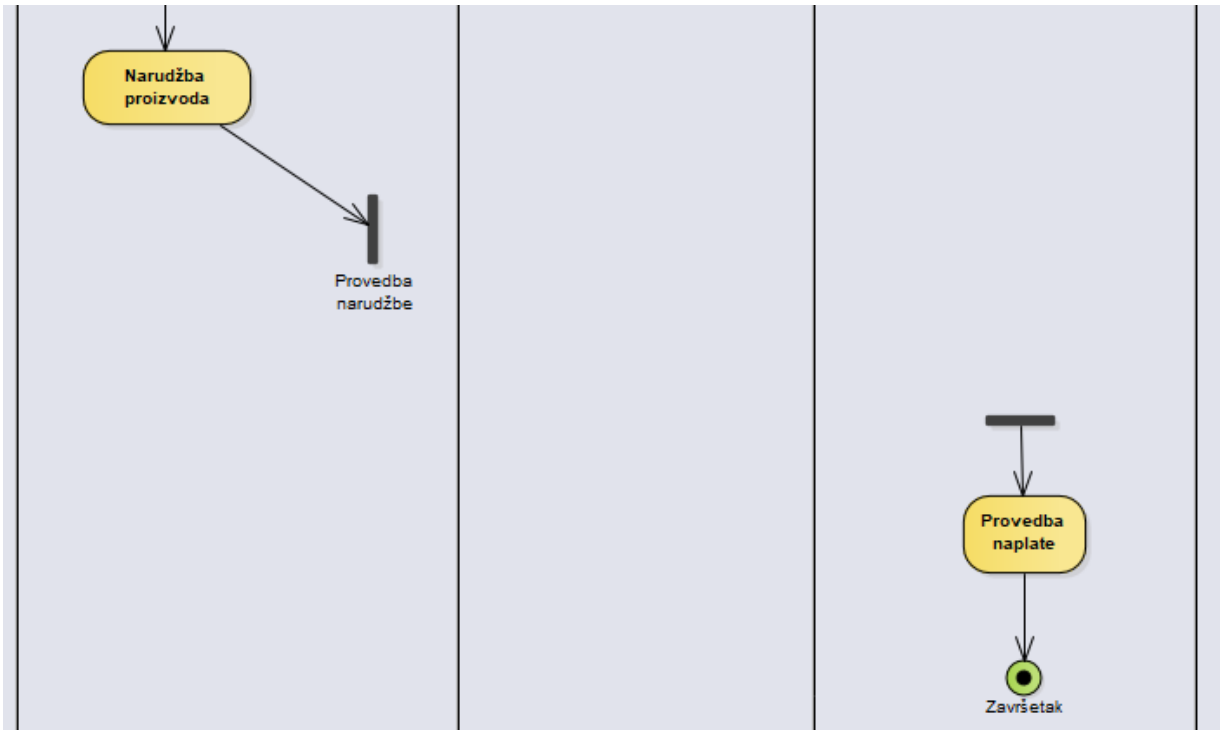
Akcije i aktivnosti koje opisuju aktivnosti pojedinih sudionika se u dijagram postavljaju izborom stavki *Action* i *Activity* iz alatnog okvira *Activity*. Predviđa se da normalan tijek kupnje započinje s aktivnosti 'Prijava u sustav' od strane kupca, a završava s aktivnosti 'Provedba naplate' od strane podsustava naplate, Slika 4.2.13.

Prijelazi koji stvaraju i označavaju upravljački tok dijagrama aktivnosti se u dijagram postavljaju izborom stavke *Control Flow* iz alatnog okvira *Activity Relationships*, Slika 4.2.10c.

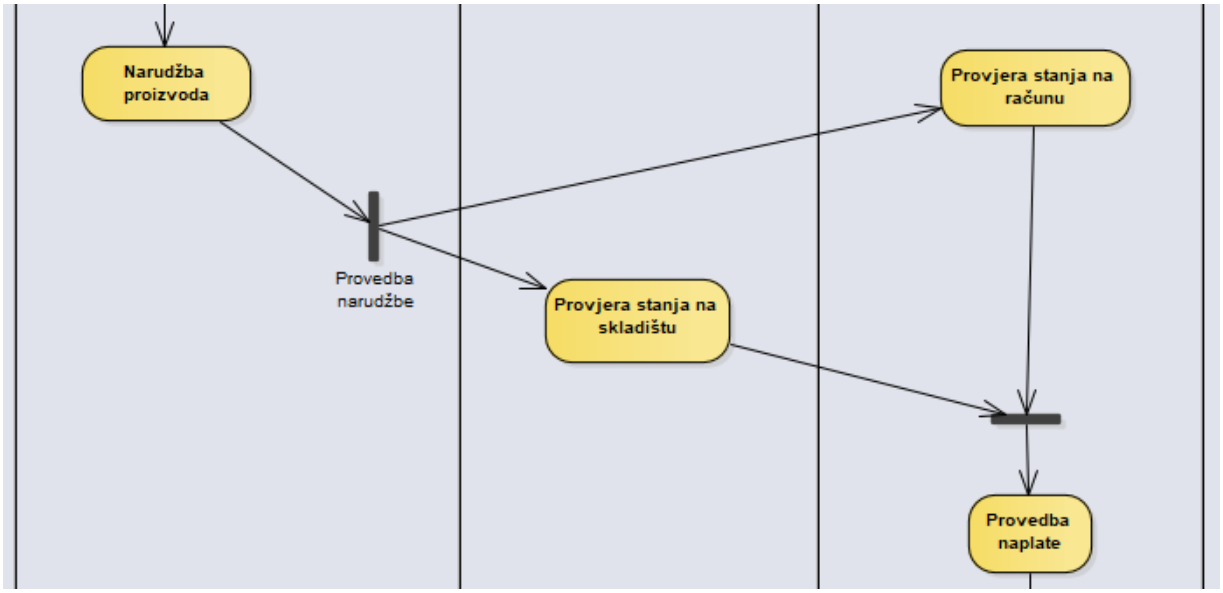


Slika 4.2.13 Postavljanje aktivnosti na dijagramu aktivnosti

Izborom para stavki *Fork/Join* iz alatnog okvira *Control Nodes* na dijagramu se prikazuje račvanje kada se upravljački tok račva na dvije usporedne linije izvođenja te skupljanje tih dviju usporednih linija upravljačkog toka kao potrebne točke sinkronizacije, Slika 4.2.14. U dvije usporedne linije izvođenja postavljaju se odgovarajuće akcije koje jedna na drugu čekaju u točki sinkronizacije, Slika 4.2.15.

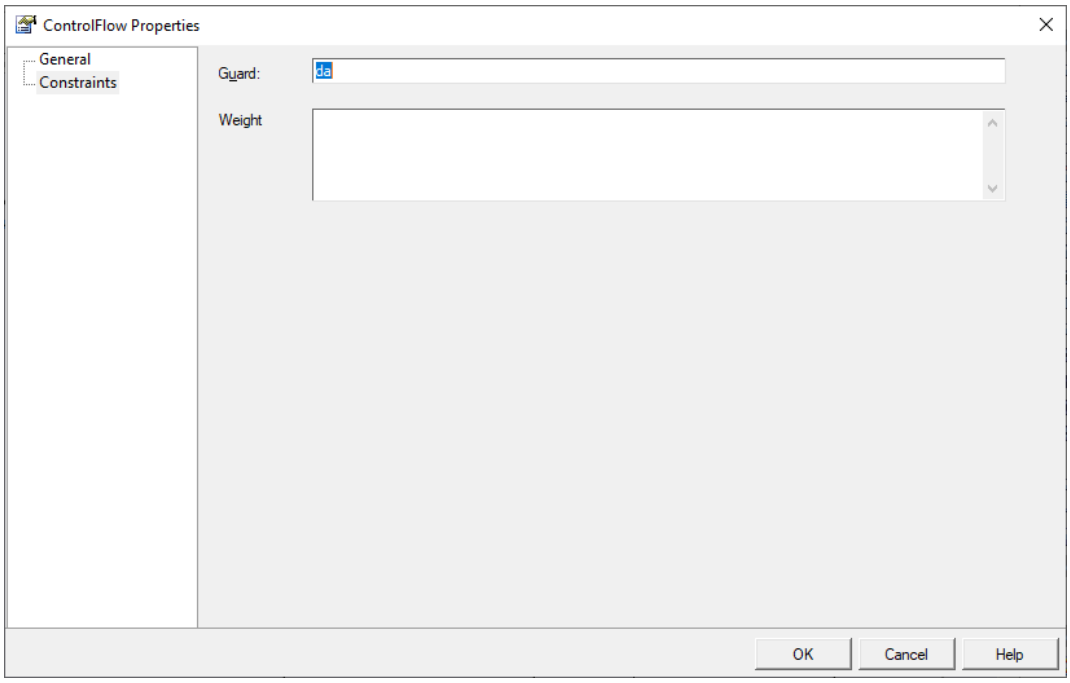


Slika 4.2.14 Postavljanje račvanja i skupljanja na dijagramu aktivnosti



Slika 4.2.15 Prikaz usporednog izvođenja na dijagramu aktivnosti

Grananje na dijagramu aktivnosti se ostvaruje izborom stavke *Decision* iz alatnog okvira *Control Nodes* uz navod ispitivanog uvjeta, čime se postavlja čvor odluke, te postavljanjem dviju ili više grana izlaska iz čvora odluke. Svakoj od grana potrebno je naznačiti pod kojim se uvjetom događa dvostrukim klikom na oznaku (liniju) prijelaza i ispunjavanjem stavke *Guard* unutar otvorenog prozora *ControlFlow Properties*, Slika 4.2.16.



Slika 4.2.16 Postavljanje uvjeta grananja na dijagramu aktivnosti

Upisuje se uvjet grananje bez navoda uglatih zagrada koje se u prikazu podrazumijevaju od strane alata.

Različite vrste završnih čvorova

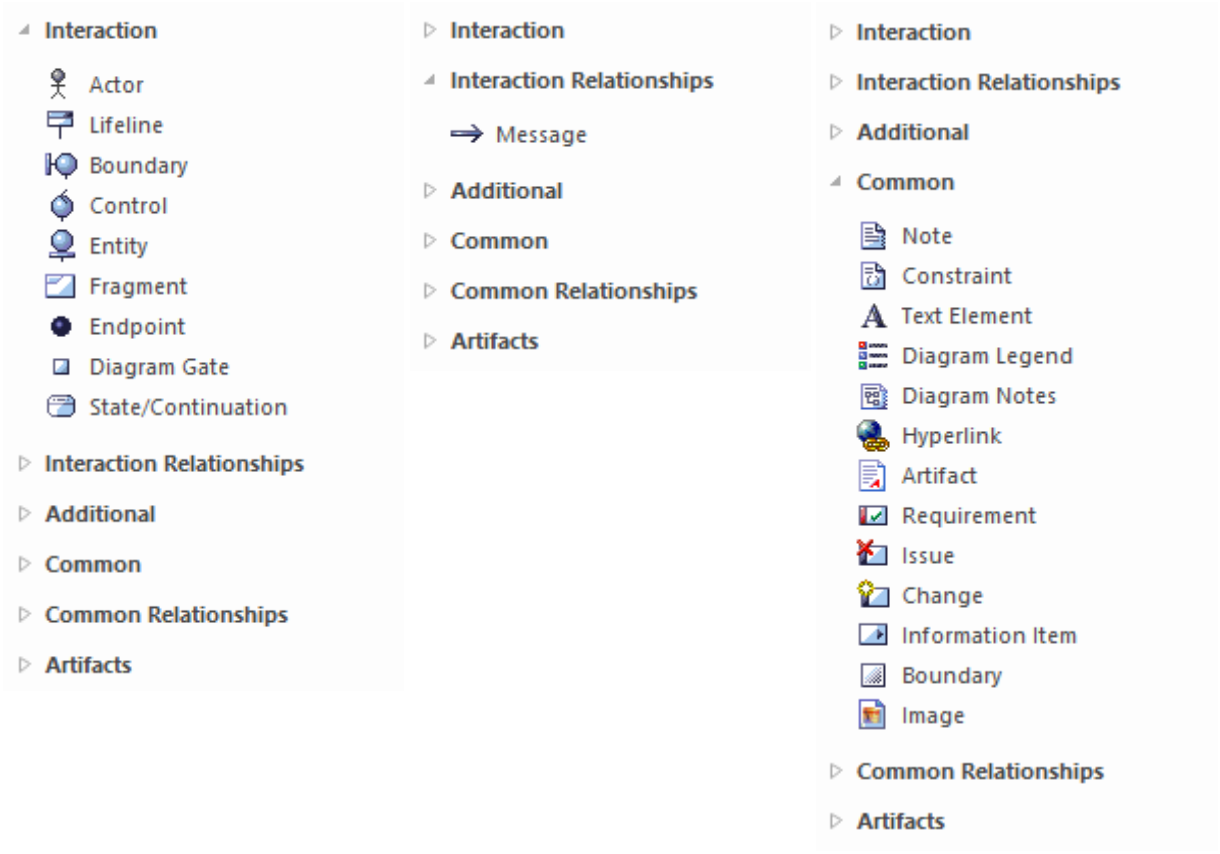
Razlikuju se notacije završetka upravljačkih tokova gdje se svaki pojedinačni ili izdvojeni upravljački tok zaključuje čvorom vrste *Flow Final* iz alatnog okvira *Control Nodes* što se u prikazu razlikuje od čvora *Final* koji se koristi za oznaku završetka cjelokupnog toka svih akcija i aktivnosti. Prva notacija bi se mogla upotrijebiti za prikaz završetka alternativnih tokova izvođenja akcija i aktivnosti na dijagramu, druga za neki redoviti ili očekivani tok akcija i aktivnosti.

Akcije vođene događajima

Posebni upravljački čvorovi nalaze se u alatnom okviru *Activity*: za slanje signala stavka *Send* i za prihvaćanja događaja stavka *Receive*. Vremenski signal se umeće izborom stavke *Receive* i dodatnim izborom na desni klik *Advanced → Accept Time Event*.

Dijagram slijeda

Nakon potvrde odabira i otvaranja prostora u sučelju za modeliranje dijagrama slijeda, dio sučelja s alatnim okvirima u svom izborniku prikazuje stavke koje se odnose na elemente dijagrama slijeda, Slika 4.2.17.



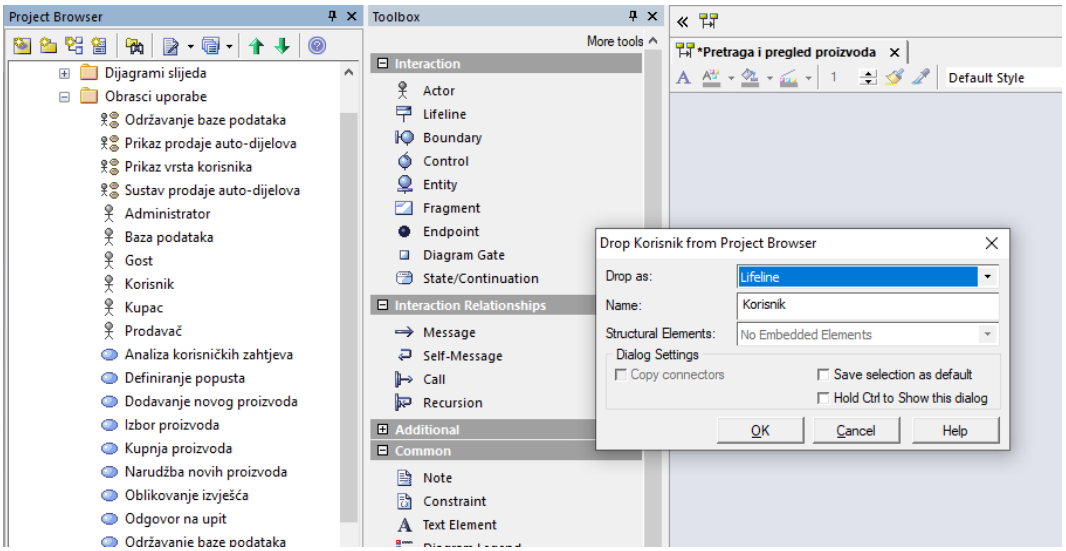
(a) *(b)* *(c)*

Slika 4.2.17 Alatni izbornici za izradu dijagrama slijeda

Alatni okvir pod naslovom *Interaction* nudi stavke za postavljanje temeljnih gradivnih komponenti (Slika 4.2.17a), pod naslovom *Interaction Relationships* nudi stavke za definiranje komunikacije između elemenata dijagrama (Slika 4.2.17b), a pod naslovom *Common* neke stavke opće vrste (Slika 4.2.17c).

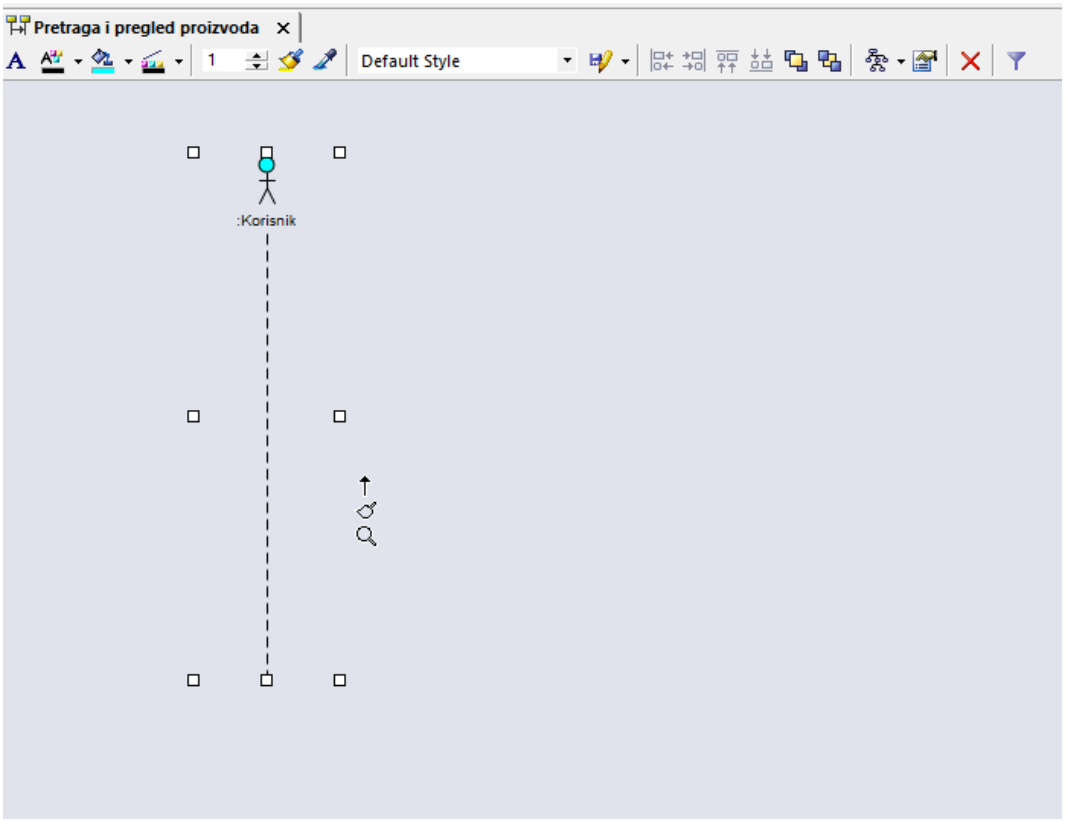
Umetanje aktora i objekata

Postavljanje postojećeg aktora na dijagram slijeda se svodi na izbor i povlačenje stavke koja predstavlja pojedinog aktora iz okvira projektnog preglednika u prostor predviđen za oblikovanje dijagrama slijeda odnosno otvoreni okvir dijagrama. Npr. povlačenjem aktora 'Korisnik' u sučelje dijagrama (Slika 4.2.18).



Slika 4.2.18 Postavljanje postojećeg aktora na dijagram slijeda

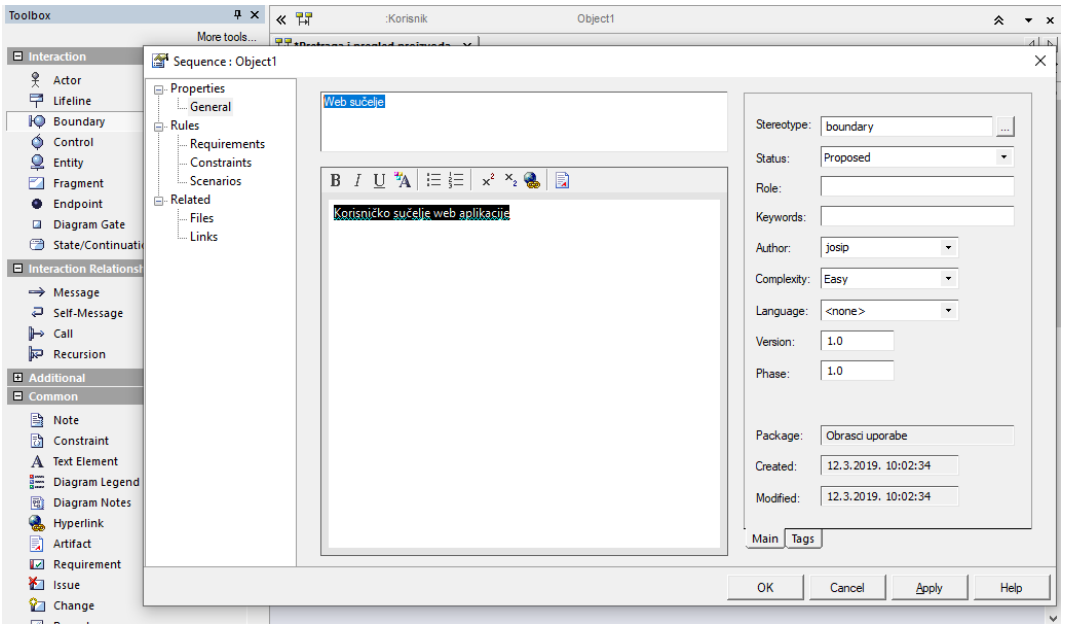
Izborom opcije *Lifeline* (pod stavkom *Drop as:*) na dijagramu se definira životna linija novog objekta ili pojedinca vrste 'Korisnik', Slika 4.2.19.



Slika 4.2.19 Nova životna linija postojećeg aktora u dijagramu slijeda

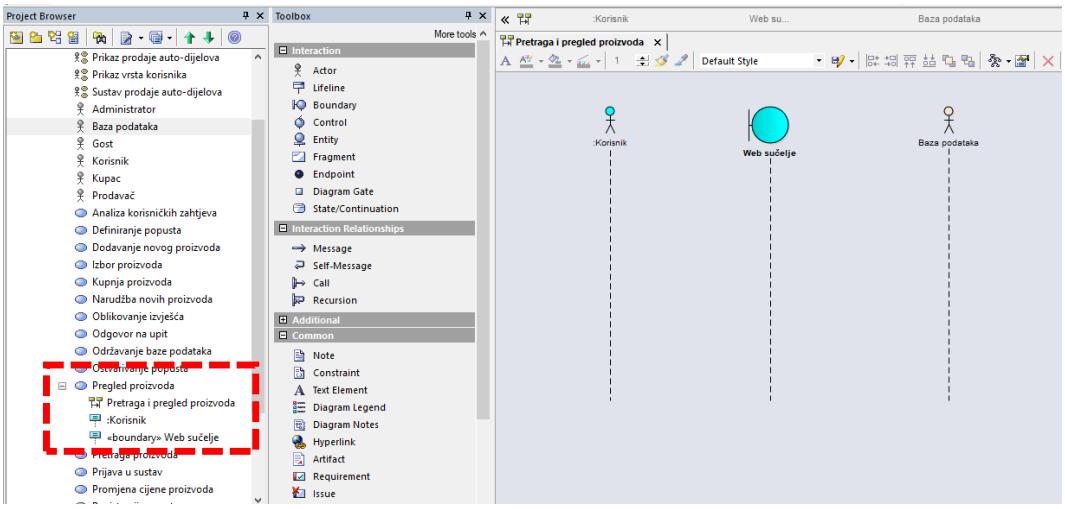
Isto tako, npr. umetanje aktora 'Baza podataka' koji je već definiran u nekom slučaju uporabe obavlja se povlačenjem stavke tog aktora iz okvira projektnog preglednika, ali se može izabrati opcija umetanja *Link* pod stavkom *Drop as:* iz razloga što je baza podataka u sustavu jedinstvena, tj. nema više objekata vrste 'Baza podataka' za razliku od korisnika sustava kojih može biti nebrojeno. Na taj način se može naglasiti ta razlika u naravi samih aktora. To znači

da je na dijagram slijeda postavljena samo poveznica na jedinstvenog aktora 'Baza podataka' dok je kod postavljanja aktora 'Korisnik' izabrana opcija *Lifeline* da bi se stvorio novi pojedinac aktora vrste 'Korisnik'.
Postavljanje novog objekta u dijagram, npr. stavke *Boundary* iz alatnog okvira *Interaction* (Slika 4.2.17a) zahtijeva unos naziva i opisa novonastalog objekta, Slika 4.2.20. Vrsta objekta *Boundary* bi predstavljala klijentski dio web aplikacije koji predstavlja sučelje prema korisniku. Isto tako može se u dijagram postaviti i objekt neke druge vrste ovisno o potrebi, npr. vrste *Control* za upravljački ili pozadinski dio aplikacije.



Slika 4.2.20 Postavljanje životne linije novog objekta u dijagram slijeda

Slika 4.2.21 prikazuje izgled sučelja dijagrama slijeda nakon postavljanja svih potrebnih aktora i objekata. U okviru projektnog preglednika moguće je vidjeti pod stavkom dijagrama slijeda ('Pregled proizvoda') popis objekata aktora 'Korisnik' i objekta 'Web sučelje' (sa stereotipom <<*boundary*>>). Stavke koja se odnosi na aktora 'Baza podataka' nema jer nije stvoren novi objekt tog aktora već je u dijagram samo postavljena poveznica na njega. Aktora 'Baza podataka' moguće je vidjeti, kao i sve ostale aktore za neki dijagram slučajeva uporabe, u popisu aktora dijagrama slučajeva uporabe u kojem je definiran.

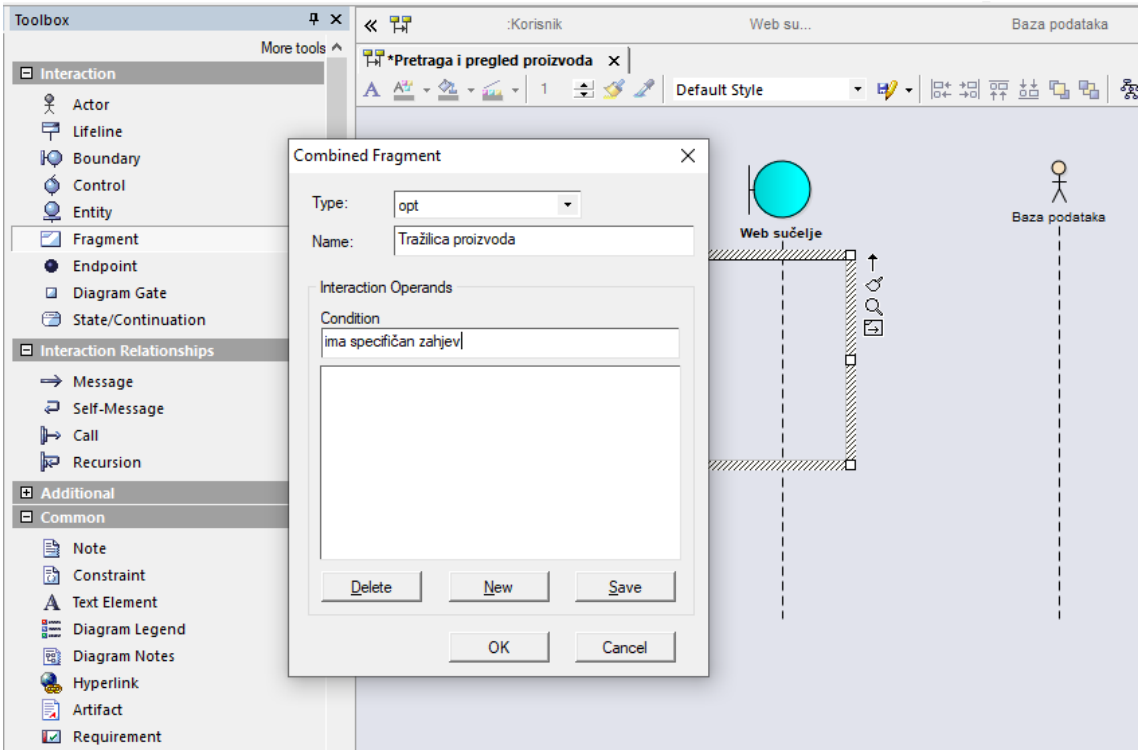


Slika 4.2.21 Aktori i objekti sa svojim životnim linijama

U dijagramu je moguće i u nazivu aktora vidjeti razliku između aktora 'Korisnik' i aktora 'Baza podataka' s obzirom na različite načine umetanja.

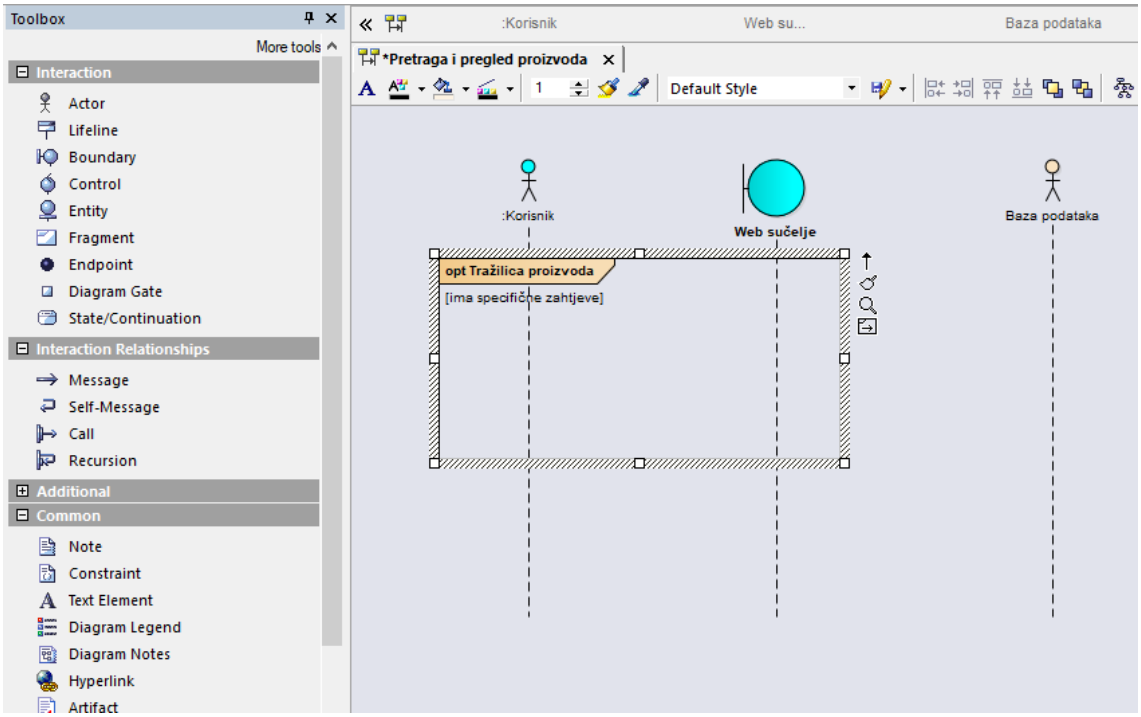
Umetanje fragmenata

Element ili grafički blok vrste *Fragment* se umeće iz alatnog okvira *Interaction*, Slika 4.2.17. Postoji više vrsta konstrukata koji se mogu prikazati ovim elementom pa je potrebno iz padajućeg izbornika koji se nudi pod stavkom *Type:* izabrati odgovarajuću stavku, npr. *opt* kao oznaka izbora opcije, nakon čega se unosi naziv opcije i uvjet uporabe opcije, Slika 4.2.22.



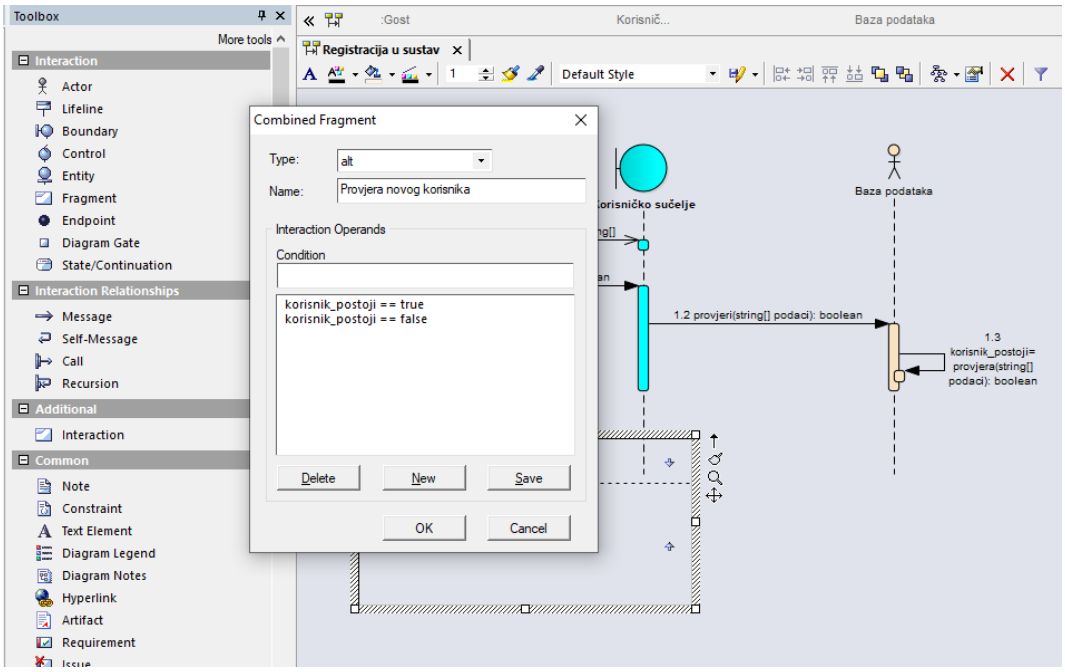
Slika 4.2.22 Postavljanje elementa vrste Fragment u dijagram slijeda

Potvrdom unosa uvjeta opcije element vrste *Fragment* se pojavljuje u dijagramu, Slika 4.2.23. Isto tako, vrsta fragmenta *alt* bi odgovarala notaciji uvjetnog grananja (ako-inače), a vrsta fragmenta *loop* notaciji petlje ili uzastopnog ponavljanja.



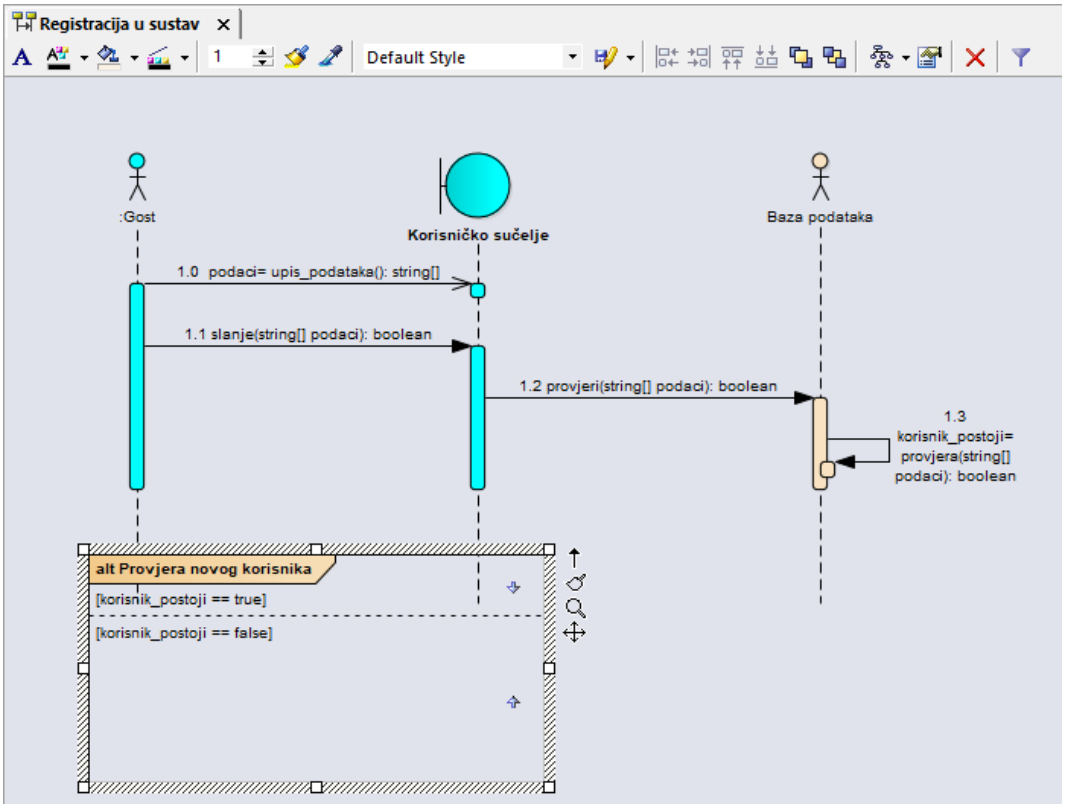
Slika 4.2.23 Izgled elementa vrste 'Fragment' na dijagramu slijeda

Fragment uvjetnog grananja se umeće u dijagram na način da se unose uvjeti izvršavanja za sve moguće grane, Slika 4.2.24.



*Slika 4.2.24 Definicija uvjetnog grananja (*alt*) na dijagramu slijeda*

Potvrdom unosa uvjeta grananja prikazuje se fragment uvjetnog grananja s dva usporedba odjeljka ili područja unutar fragmenta, Slika 4.2.25.

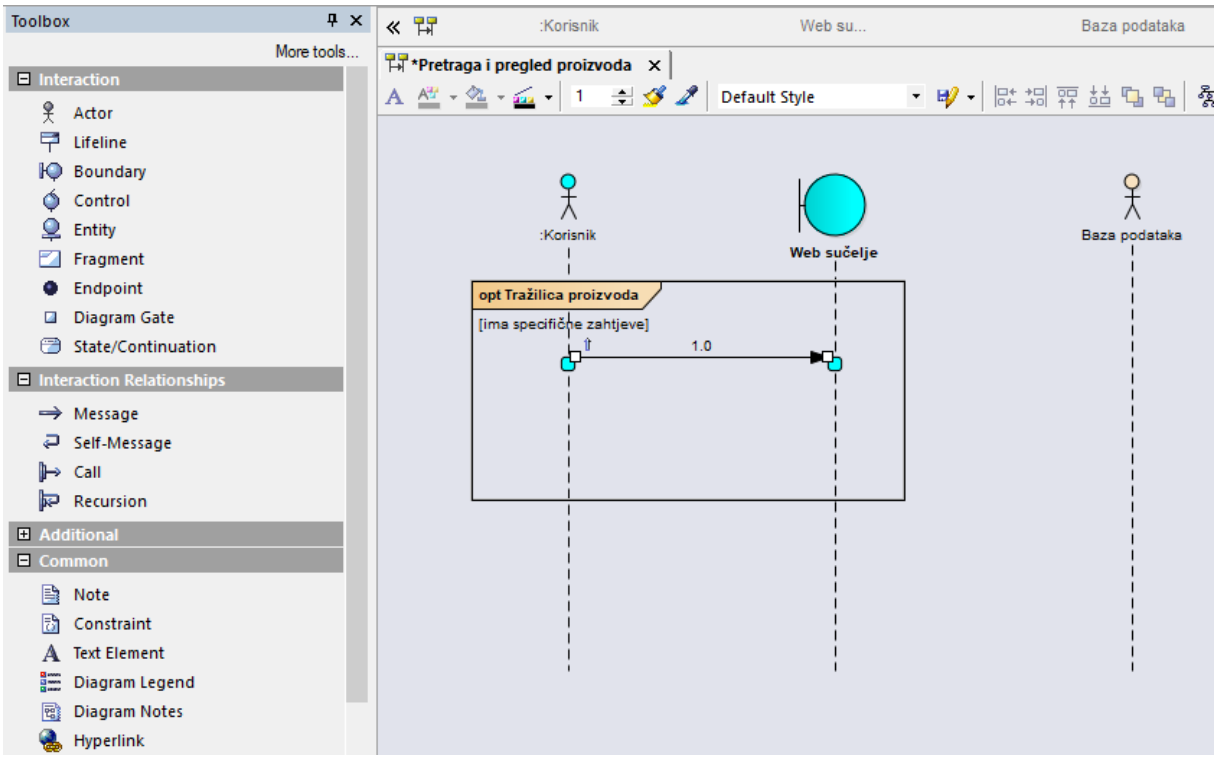


Slika 4.2.25 Prikaz uvjetnog grananja na dijagramu slijeda

Na sličan način se za fragment izvođenja petlje unosi uvjet izvršavanja petlje, čime se u dijagramu dobiva notacija fragmenta s oznakom *loop*.

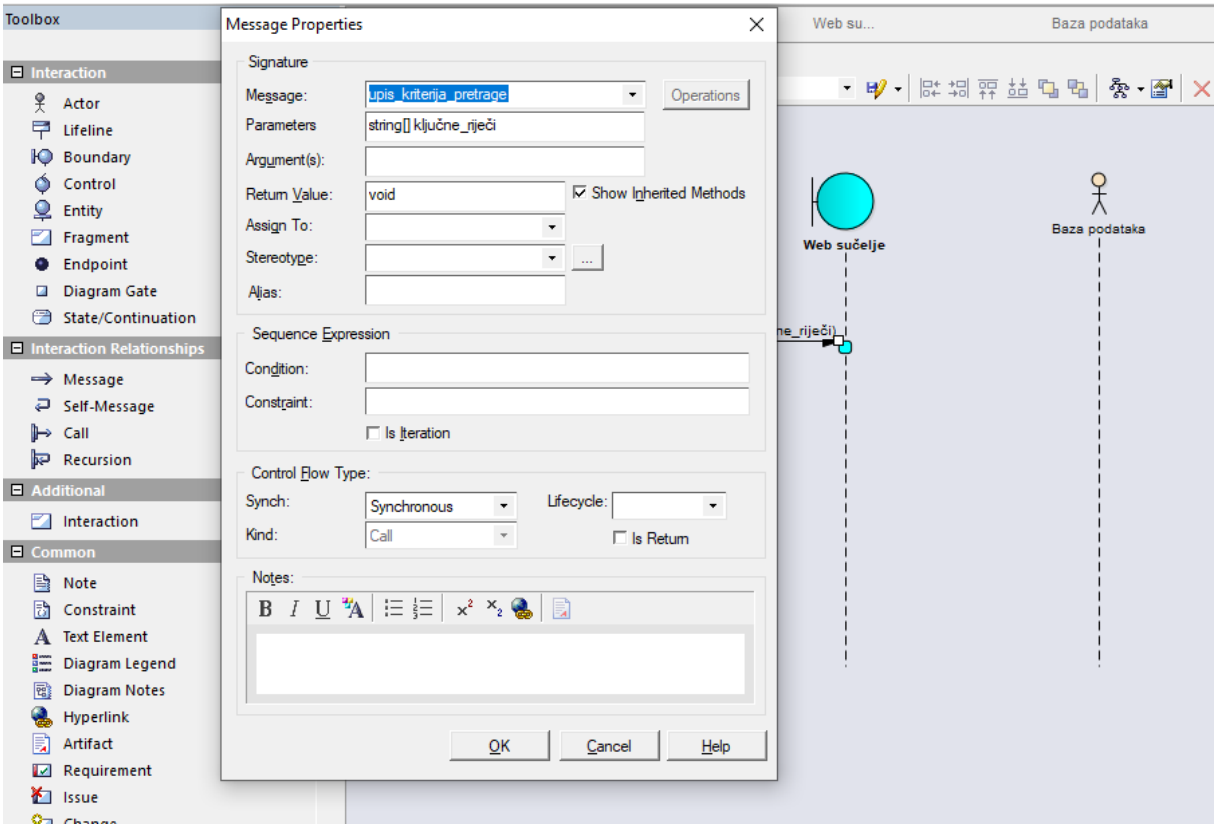
Umetanje poruke

Nova poruka se u dijagram umeće izborom stavke *Message* iz alatnog okvira *Interaction Relationships* te povlačenjem nove linije pokazivačem miša u smjeru od životne linije objekta aktora ili objekta koji šalje poruku do životne linije aktora ili objekta kome je poruka namijenjena. Potom se pojavljuje nova poruka između dotičnih životnih crta koja u početku ima samo oznaku rednog broja (*1.0*), Slika 4.2.26.
Detaljno se poruka može opisati unutar alatnog okvira *Properties*, Slika 4.2.27. Unosi se naziv poruke pod stavkom *Message:*, parametri pod stavkom *Parameters:* i povratna vrijednost pod stavkom *Return Value:*. Poruka je podrazumijevano sinkrona što se može promijeniti pod stavkom *Synch:*. Označavanjem polja *Is Iteration* naglašava se mogućnost ponavljanja slanja. Unutarnji postupak ili procedura odnosno „poruka samom sebi“ umeće se izborom stavke *Message* i dvostrukim klikom na životnu liniju objekta koji je u pitanju.



Slika 4.2.26 Nova poruka u dijagramu slijeda

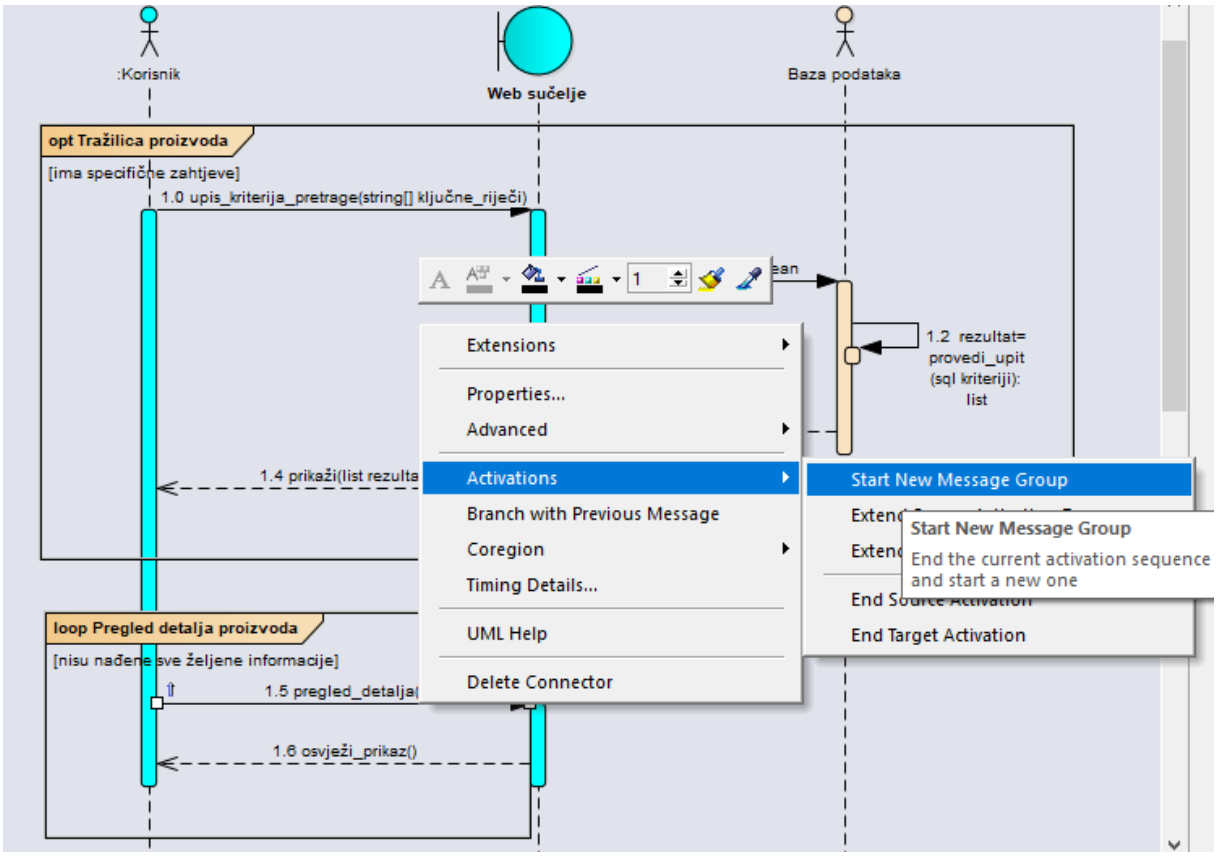
Povratna poruka se definira u okviru prozora s detaljima poruke (Slika 4.2.27) označavanjem polja *Is Return*. Početne poruke su bile sinkrone a povratne poruke su prikazane izričito odgovarajućom notacijom isprekidane crte.



Slika 4.2.27 Upis detalja poruke dijagrama slijeda

Načini pobrojavanja

Desnim klikom na tu poruku i izborom *Activations* → *Start New Message Group* započinje se nova skupina poruka koja kreće s pobrojavanjem na istoj razini na kojoj je poruka s kojom je započelo pobrojavanje prethodne skupine poruka, npr. od *2.0* nadalje ako je prethodna skupina poruka bila pobrojana s *1.**, Slika 4.2.28.



Slika 4.2.28 Pokretanje nove skupine poruka na dijagramu slijeda

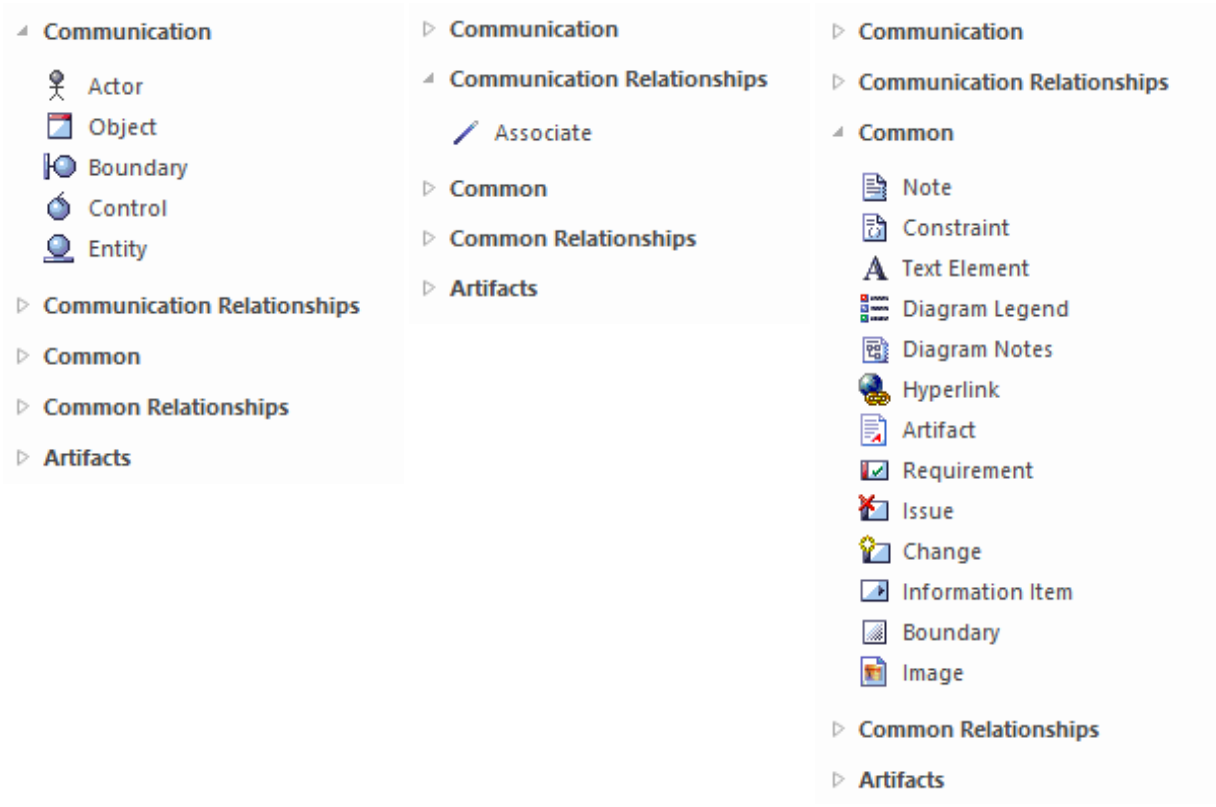
Primjeri prikaza različitih razina pobrojavanja mogu se vidjeti u dijagramima Slika 2.5.2 i Slika 2.5.3.

Referenciranje drugog dijagrama

Neki dijagram se postavlja unutar nekog drugog dijagrama povlačenjem njegove stavke iz okvira projektnog preglednika nakon čega je prikazan simbolom pravokutnika sa svojim nazivom s oznakom 'ref' u gornjem lijevom kutu. Slika 2.5.4 daje primjer jednog takvog umetanja.

Dijagram komunikacije

Alatni okviri dijagrama komunikacije prije svega sadrže stavke koje se odnose na sudionike komunikacije pod naslovom *Communication*, povezanost sudionika pod naslovom *Communication Relationships* kao i neke opće elemente (*Common*) u pravilu zajedničke svim vrstama dijagrama, Slika 4.2.29.



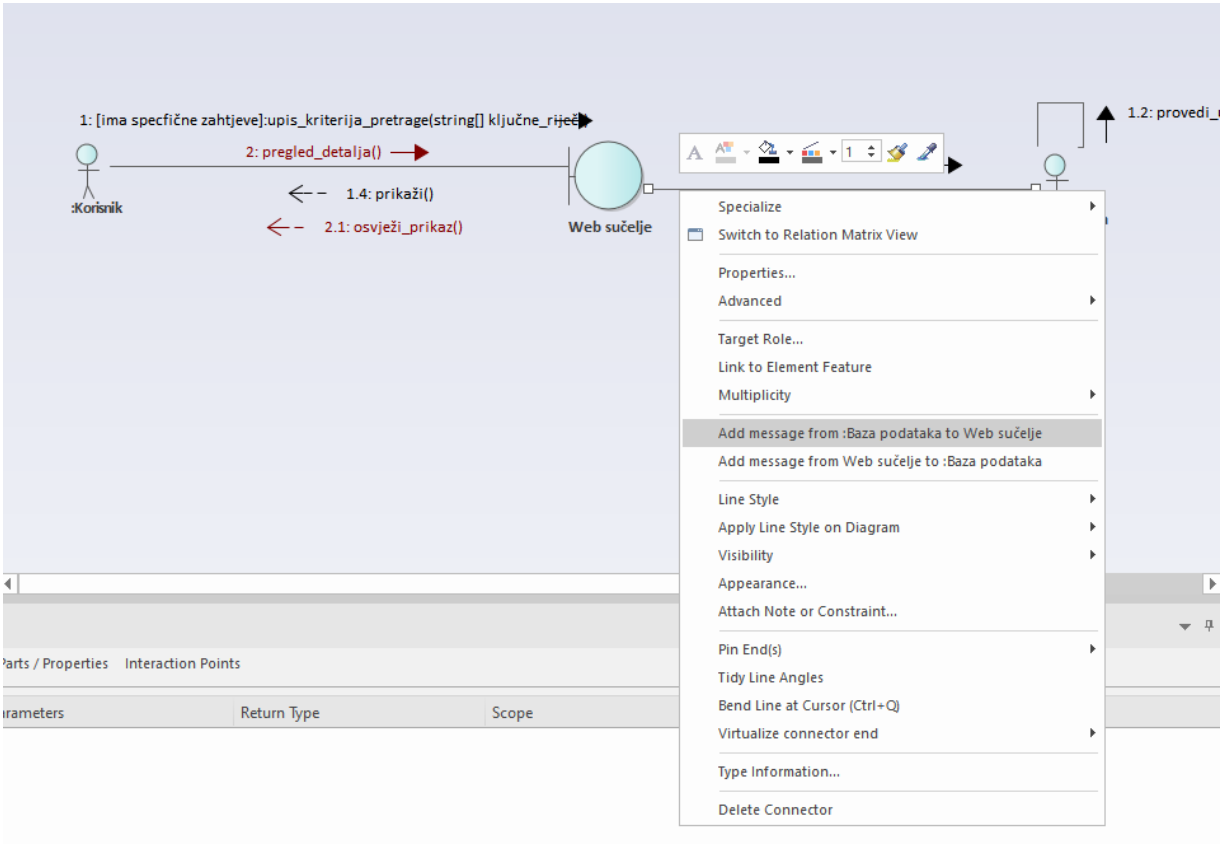
(a) *(b)* *(c)*

Slika 4.2.29 Alatni okviri dijagrama komunikacije

Jedina vrsta veze koju je moguće unijeti u dijagram je veza običnog pridruživanja na kojoj se onda definiraju poruke između povezanih sudionika.

Umetanje poruka

Poruka se umeće na prethodno definiranu vezu pridruživanja između sudionika koji komuniciraju desnim klikom na vezu i izborom stavke *Add Message from … to …* gdje se konkretno navodi prema nazivima sudionika tko kome šalje poruku.

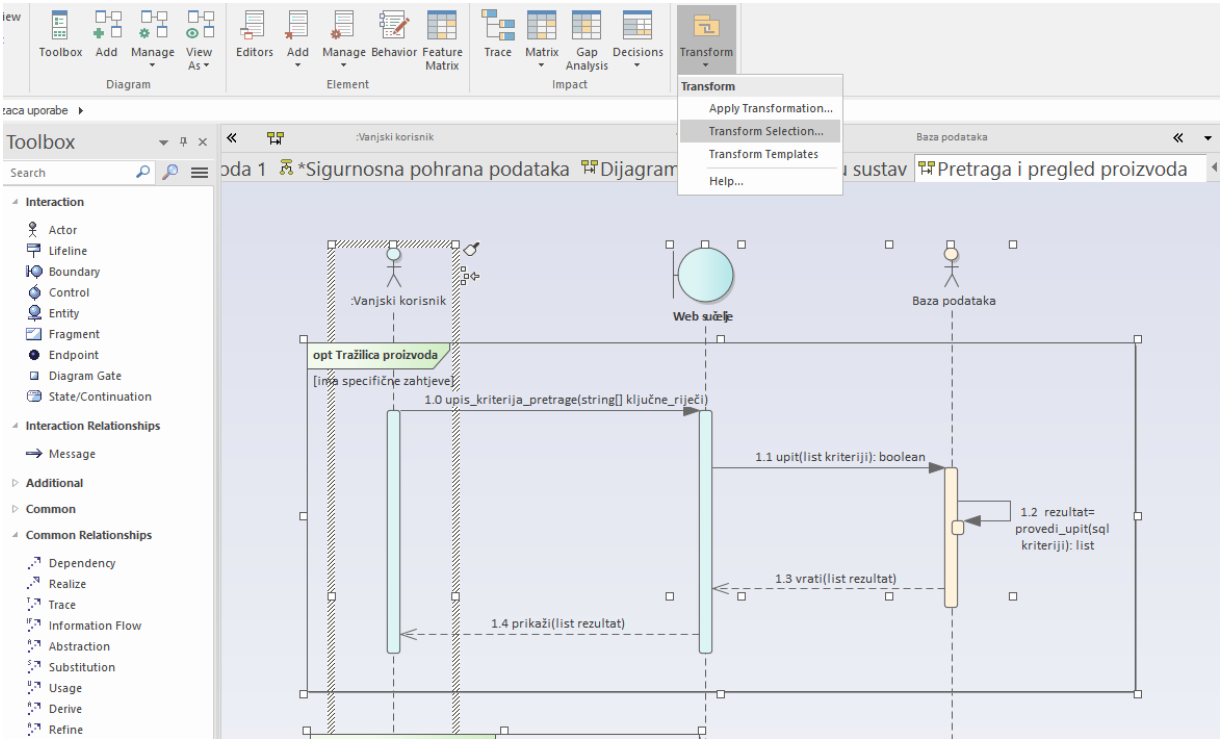


Slika 4.2.30 Umetanje poruka između povezanih sudionika

Na jednoj definiranoj vezi umeće se nebrojeni broj poruka koje se razmjenjuju između povezanih sudionika.

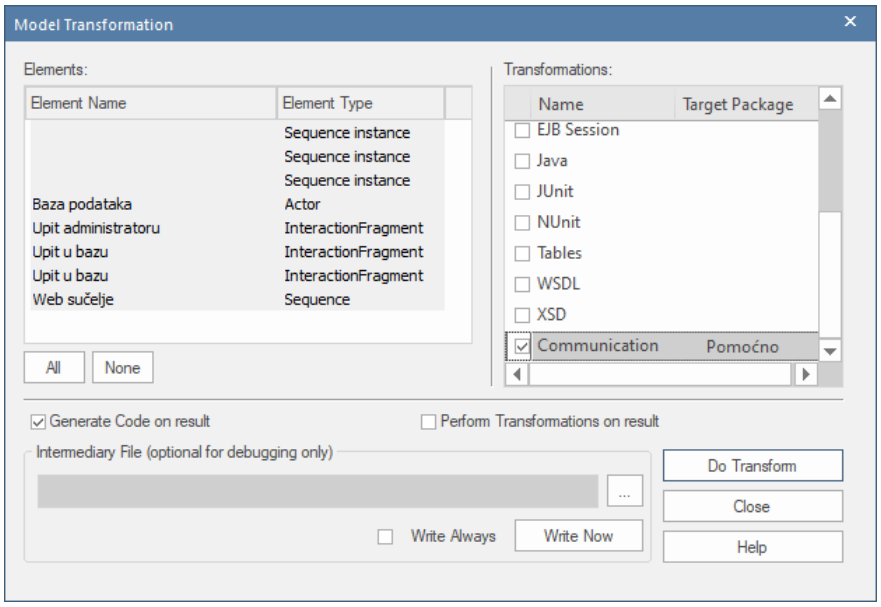
Automatska pretvorba iz dijagrama slijeda

Dijagram komunikacije se automatski može dobiti za neki dijagram slijeda izborom alata *Transform* izbornika *Design*, Slika 4.2.31.



Slika 4.2.31 Postupak automatske pretvorbe dijagrama slijeda u dijagram komunikacije

Posebnim dijalogom se treba potvrditi kakva se transformacija želi napraviti, Slika 4.2.32.

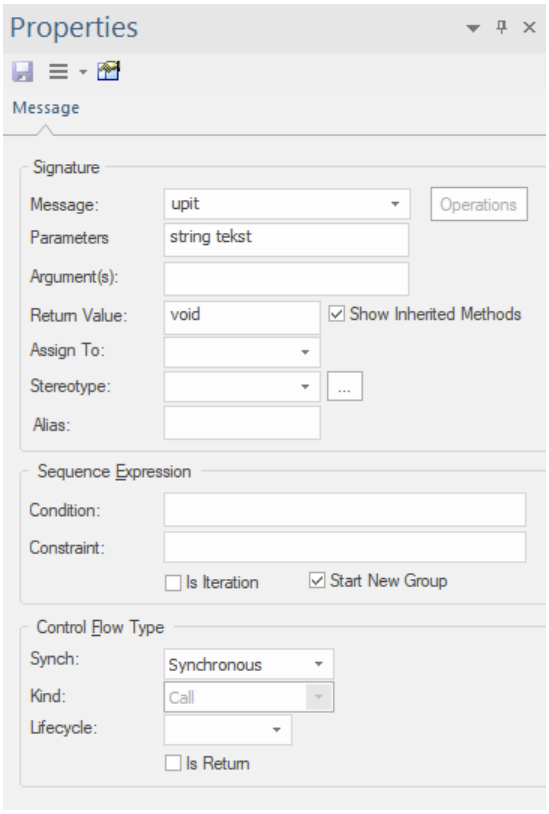


Slika 4.2.32 Pokretanje automatske pretvorbe dijagrama slijeda u dijagram komunikacije

Klikom na gumb *Do Transform* pokreće se željena pretvorba.

Nova skupina pobrojavanja poruka

Umetanje nove skupine pobrojavanja postiže se označavanjem stavke *Start New Group* unutar okvira *Properties* za poruku od koje nova skupina poruka kreće i spremanjem postavke s kombinacijom tipki *Ctrl+S* te klikom na ikonicu „diskete“, tj. pohrane u gornjem lijevom kutu, Slika 4.2.33.

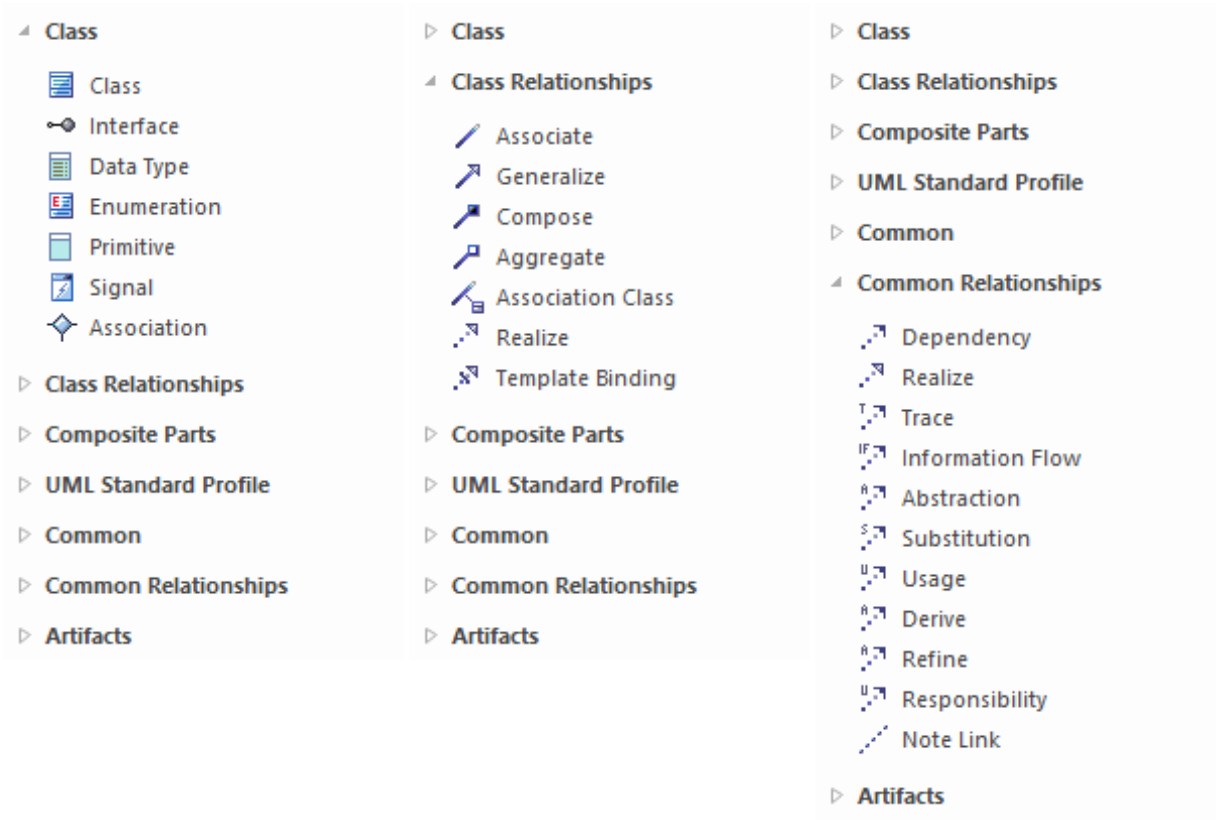


Slika 4.2.33 Definicija nove skupine pobrojavanja poruka

Primjeri umetanja novih razina pobrojavanja mogu se vidjeti na izrađenim dijagramima uvodno elaboriranog primjera modeliranja, Poglavlje 2.6.

Dijagram razreda

Sadržaj alatnog okvira *Toolbox* se prilagođava principima oblikovanju dijagrama razreda pomoću dva tome svojstvena okvira s naslovima *Class* i *Class Relationships*, Slika 4.2.34.



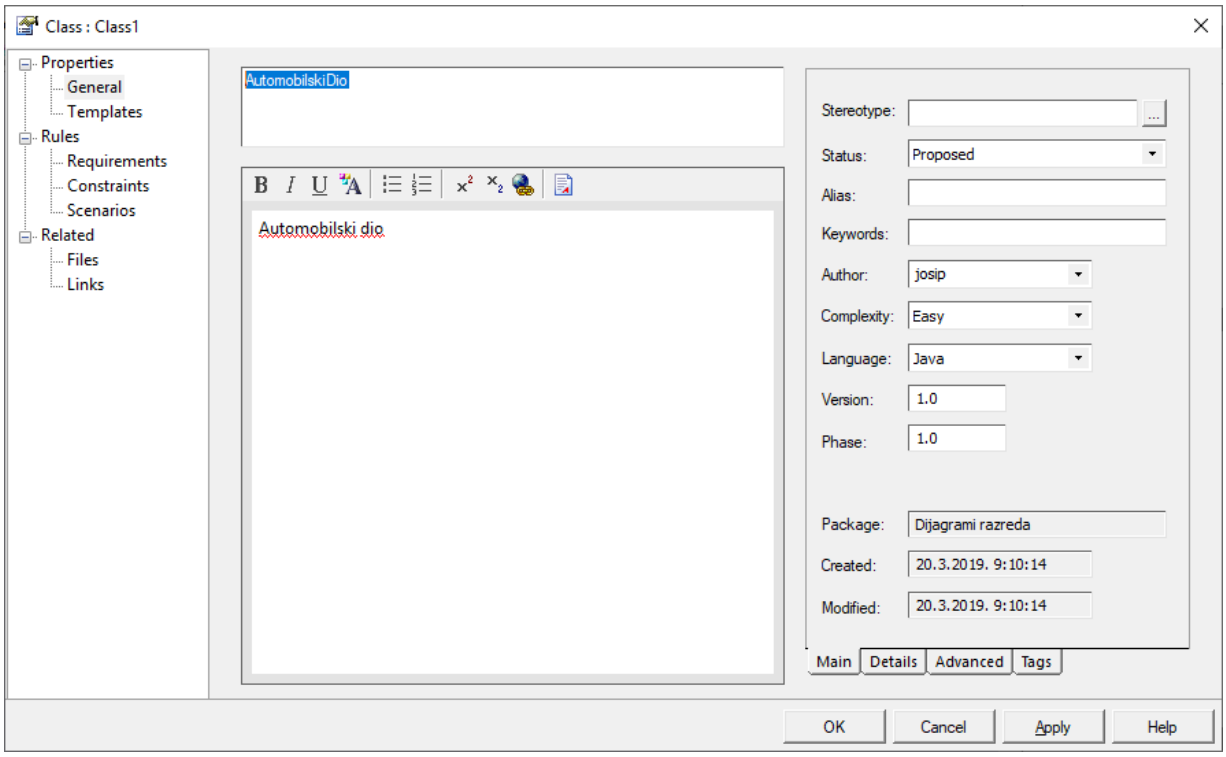
(a) *(b)* *(c)*

Slika 4.2.34 Alatni okviri dijagrama razreda

Alatni okvir *Class* nudi na izbor umetanja različitih oblika entiteta razreda kao što su razred (*Class*), sučelje (*interface*) i pobrojavanje (*Enumeration*) dok alatni okvir *Class Relationships* nudi umetanje mogućih vrsta odnosa između njih, pridruživanja, agregacije i poopćenja.

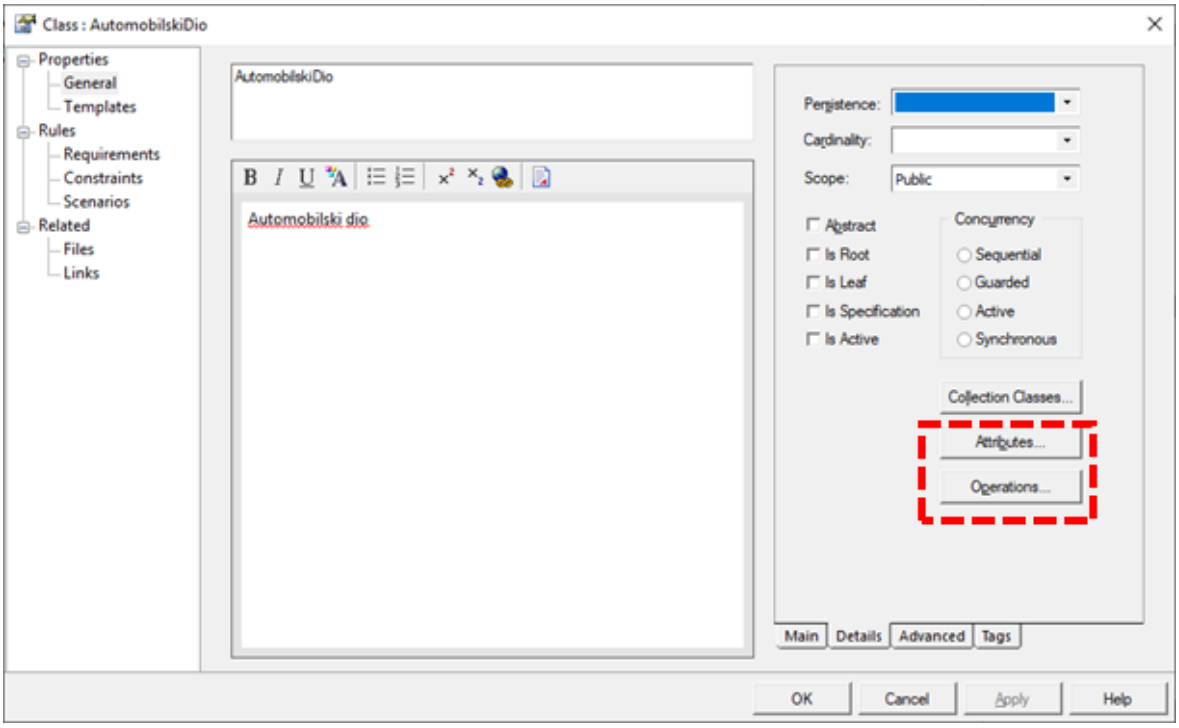
Definicija razreda

Izborom stavke *Class* iz alatnog okvira *Class* započinje se definicija novog razreda, Slika 4.2.35. U posebnom okviru pod karticom *Main* postavljaju se osnovne opisne informacije o razredu, između ostalog i pretpostavljeni programski jezik razvoja (u ovom slučaju postavljeno je na programski jezik Java) što je osobito važno ako se koristi funkcionalnost generiranja koda iz UML specifikacije.



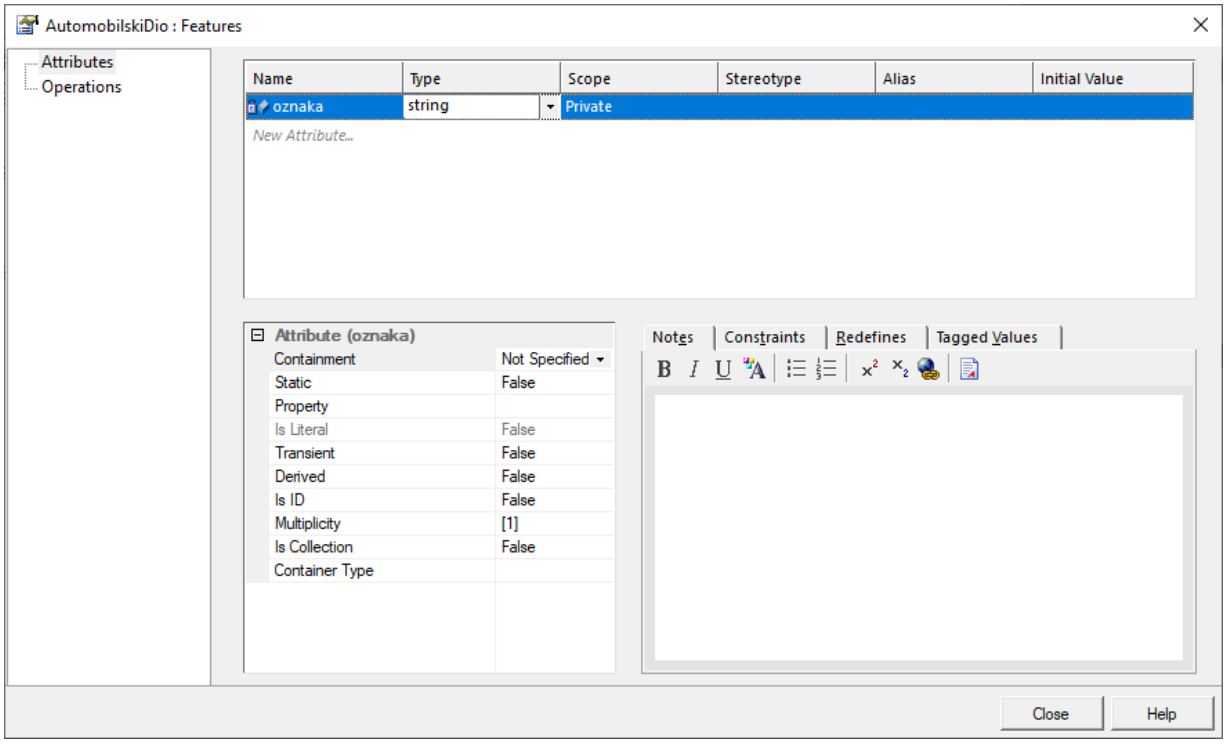
Slika 4.2.35 Definicija novog razreda

Izborom kartice *Details* sadržaj okvira se mijenja te je moguće pristupiti definiciji atributa i metoda razreda putem gumba *Attributes...* i *Operations…*, Slika 4.2.36., Slika 4.2.37.



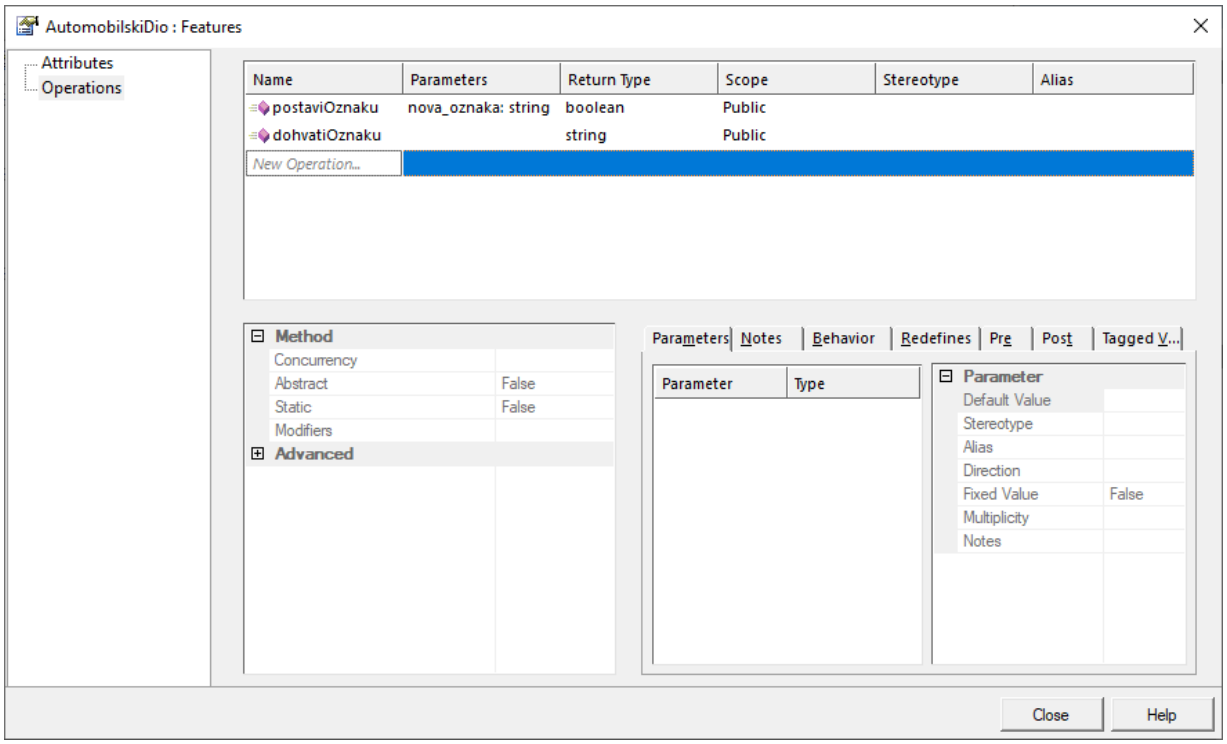
Slika 4.2.36 Pristup definiciji atributa i metoda razreda

Alat *Enterprise Architect* ima predefinirani skup generičkih vrsta podataka, ali je pojedine tipove potrebno i moguće dodatno definirati kako bi bili na izbor pri definiciji vrste atributa. To je moguće jednostavno učiniti prema uputi u jednom od sljedećih odlomaka naziva „Generiranje izvornog koda“.



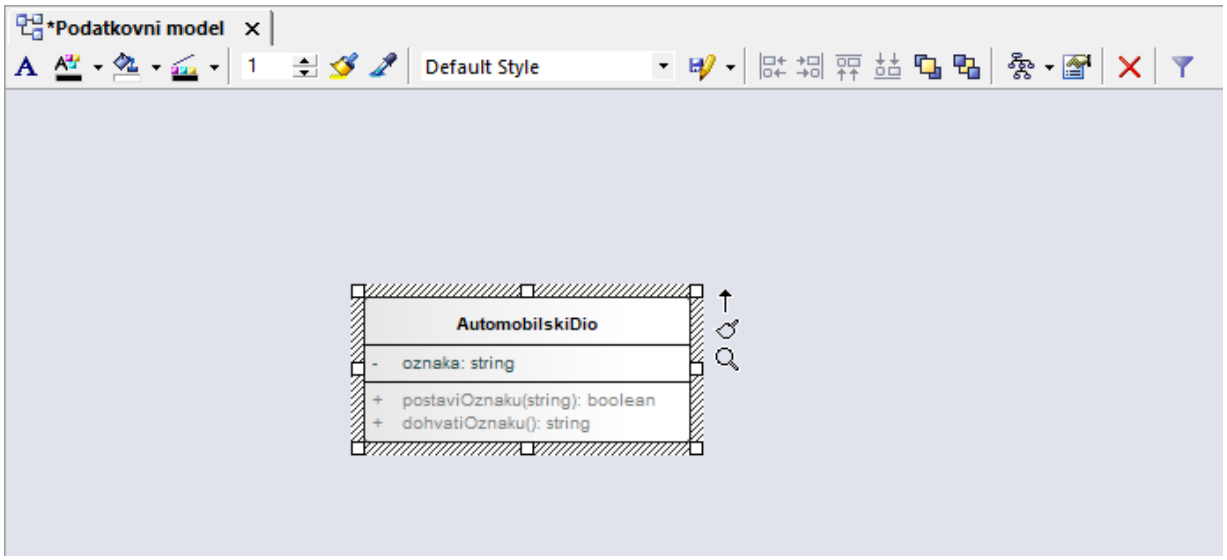
Slika 4.2.37 Definicija novog atributa razreda

Uz naziv i vrstu atributa moguće je definirati i njegov doseg vidljivosti (*Scope*) te je nakon toga moguće definirati i odgovarajuće metode razreda izborom stavke *Operations* iz pokrajnjeg okvira. Ako se postavi doseg vidljivosti atributa kao privatni (*Private*), običaj je definirati metode postavljanja (engl. *set*) i dohvaćanja (engl. *set*) vrijednosti atributa pa se to i u ovome primjeru čini definicijom metoda 'postaviOznaku' i 'dohvatiOznaku' s odgovarajućim definicijama parametara (*Parameters*) i povratne vrste podataka (*Return Type*) te dosegom vidljivosti (*Public*), Slika 4.2.38.



Slika 4.2.38 Definicija novih metoda razreda

Po završetku definicije njegovih atributa i metoda, prikaz novog razreda se pojavljuje unutar radne površine dijagrama, Slika 4.2.39.



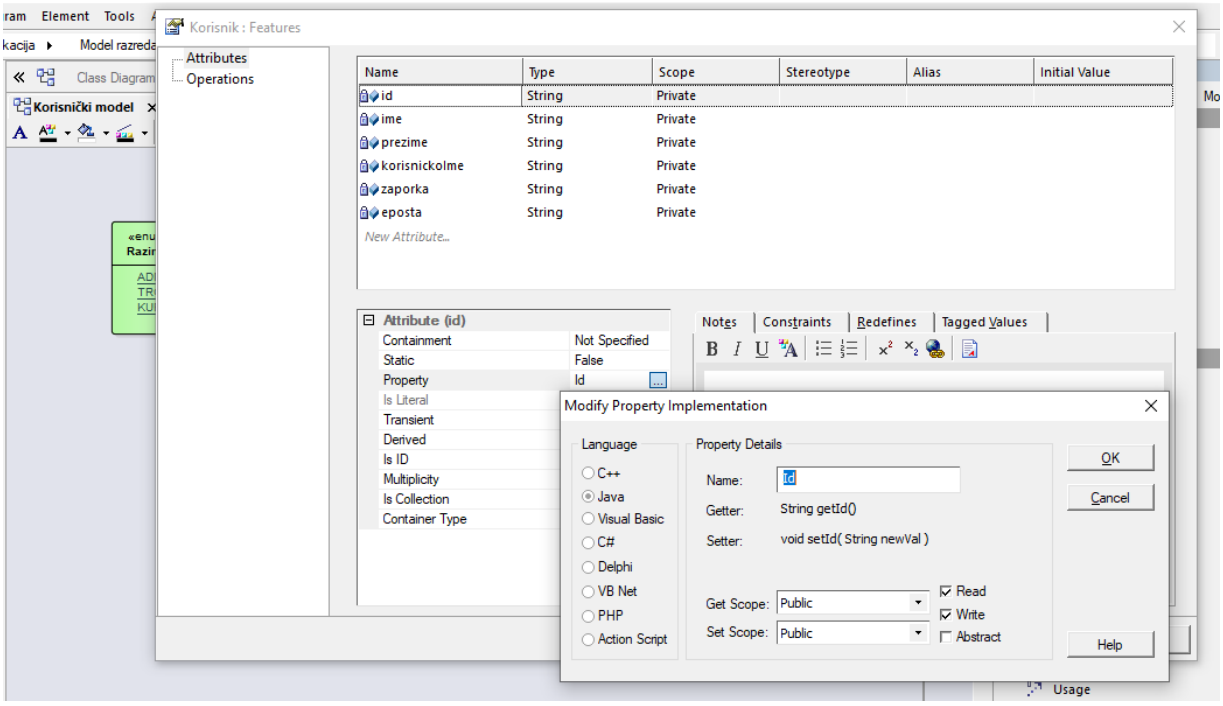
Slika 4.2.39 Prikaz novog razreda na dijagramu

Svaki razred, pa tako i razred 'AutomobilskiDio', može imati više atributa različite vrste. Ovdje je definiran samo jedan njemu svojstven atribut jedinstvene oznake proizvođača, ali za njegov potpuniji opis razumljiv kupcima i trgovcima potrebno ga je dopuniti navodima novih atributa

koji nerijetko nisu oni generičke vrste podataka, već prema vrstama definiranim drugim razredima. O tome se detaljnije govori u odlomku naziva „Povezivanje razreda“.

Definicija svojstava

Metode za dohvat i postavljanje vrijednosti atributa koji se u nekim jezicima podrazumijevaju pod **svojstvima** razreda je naravno moguće napisati ručno u bilo koje vrijeme razvoja programskog koda, ali ih je moguće definirati već u samom dijagramu da bi se automatski stvorile pozivom funkcionalnosti generiranja programskog koda. Mogu se definirati na način da se pojedinom atributu definira polje *Property* kao oznake svojstva koje posjeduje i koje se postavlja i dohvaća, Slika 4.2.40.

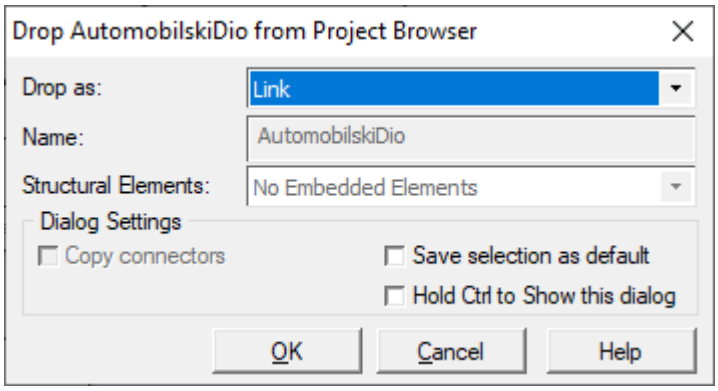


Slika 4.2.40 Postavljanje polja Property *za pojedini atribut razreda*

U dijaloškom okviru s popisom atributa izabere se atribut te se postavi naziv svojstva pri čemu alat automatski daje podrazumijevani naziv metoda za postavljanje i dohvat atributa, tj. ako se za atribut 'id' postavi naziv svojstva 'Id', metoda za dohvat (engl. *getter*) će imati naziv 'getId()', a metoda za postavljanje (engl. *setter*) naziv 'setId()'.

Umetanje postojećeg razreda

Razred 'AutomobilskiDio' se postavlja u dijagram povlačenjem njegove stavke iz okvira *Project Browser* (nije potrebno ponovno definirati razred 'AutomobilskiDio'). Povlačenjem stavke i izborom opcije *Link* pod stavkom *Drop As* (Slika 4.2.41) stvara se veza na postojeći razred te će se svaka promjena u definiciji razreda odraziti u prikazu oba dijagrama gdje se razred uključuje.

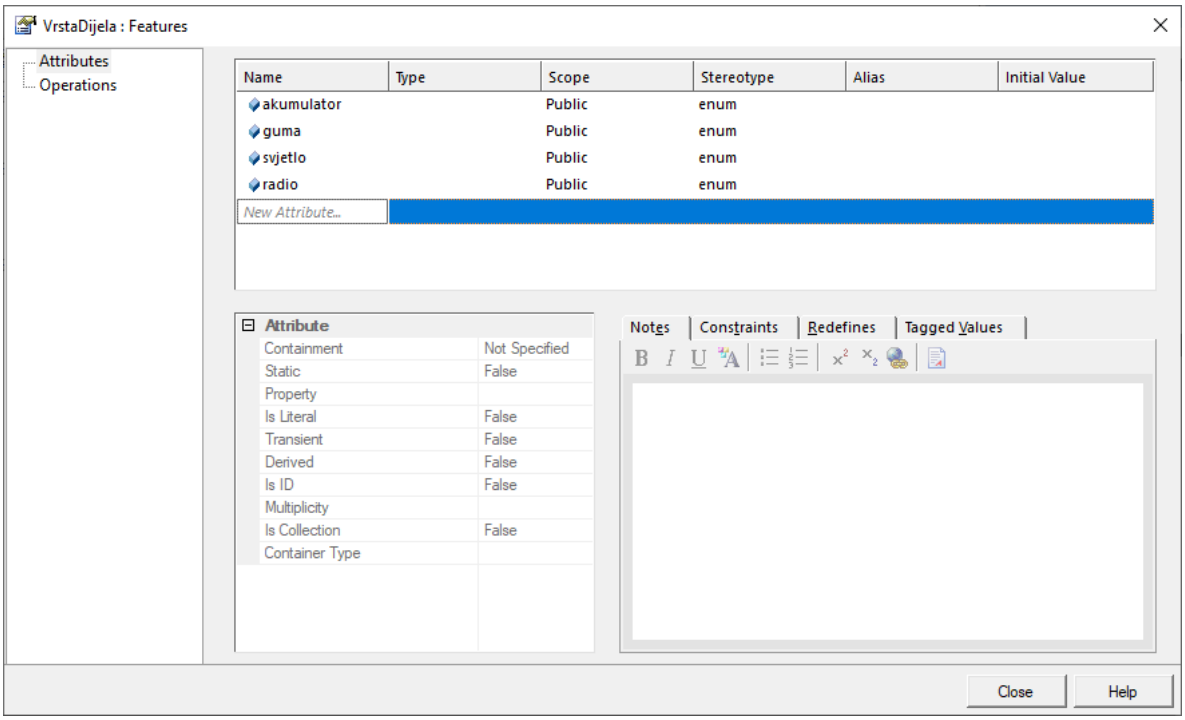


Slika 4.2.41 Postavljanje postojećeg razreda u dijagram

Isto vrijedi i za razrede 'ProizvodjacDijelova' i 'Korisnik' koji se prikazuju i na ovom dijagramu kao i na onom podatkovnog modela, odnosno korisničkog modela.

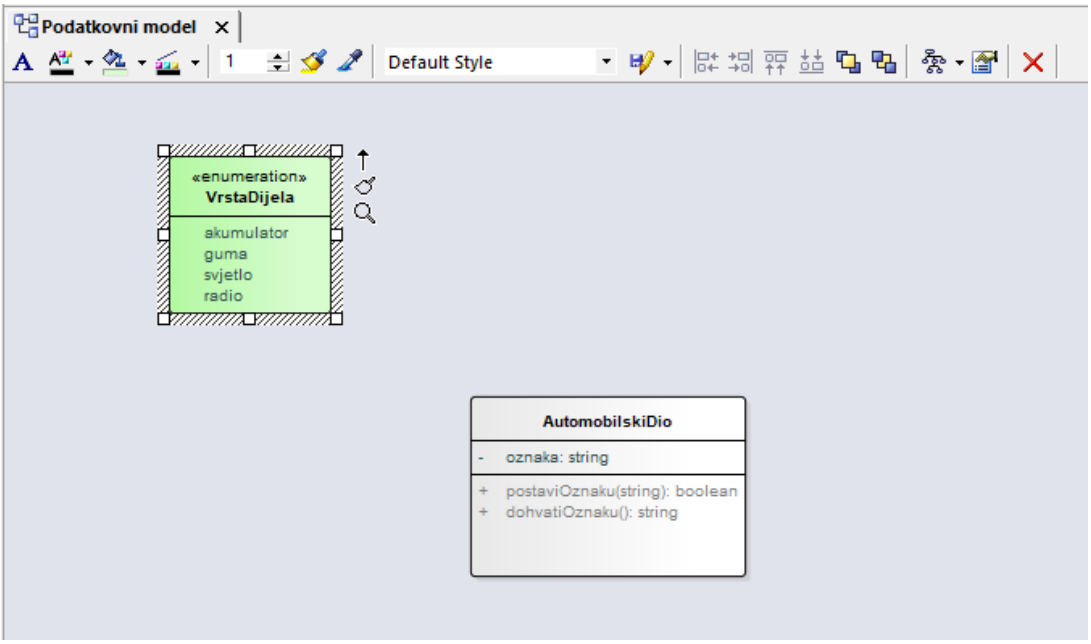
Povezivanje razreda

Svaki automobilski dio veže više različitih informacija koje su važne i za trgovce i za kupce. To su ponajprije: vrsta dijela, model automobila, proizvođač i lista dobavljača. Svaki od navedenih je zaseban entitet koji se oblikuje vlastitom strukturom ili razredom.
Vrsta dijela je najjednostavniji entitet koji govori samo o kojoj okvirnoj vrsti ili skupini automobilske opreme je riječ: akumulatorima, gumama, svjetlima, autoradiju. Ne pretpostavljaju se neke posebne odgovornosti za taj entitet, već ovaj entitet samo služi nabrajanju različitih vrsta dijelova. Zato je dovoljno taj entitet zapisati u obliku strukture pobrojavanja. Izborom stavke *Enumeration* i navodom takvih različitih atributa stvara se odgovarajući skup elemenata pobrojavanja, Slika 4.2.42.



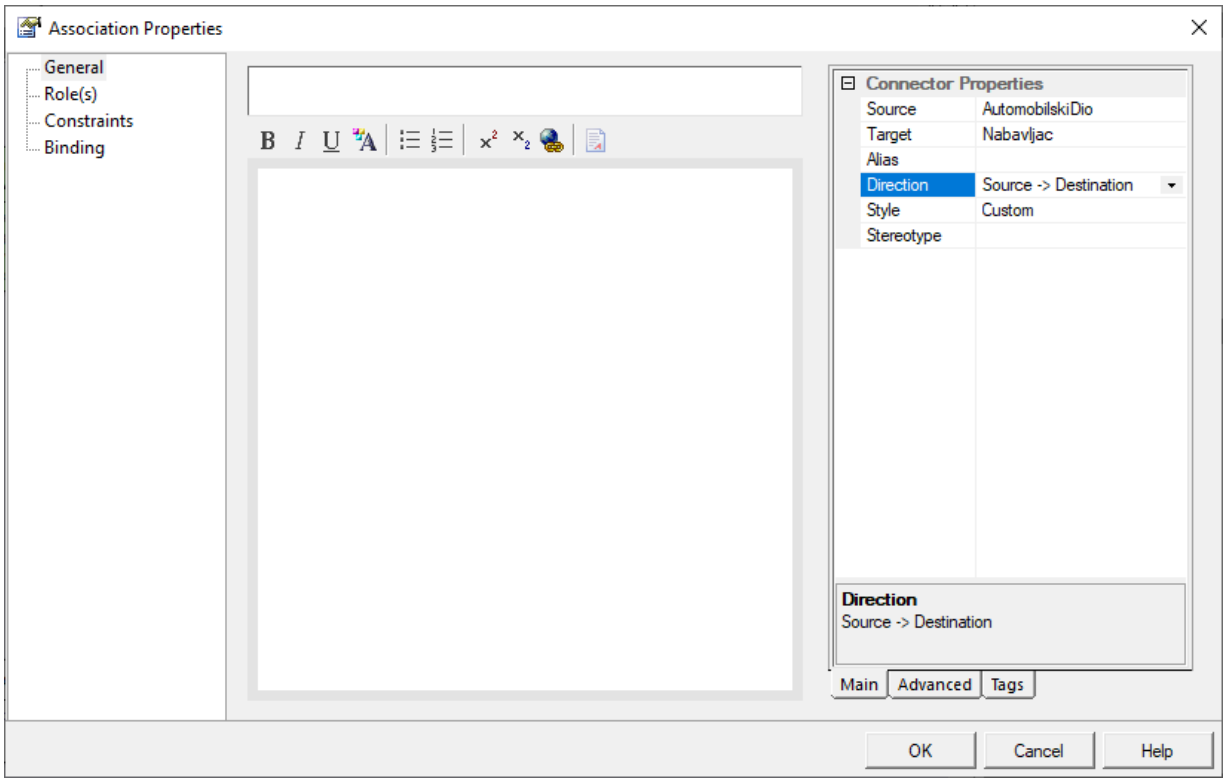
Slika 4.2.42 Definicija stavki pobrojavanja

Prikaz pobrojavanja na dijagramu razreda se bitno razlikuje od prikaza razreda, Slika 4.2.43.



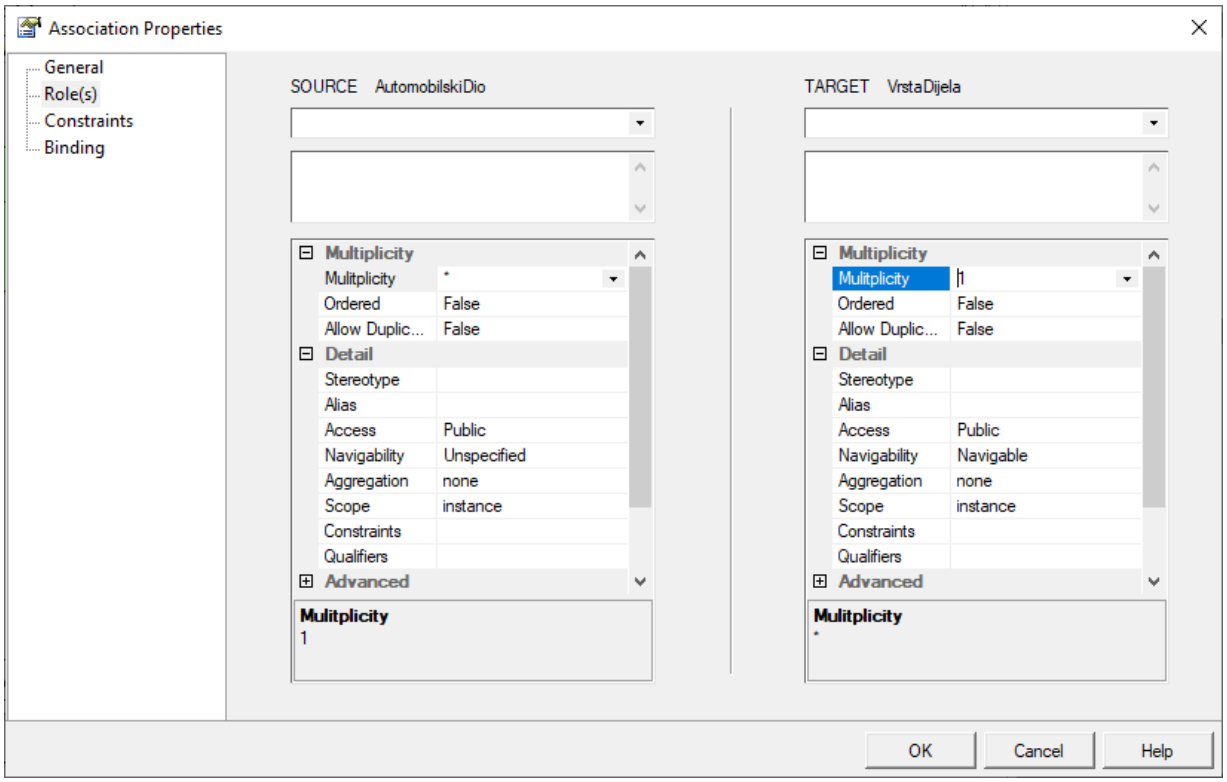
Slika 4.2.43 Pobrojavanje u dijagramu razreda

S obzirom na to da automobilski dio spada u neku skupinu dijelova, pripada nekoj vrsti koja se može definirati unutar strukture pobrojavanja *VrstaDijela* definira se veza pridruživanja izborom stavke *Associate* iz alatnog okvira *Class Relationships*. U novootvorenom prozoru *Association Properties* izbornika *General* može se postaviti usmjerenje veze (pod stavkom *Direction*), Slika 4.2.44.



Slika 4.2.44 Postavljanje usmjerenja veze pridruživanja

Tako se pod stavkom *Multiplicity* mogu definirati višestrukosti veze, Slika 4.2.45.

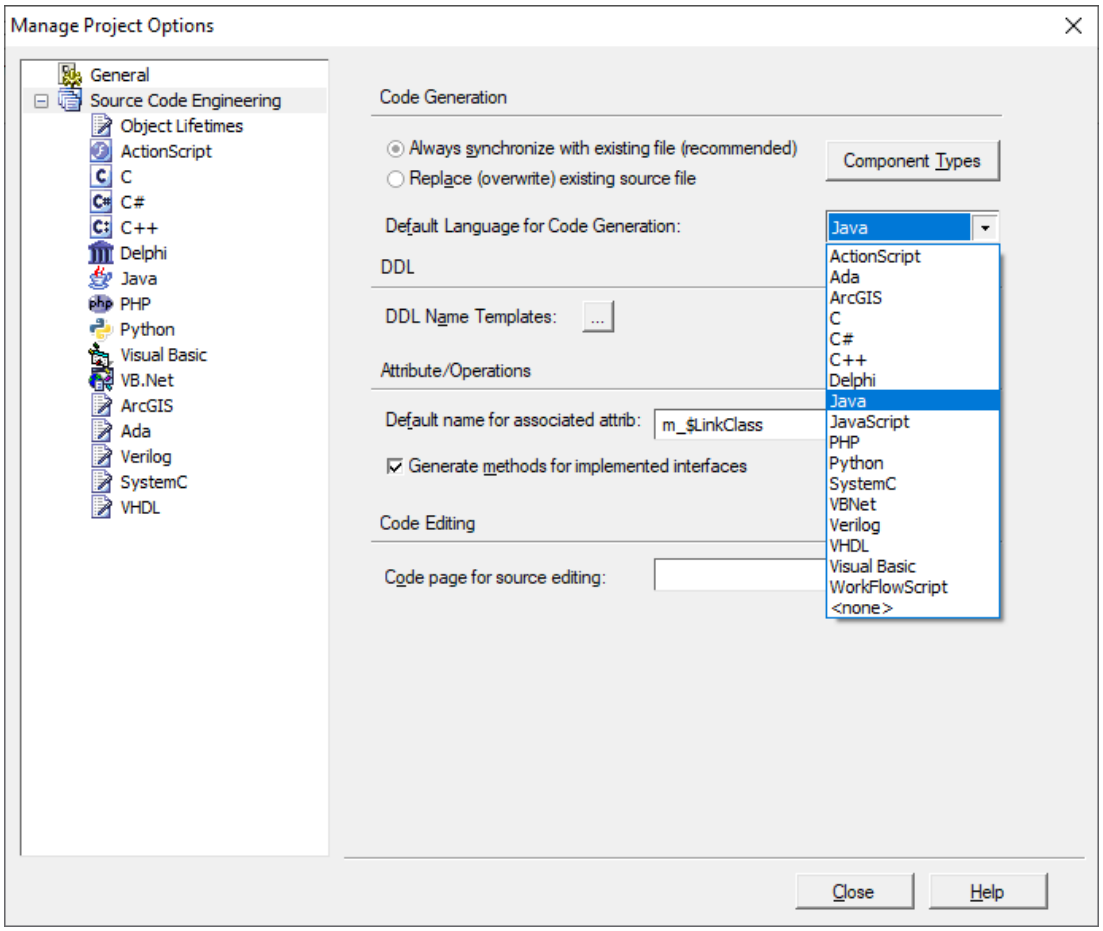


Slika 4.2.45 Postavljanje višestrukosti veze pridruživanja

Slika 2.7.1 prikazuje definirani odnos razreda 'AutomobilskiDio' i pobrojavanja 'VrstaDijela'. Razred 'AutomobilskiDio' pokazuje na, odnosno koristi entitet pobrojavanja da bi definirao skupinu dijelova kojoj pripada. Naznačene višestrukosti govore da jedan automobilski dio pripada samo jednoj vrsti dijelova (naznačeno s '1'), a da jedna vrsta dijelova može biti dodijeljena nebrojeno mnogo automobilskih dijelova (naznačeno sa '*').

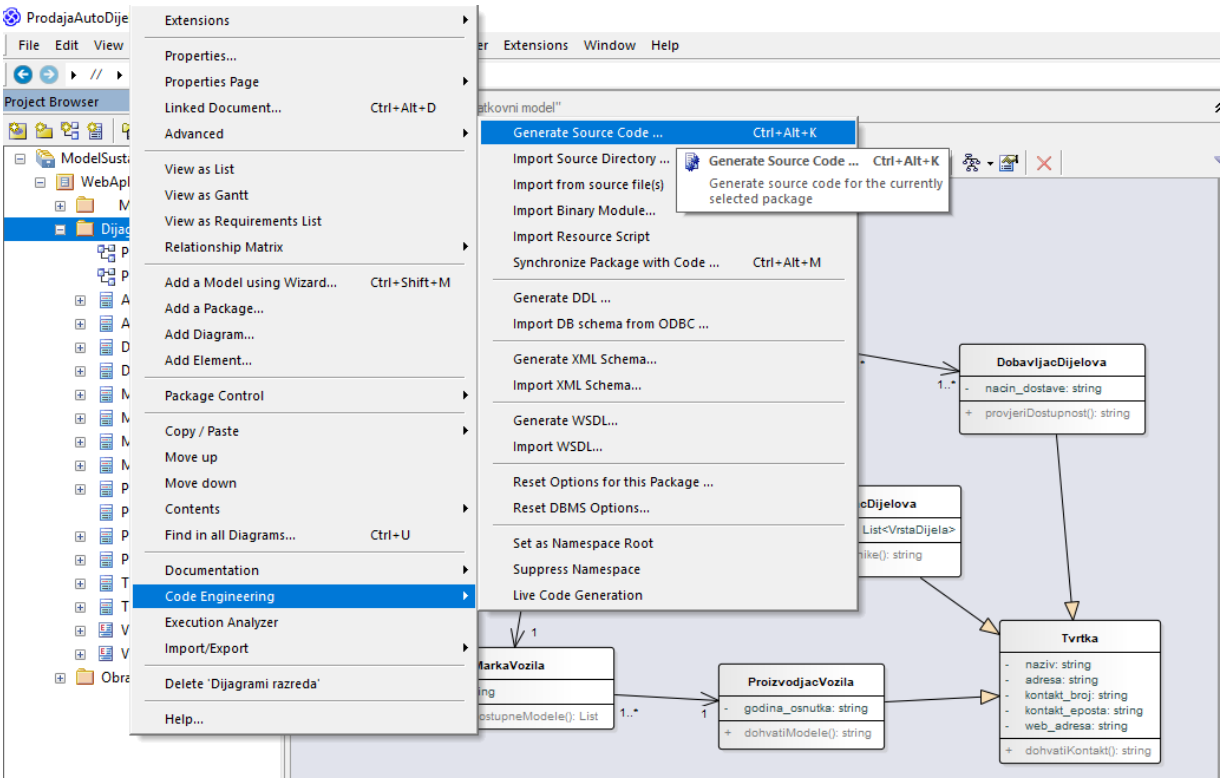
Generiranje izvornog koda

Prije samog generiranja koda potrebno je provjeriti ili postaviti ciljni programski jezik generiranja u opcijama projekta, Slika 4.2.46.



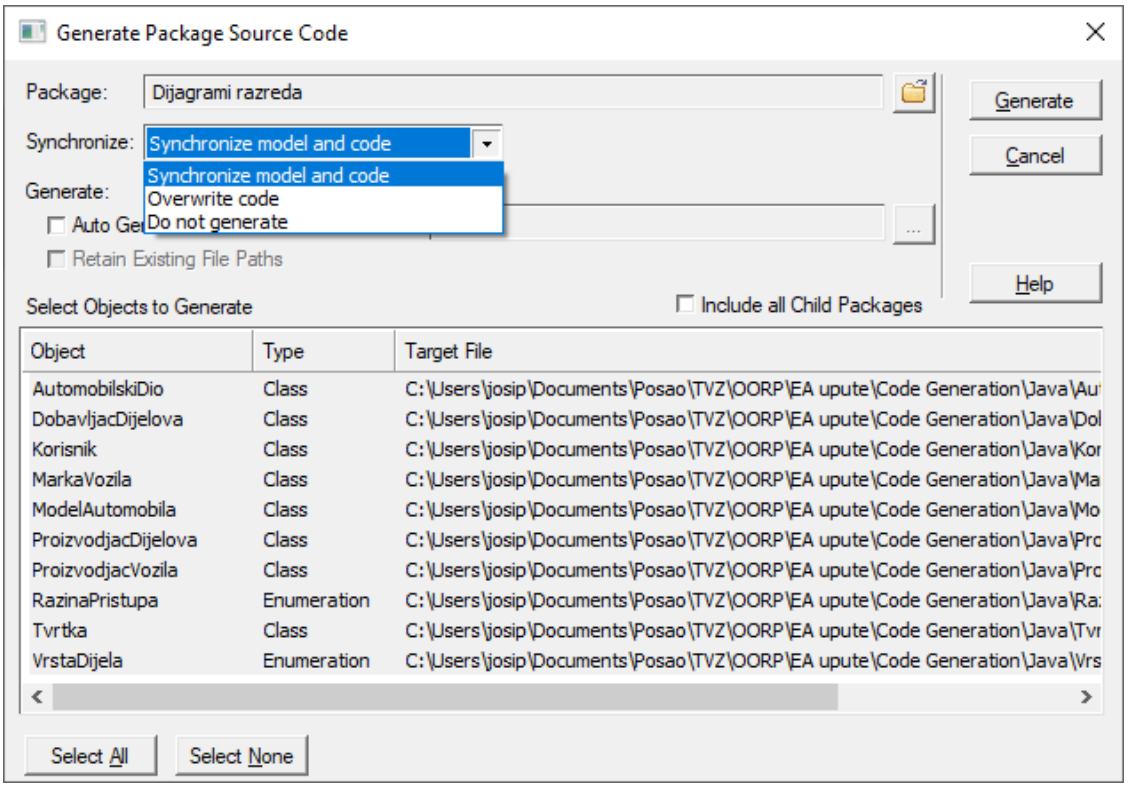
Slika 4.2.46 Postavljanje podrazumijevanog ciljnog jezika za generiranje izvornog koda

Nakon toga se za paket za koji se želi generirati kôd započinje ta funkcionalnost, Slika 4.2.47.



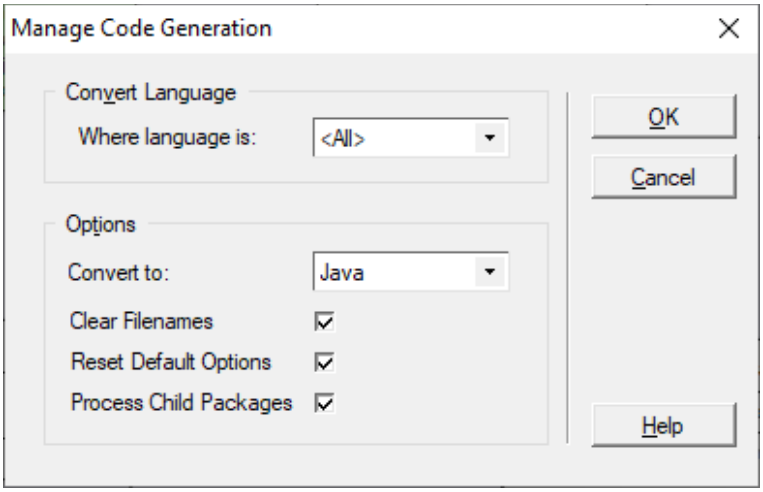
Slika 4.2.47 Poziv funkcionalnosti generiranja koda

Za sve strukture (razredi i pobrojavanja) odabranog paketa za koje se može generirati programski kôd moguće je odabrati opciju sinkronizacije s postojećim kodom (*Synchronize model and code*) ili prepisivanja postojećeg koda (*Overwrite code*), Slika 4.2.48. U popisu datoteka generiranog programskog koda u odabranom jeziku (ovdje se pretpostavlja Java) može se vidjeti da su za sve postojeće razrede i pobrojavanja stvorene odgovarajuće datoteke programskog koda, Slika 2.7.7.



Slika 4.2.48 Opcije pri generiranju novog programskog koda

Moguće je naravno i promijeniti ciljni programski jezik generiranja koda uz opciju brisanja prethodno postavljenih postavki uz odabir programskog jezika, Slika 4.2.49.

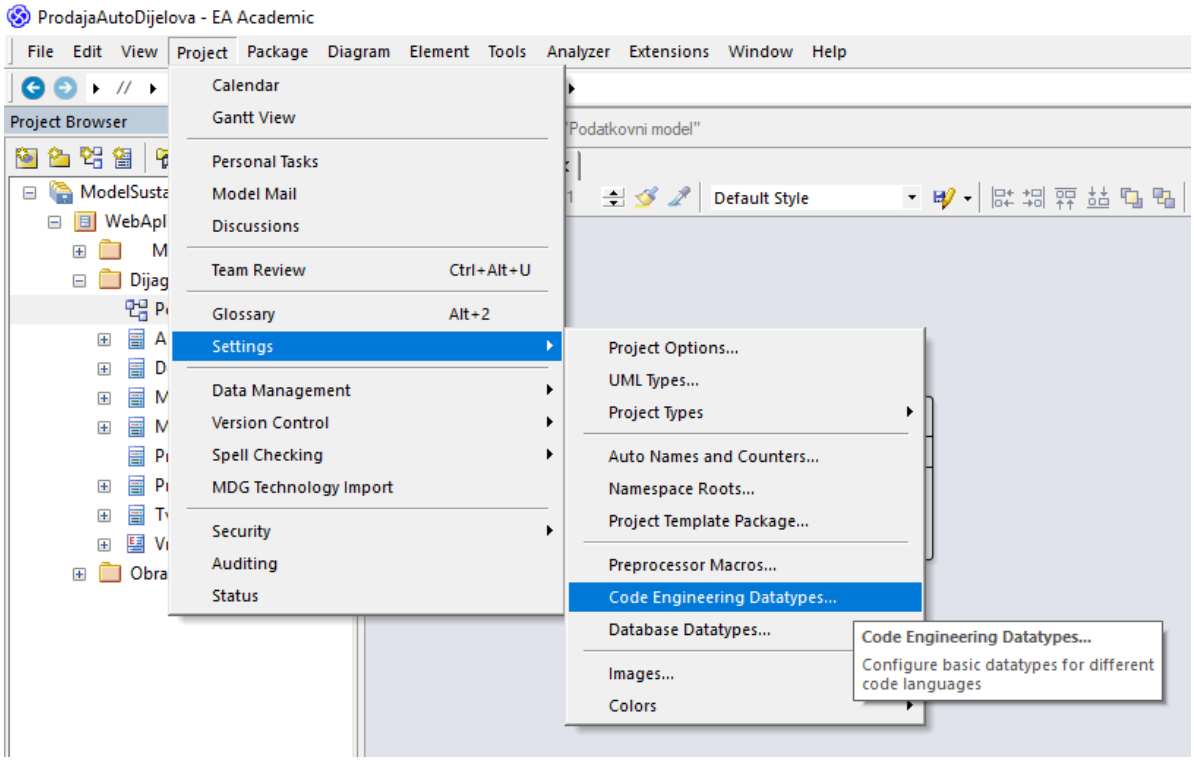


Slika 4.2.49 Ponovno postavljanje jezika generiranja

U upravljačkom prozoru potrebno je kvačicama označiti odgovarajuće opcije kako bi se iznova generirao ciljni programski kôd.

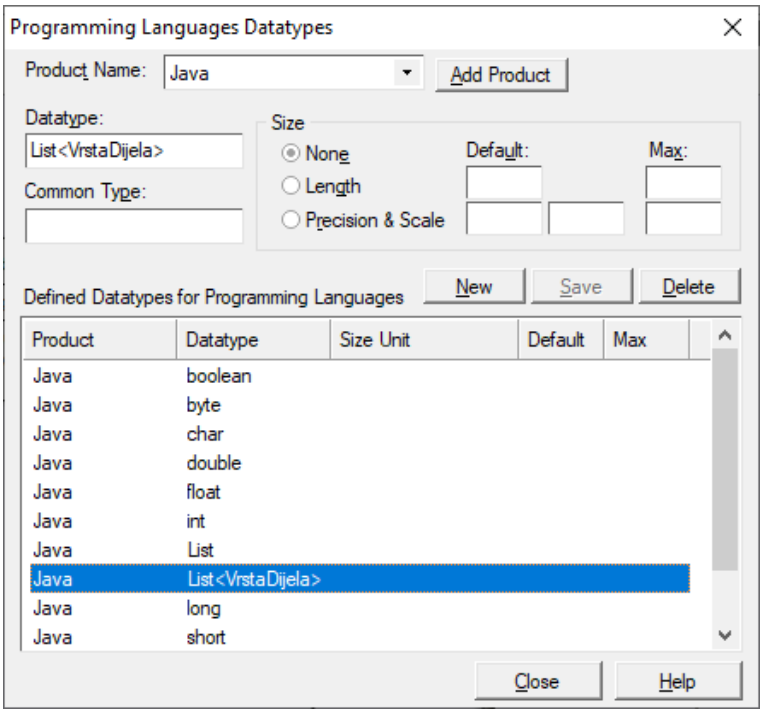
Definicija vrste podataka

Novu vrstu podataka koja se smatra svojstvena određenoj programskoj tehnologiji a ne spada u početni skup generičkih vrsta podataka pretpostavljenih unutar alata *Enterprise Architect* izborom opcije *Code Engineering Datatypes* ili *Coding Datatypes*, Slika 4.2.50.



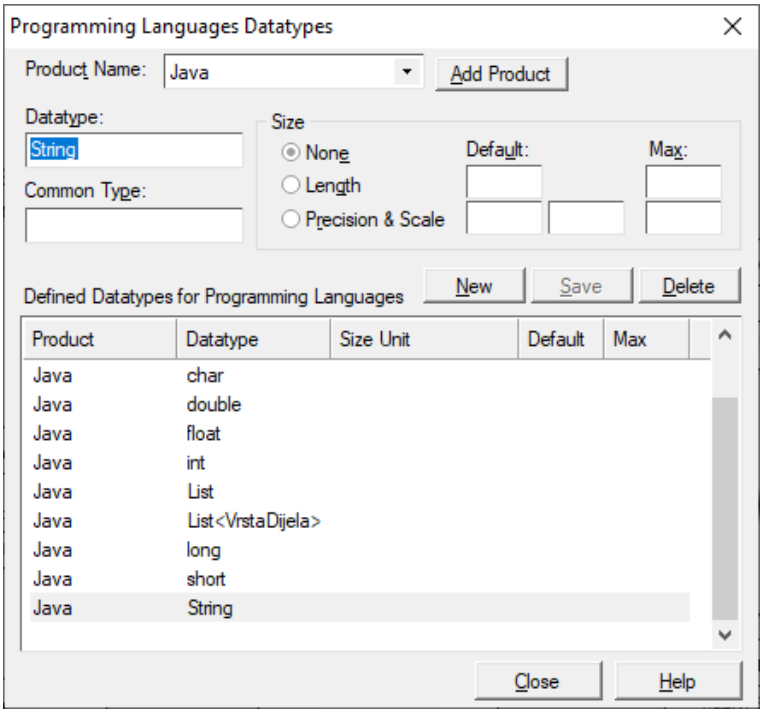
Slika 4.2.50 Izbor definicije nove vrste podataka

Unutar prozora *Programming Language Datatypes* moguće je za izabrani programski jezik (stavka *Product Name*) dodavati i brisati novodefinirane vrste podataka koji će potom biti na raspolaganju pri definiciji atributa razreda, Slika 4.2.51.



Slika 4.2.51 Definicija nove vrste podataka

Tako se u primjeru na slici definira vrsta podataka *String* za programski jezik Java s obzirom na to da ga u početnom skupu podataka nije bilo, Slika 4.2.52.

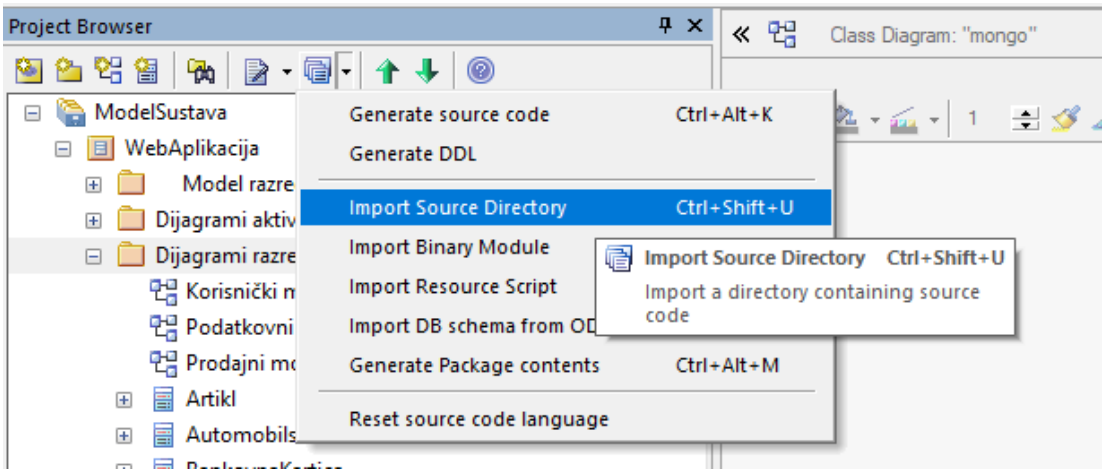


Slika 4.2.52 Dodavanje vrste podataka String za programski jezik Java

Nakon toga se kod definicije atributa i metoda mogu koristiti i novodefinirani tipovi podataka.

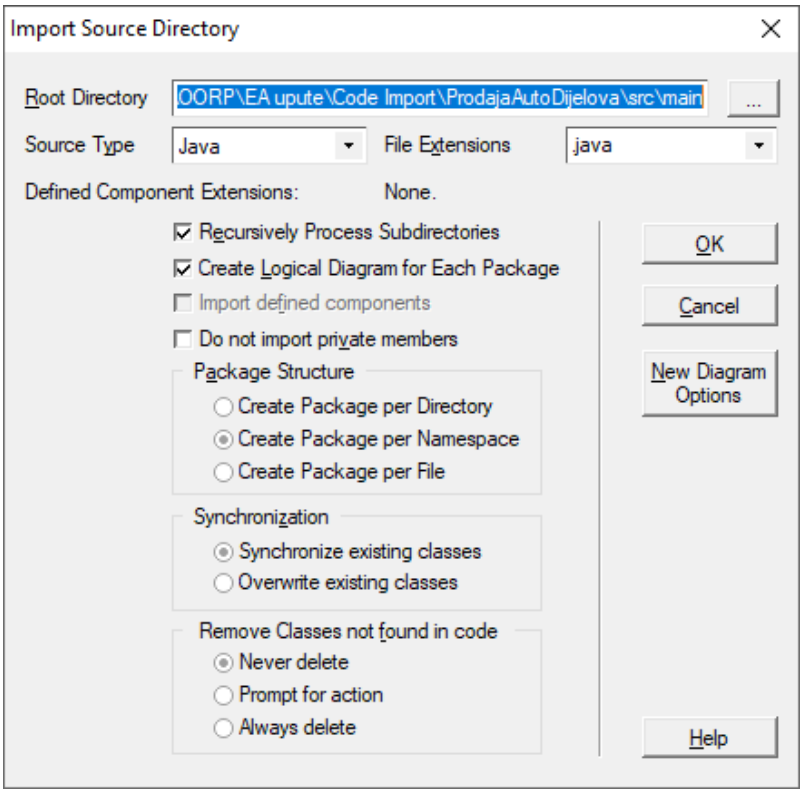
Uvoz izvornog koda

Funkcionalnost se poziva izborom opcije *Import Source Directory*, Slika 4.2.53.



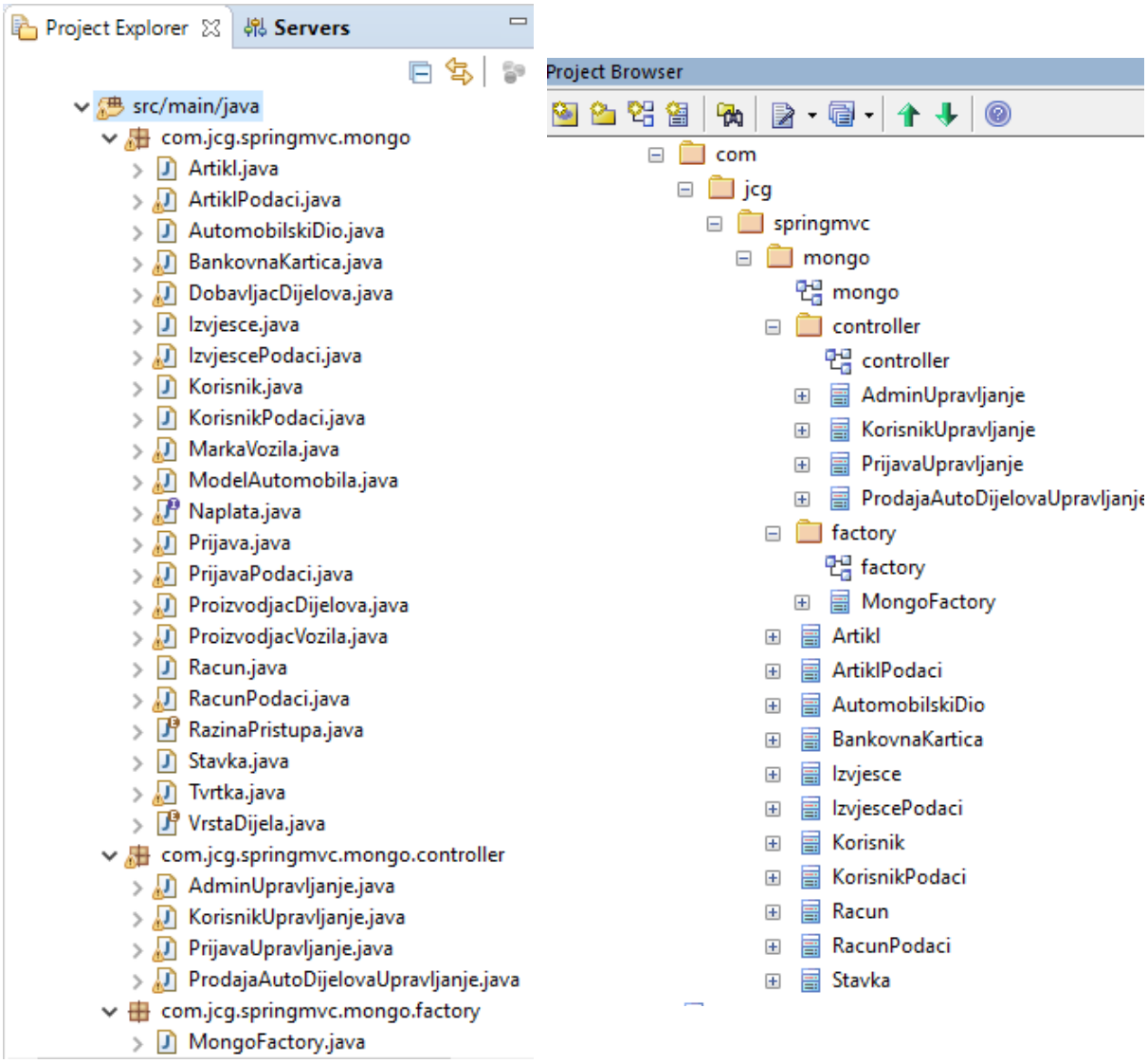
Slika 4.2.53 Pokretanje uvoza izvornog koda

Odabirom kazala iz kojeg se uvoz radi (pod stavkom *Root Directory*) i vrstom programskog koda (pod stavkom *Source Type*) uz pregled ostalih postavki pokreće se uvoz programskog koda, Slika 4.2.54.



Slika 4.2.54 Postavljanje uvjeta uvoza programskog koda

Slika 4.2.55a prikazuje pregled razvijenog programskog koda koji se nalazi unutar kazala koje se stavlja na uvoz.



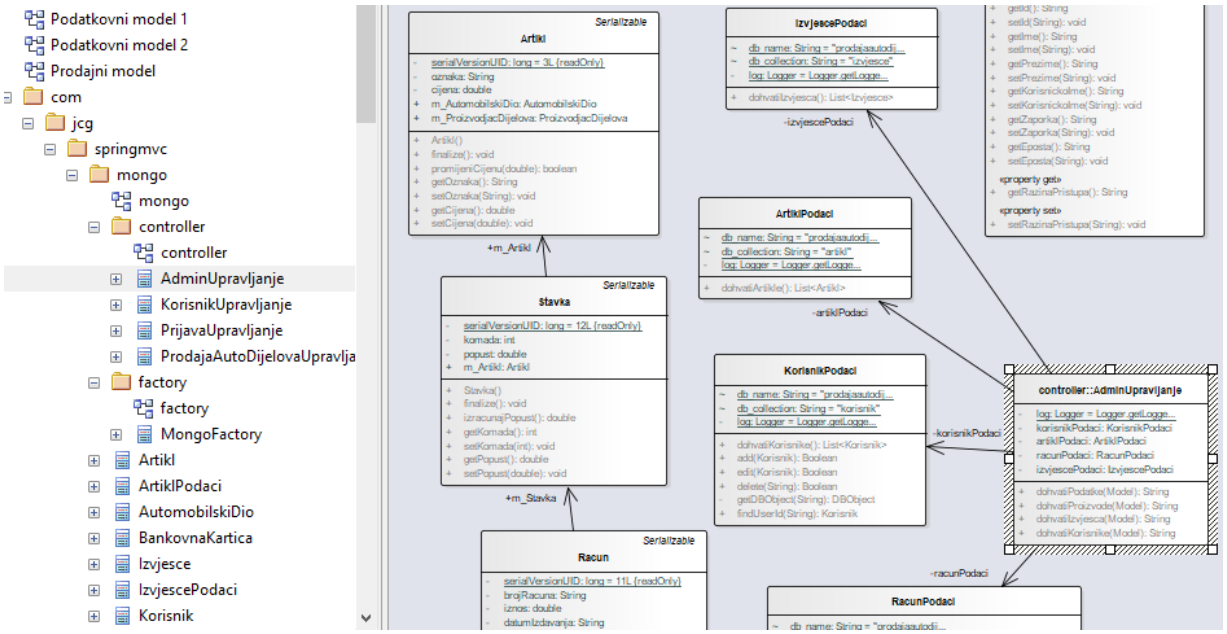
(a) *(b)*

Slika 4.2.55 Usporedba strukture datoteka izvornog koda i stvorenih razreda

Izvorni kôd se sastoji od tri paketa: 'com.jcg.springmvc.mongo', 'com.jcg.springmvc.mongo.controller' i 'com.jcg.spring.mvc.mongo.factory'. Nakon uvoza u alat *Enterprise Architect* unutar strukture paketa u koji je uvoz napravljen uspostavlja se odgovarajuća struktura dijagrama i razreda, Slika 4.2.55b.

Kombiniranje različitih paketa

Ako se u dijagram podpaketa 'mongo' prebaci razred 'AdminUpravljanje' iz podpaketa 'controller' automatski će se pojaviti i veze koje logički postoje između tih kombiniranih razreda iz različitih paketa, Slika 4.2.56.

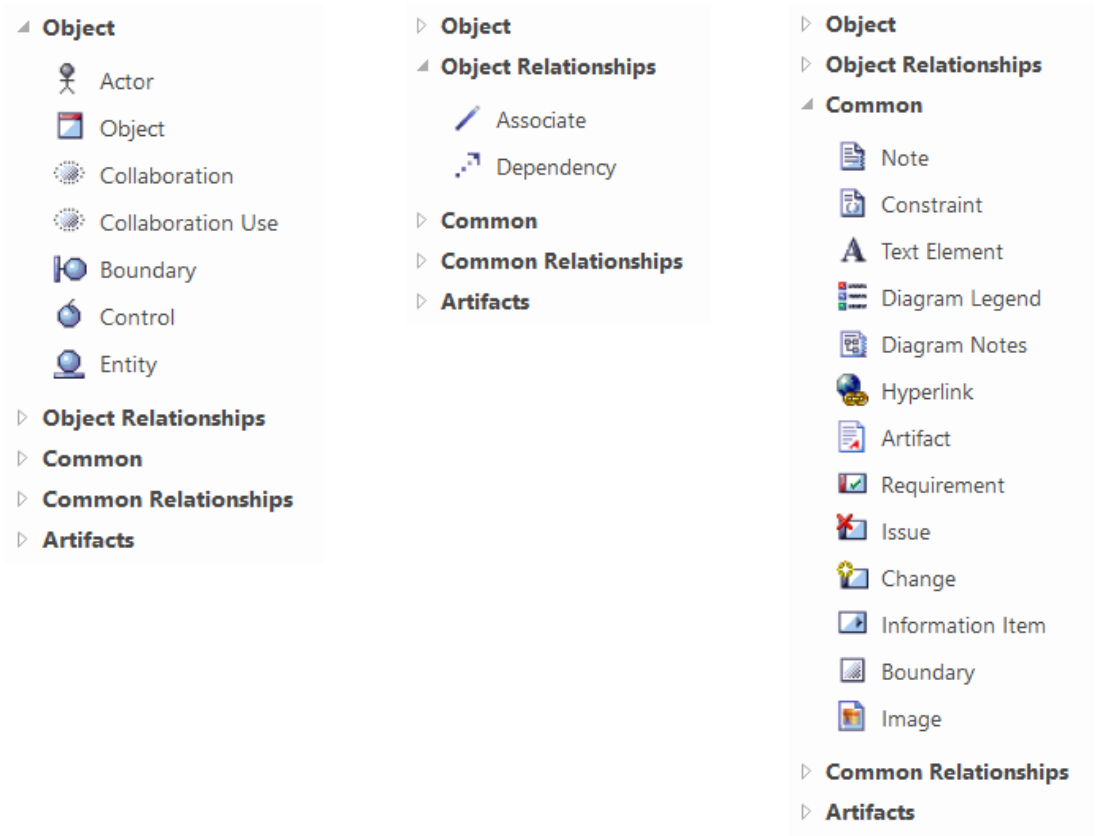


Slika 4.2.56 Kombiniranje razreda različitih paketa

Razred 'AdminUpravljanje' upravlja zahtjevima upravljanja administratorskih ovlasti te ima reference na razrede dohvata podataka o artiklima (razred 'ArtiklPodaci'), računima (razred 'RacunPodaci'), korisnicima (razred 'KorisnikPodaci') i izvješćima (razred 'IzvjescePodaci').

Dijagram objekata

Slika 4.2.57 prikazuje glavne alatne okvire dijagrama objekata. Alatni okvir *Object* sadrži različite vrste objekata koji mogu biti uključeni u dijagram, alatni okvir *Object Relationships* vrste veza koje opisuju odnose između objekata, a alatni okvir *Common* uobičajeno sadrži elemente koji su općeniti za različite vrste dijagrama.

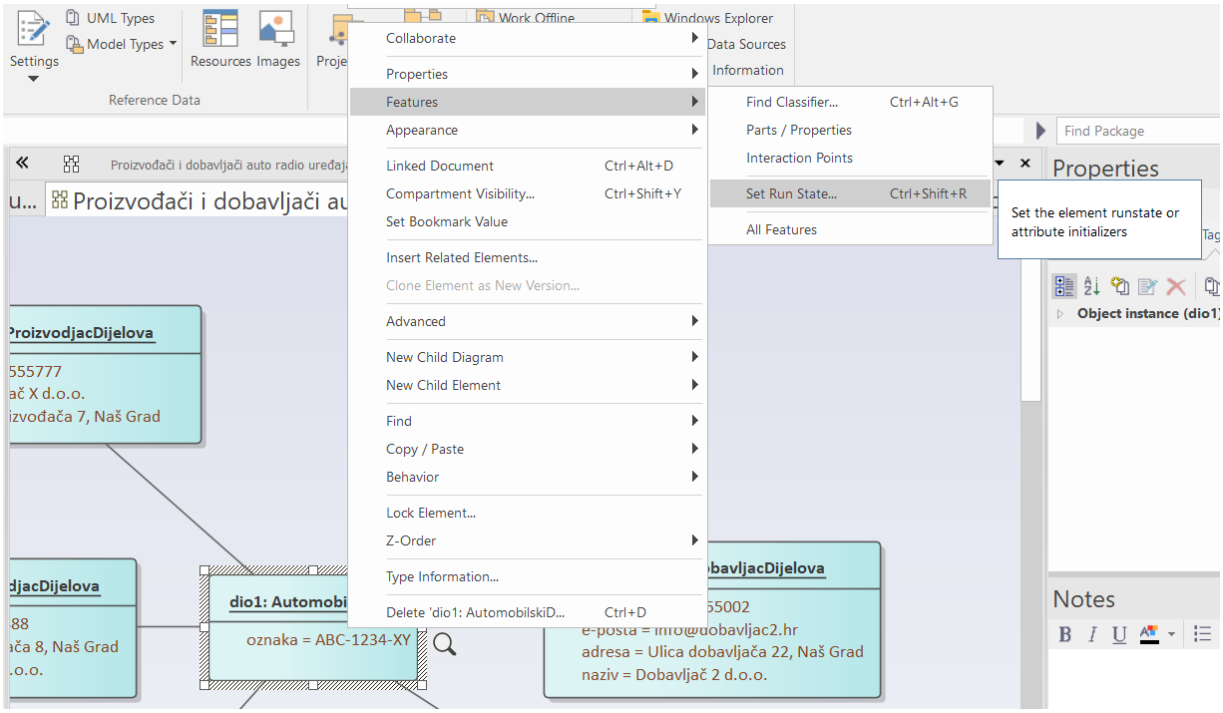


(a) *(b)* *(c)*

Slika 4.2.57 Alatni okviri dijagrama objekata

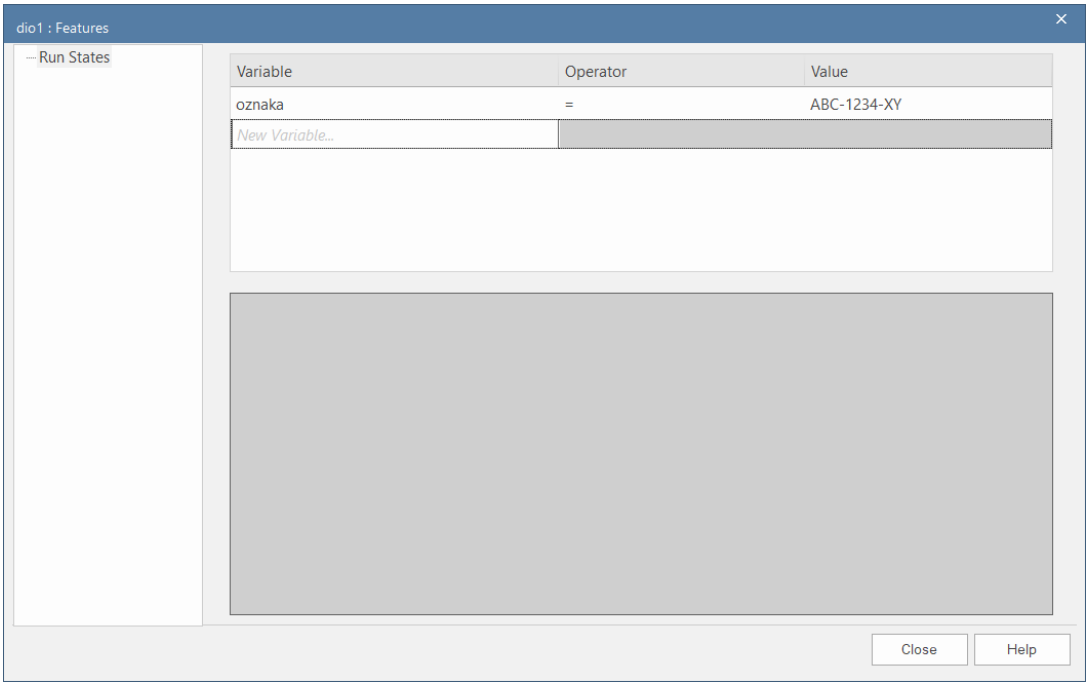
Opis stanja objekta

Opis stanja objekta definira se kao *Run Time State*, Slika 4.2.58.



Slika 4.2.58 Poziv opisa stanja objekta

Pozivom te funkcionalnosti može se definirati stanje objekta prema atributima razreda iz kojeg je objekt nastao, Slika 4.2.59.

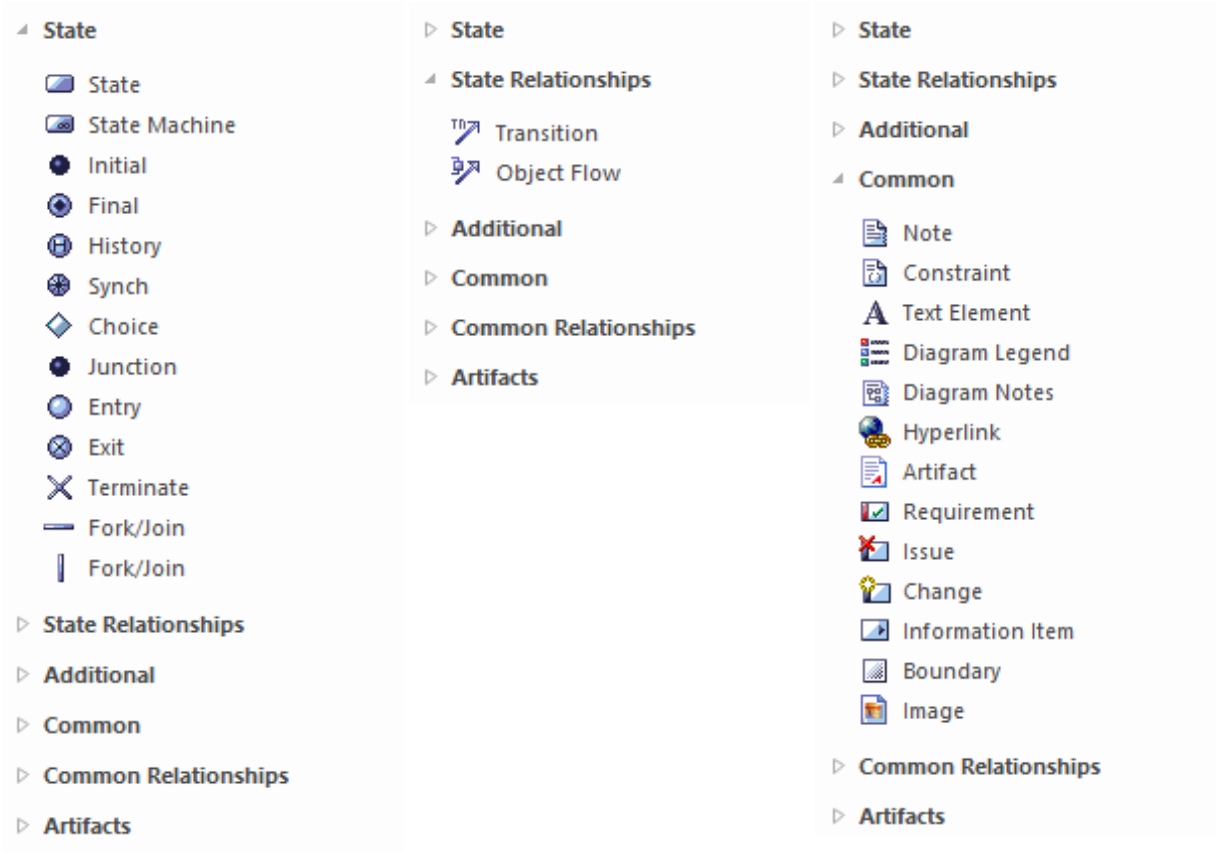


Slika 4.2.59 Opis stanja objekta (Run States)

Primjer jednostavnog dijagrama stanja prethodno je prikazan u uvodnom primjeru, Slika 2.8.1.

Dijagram stanja

Slika 4.2.60 prikazuje glavne alatne okvire pri izradi dijagrama stanja.

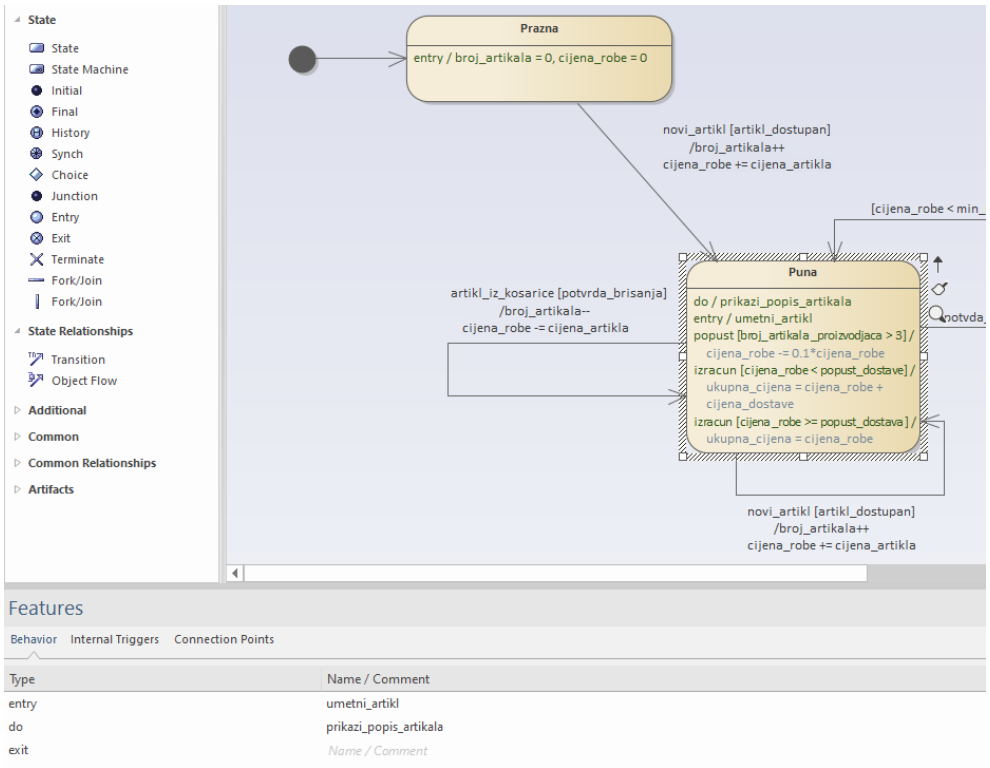


Slika 4.2.60 Alatni okviri dijagrama stanja

Alatni okvir *State* sadrži elemente prikaza stanja i različitih vrsta čvorova koji mogu biti prisutni u dijagramu stanja a alatni okvir *State Relationships* elemente prikaza upravljačkog toka i toka objekata. Alatni okvir *Common* uobičajeno sadrži elemente koji su opće korišteni u različitim vrstama dijagramima.

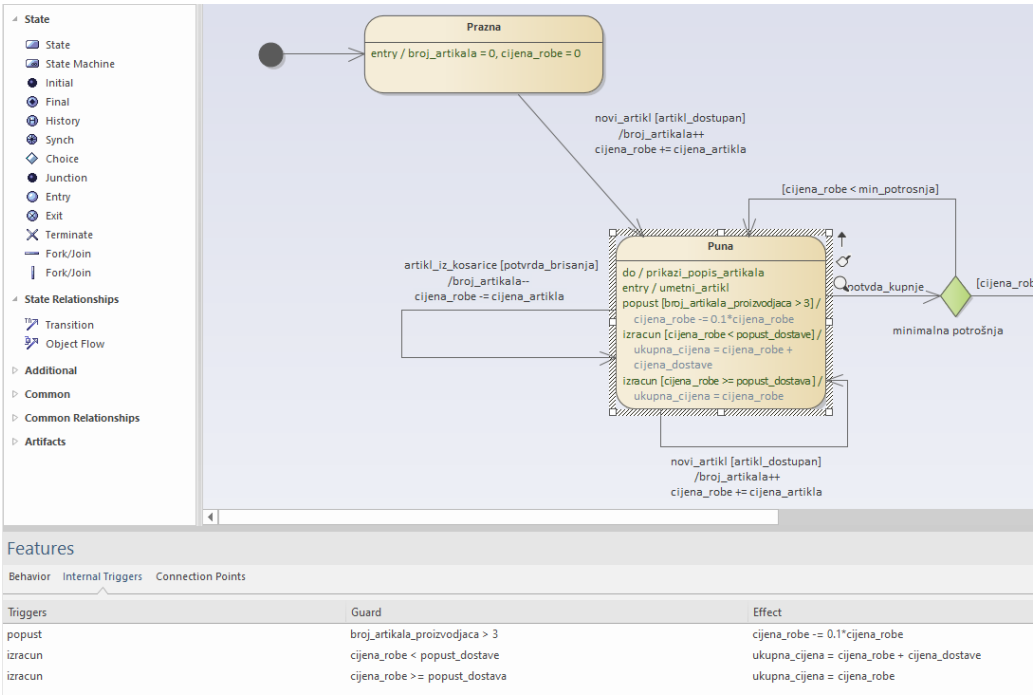
Opis stanja

Slika 4.2.61 i Slika 4.2.62 prikazuju primjer definicije stanja unutar dijagrama. U alatnom okviru *Features* navode se ponašanja pod karticom *Behavior* i unutarnje okidače pod karticom *Internal Triggers*.
Pod „ponašanjima“ se misli da akcije pri ulazu u stanje (vrsta *entry*), za vrijeme boravka u stanju (vrsta *do*) i prilikom izlaska iz stanja (vrsta *exit*), Slika 4.2.61.



Slika 4.2.61 Opis unutarnjeg ponašanja nekog stanja

Pod „unutarnjim okidačima“ misli se na sve ostale akcije (*Effect*) koje se događaju pod određenim uvjetima (*Guard*), Slika 4.2.62.

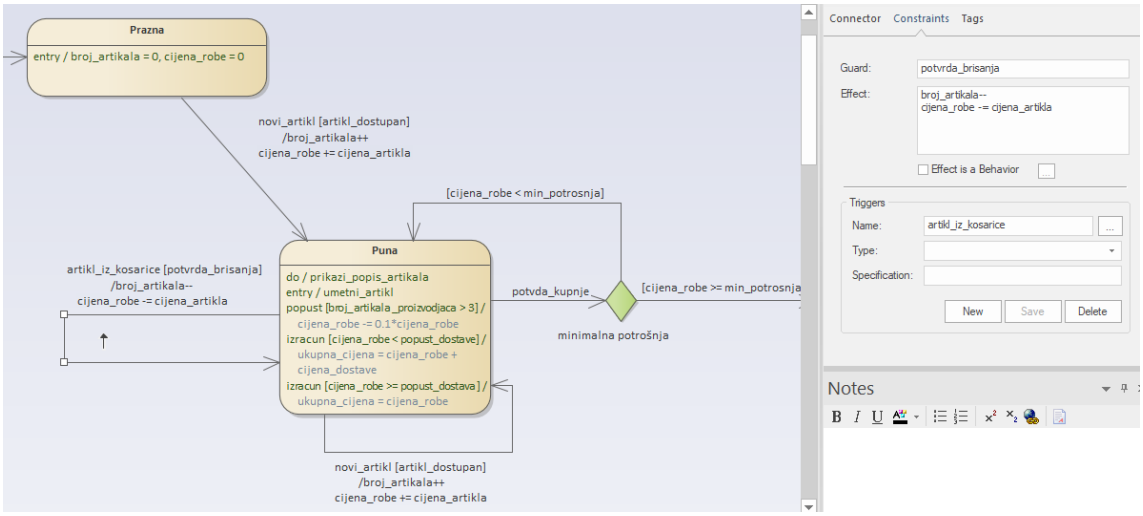


Slika 4.2.62 Opis unutarnjih okidača nekog stanja

Moguće je definirati proizvoljan broj „okidača“ s odgovarajućim nazivima, uvjetima i akcijama koji će u prikazu biti obuhvaćeni unutar prikaza stanja čije ponašanje opisuju.

Opis prijelaza

Prijelazi se opisuju nazivom događaja koji ih je pokrenuo (*Name*), uvjetom koji mora biti pri tome ispunjen (*Guard*) i akcijama koje se pri tome izvršavaju (*Effect*).

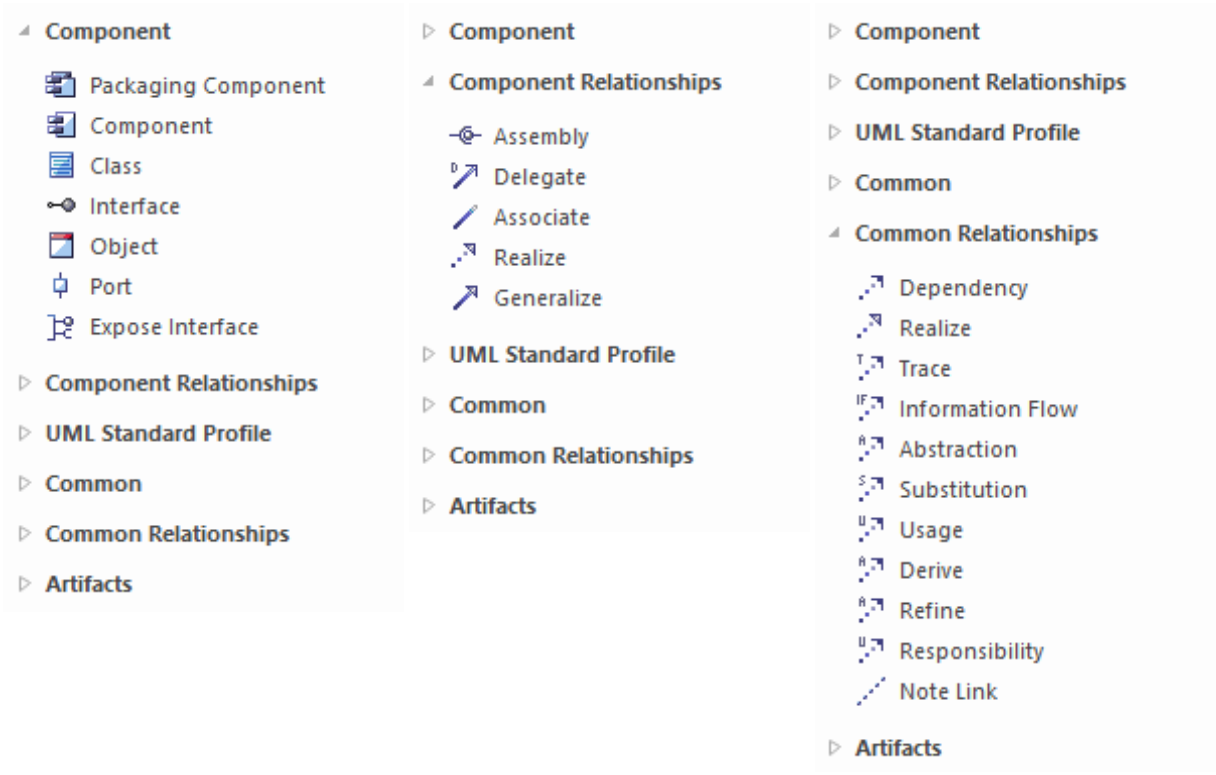


Slika 4.2.63 Opis prijelaza između stanja

U prikazu prijelaza na dijagramu opis prijelaza će imati uobičajeni oblik notacije UML-dijagrama stanja: *naziv [uvjet]/akcije*.

Dijagram komponenti

Alatni okvir *Toolbox* koji se po otvaranju dijagrama pojavljuje sastoji se od nekoliko dijelova od kojih su za dijagram komponenti temeljni oni pod naslovima *Component* i *Component Relationships*, Slika 4.2.64. Prvi (a) služi za umetanje komponenti i drugih elemenata dijagrama, a drugi (b) za umetanje veza za opis odnosa između postavljenih komponenti.



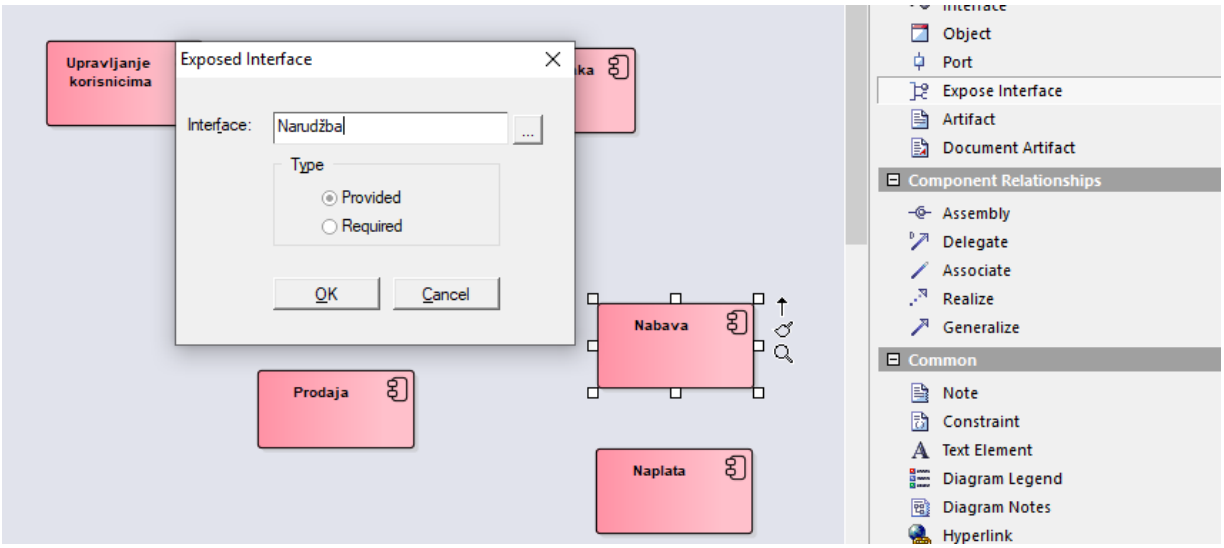
(a) (b) (c)

Slika 4.2.64 Alatni okviri dijagrama komponenti

Prema sadržaju alatnog okvira *Component* u dijagram komponenti se mogu postavljati vrste elemenata od razine prikaza paketa (*Package*) i paketnih komponenti (*Packaging Component*) za prikaz skupova komponenti sadržanih unutar nekog imenovanog prostora do razine prikaza „artefakata“, datoteka i dokumenata koji su rezultat krajnjeg ostvarenja, Slika 4.2.64a.

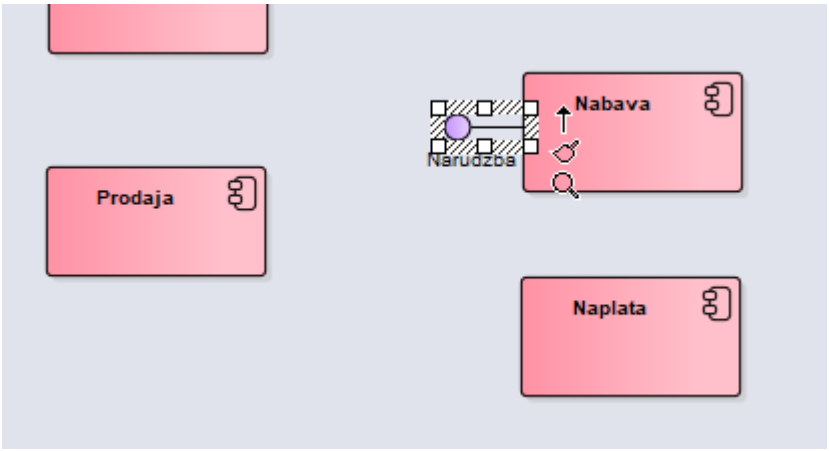
Definicija sučelja

Definiranje sučelja za neku komponentu pokreće se izborom stavke *Expose Interface* iz alatnog okvira *Component*, Slika 4.2.65.



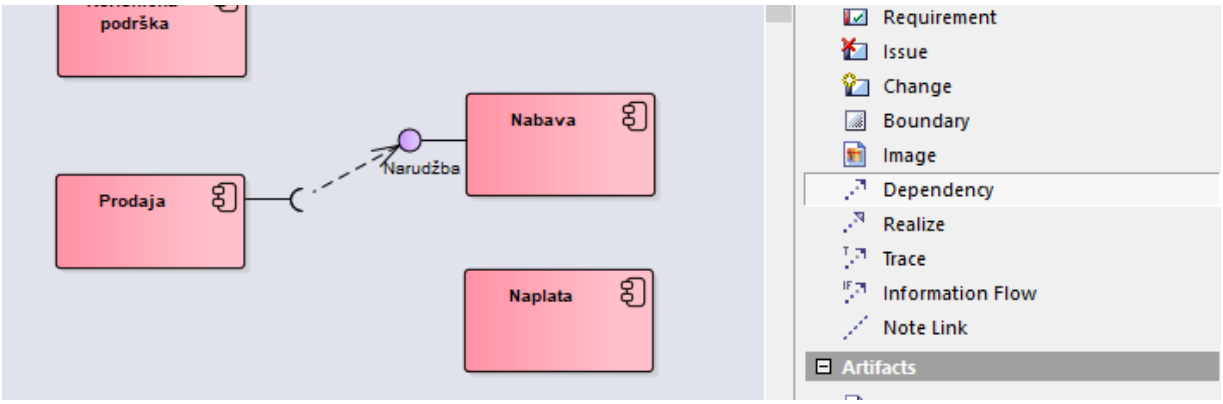
Slika 4.2.65 Definicija sučelja na dijagramu komponenti

Ako se definira realizirano ili ponuđeno sučelje ono se na dijagramu prikazuje kao izloženo s vezom na komponentu s kružićem na kraju, Slika 4.2.66.



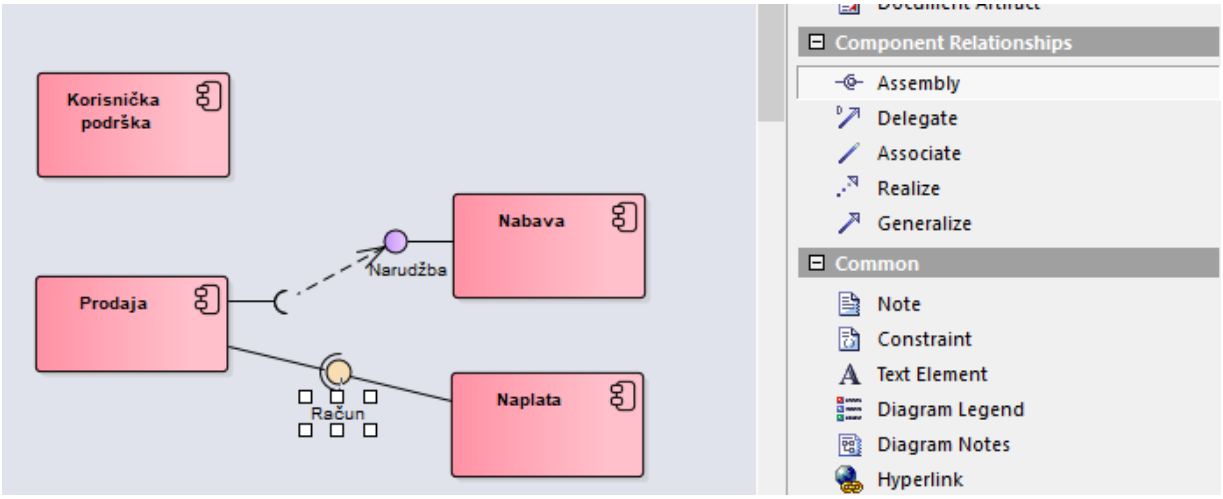
Slika 4.2.66 Prikaz sučelja komponente

Zahtijevano sučelje se prikazuje kao izložen polukružić. Ako npr. jedna komponenta ima ponuđeno sučelje koje koristi druga komponenta sa svojim zahtijevanim sučeljem takva veza se općenito definira kao veza ovisnosti između zahtijevanog i ponuđenog sučelja, Slika 4.2.67. Veza ovisnosti se dohvaća stavkom *Dependency* iz alatnog okvira *Common Relationships*.



Slika 4.2.67 Prikaz povezivanja zahtijevanog i ponuđenog sučelja komponenti

Moguć je i jednostavniji ili sažetiji prikaz povezanih sučelja na način da se iz alatnog okvira *Component Relationships* izabere stavka *Assembly* kao jedinstvena vrsta veze između komponenti. Povlačenjem takve veze od komponente koja zahtijeva prema komponenti koja nudi sučelje dobiva se prikaz na način da se ponuđeno sučelje svojim simbolom uglavljuje u simbol zahtijevanog sučelja, na taj način naznačujući da realizira zahtijevano sučelje, Slika 4.2.68.

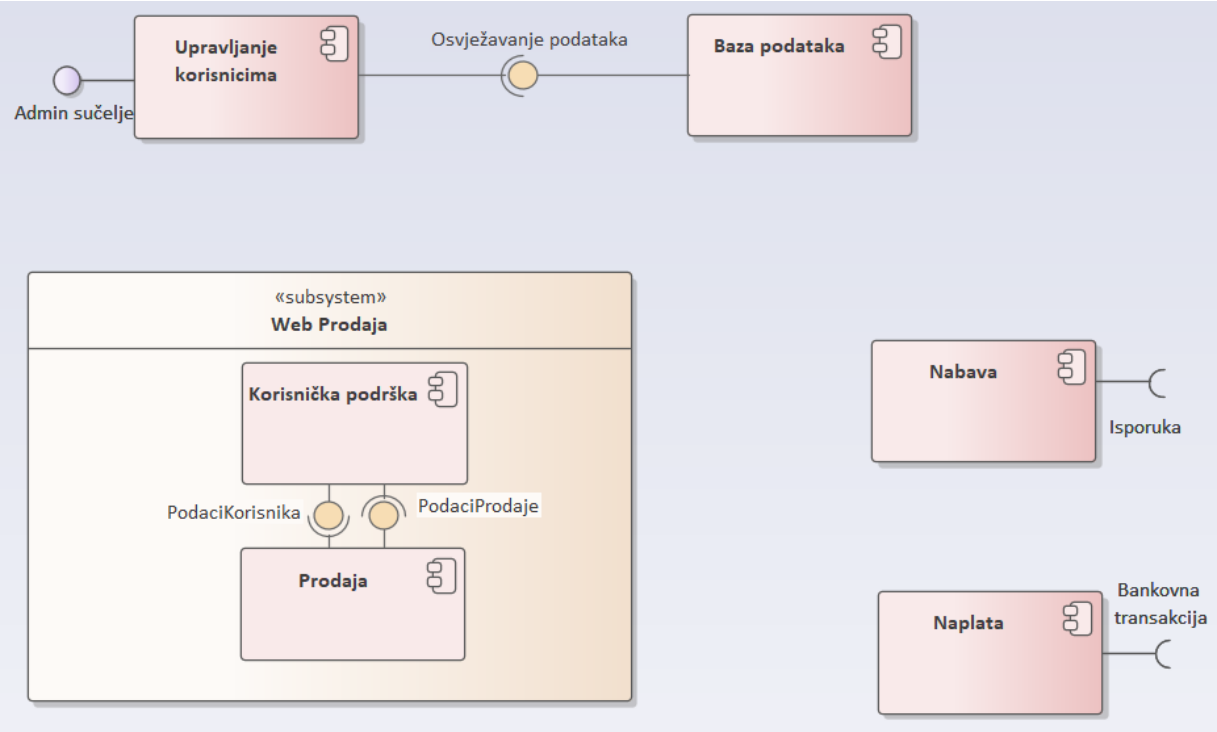


Slika 4.2.68 Pojednostavljeni prikaz povezivanja zahtijevanog i ponuđenog sučelja komponenti

U ovom primjeru na taj način je prikazan odnos komponenti 'Prodaja' i 'Naplata' gdje komponenta 'Prodaja' zahtijeva sučelje za naplatu računa. Komponenta 'Naplata' ostvaruje takvo sučelje naziva 'Račun' kako mu je dano prilikom definicije.

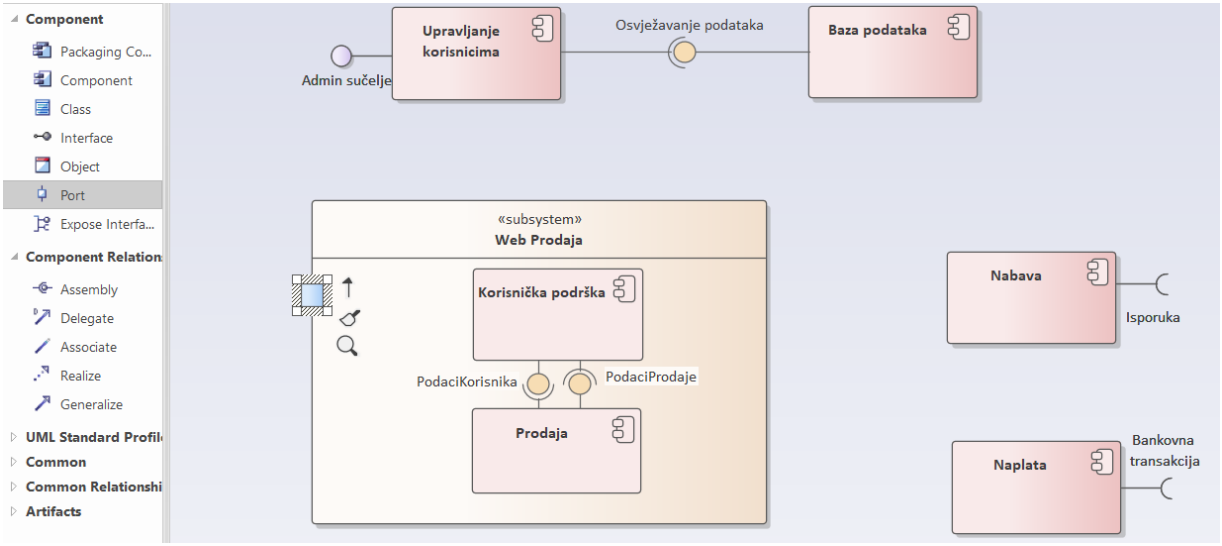
Umetanje složenijih komponenti

Složenije komponente se grade i u sebi mogu sadržavati manje komponente i mogu predstavljati dijelove sustava ili podsustave. Izborom jednostavnog grafičkog okvira pod stavkom *Boundary* u alatnom okviru *Common* i postavljanjem postojećih komponenti unutar tog okvira uokviruju se složenije komponente. Primjerice, podsustav web prodaje sastoji se od komponente 'Prodaja' i komponente 'Korisnička podrška', Slika 4.2.69. S obzirom na to da to u prikazu postaje odvojeni zatvoreni podsustav moguće je naznačiti da postoje odabrane pristupne točke ili vrata (engl. *port*) putem kojih se pristupa i komunicira s takvim podsustavom.



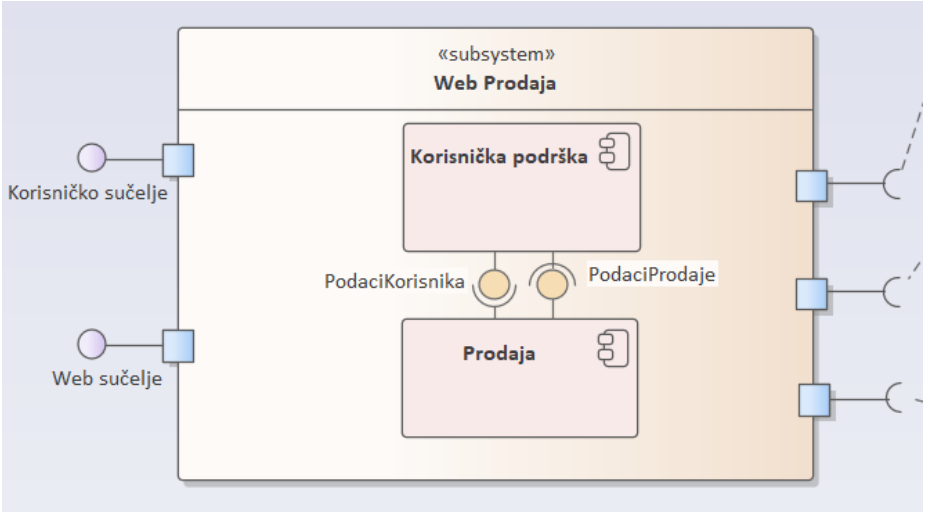
Slika 4.2.69 Stvaranje podsustava na dijagramu komponenti

Izborom stavke *Port* iz alatnog okvira *Component* postavlja se pristupna točka ili **priključnica** na granici podsustava, Slika 4.2.70. Putem takve pristupne točke ostvarit će se sučelje podsustava s okolnim komponentama, tj. pristup postojećim sučeljima i komunikacija sa složenijim komponentama će se provoditi kroz takvu pristupnu točku. Slika 2.10.6 prikazuje da komponente 'Prodaja' i 'Korisnička podrška' imaju ukupno pet definiranih različitih sučelja sa svojom okolinom pa će se umetnuti taj broj pristupnih točaka da bi se omogućio pristup unutarnjim sučeljima komponente, Slika 4.2.71.



Slika 4.2.70 Definicija pristupne točke podsustava na dijagramu komponenti

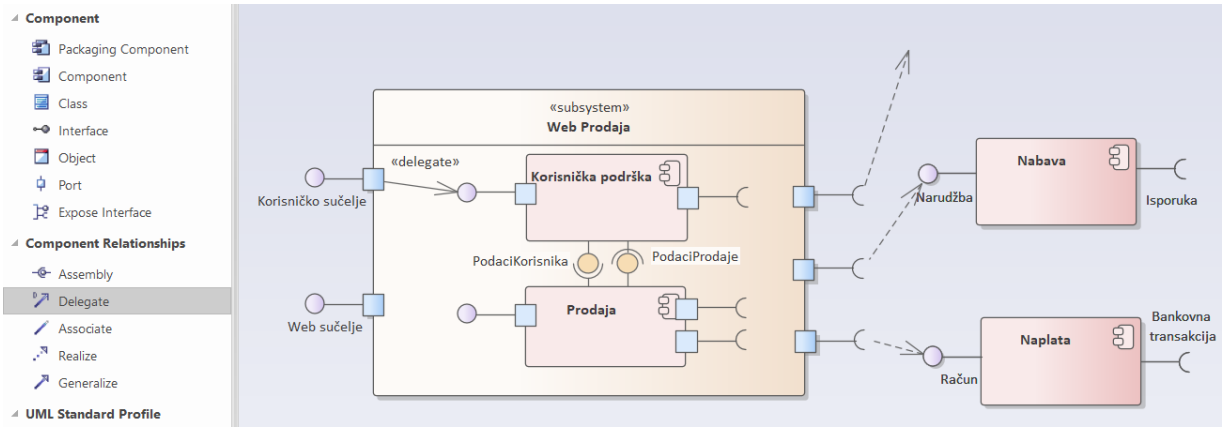
Pristupne točke označavaju mjesta komunikacije vanjskog svijeta sa zatvorenim podsustavom. Slika 4.2.71 prikazuje koja sučelja su definirana prema vanjskom svijetu za središnji podsustav web-prodaje unutar cijelog sustava prodaje auto-dijelova, ali nije precizno prikazana unutarnja povezanost i struktura podsustava te djelomično izgleda kao „crna kutija“. Ako se želi prikazati koja je komponenta unutar tog podsustava odgovorna za ostvarivanje ili zahtijevanje pojedinog sučelja potrebno je naznačiti i unutarnja sučelja komponenti podsustava i njihovu povezanost s vanjskim sučeljima podsustava.



Slika 4.2.71 Prikaz pristupnih točaka s vanjskim sučeljima podsustava na dijagramu komponenti

Kako je iz prethodno oblikovanog dijagrama razvidno (Slika 2.10.6), putem vanjskog sučelja 'Korisničko sučelje' pristupa se komponenti 'Korisnička podrška'. Zato je potrebno redefinirati

unutarnje sučelje komponente 'Korisnička podrška' te ga povezati s vanjskim sučeljem 'Korisničko sučelje' izborom stavke *Delegate* kao posebne vrste veze za to, Slika 4.2.72.

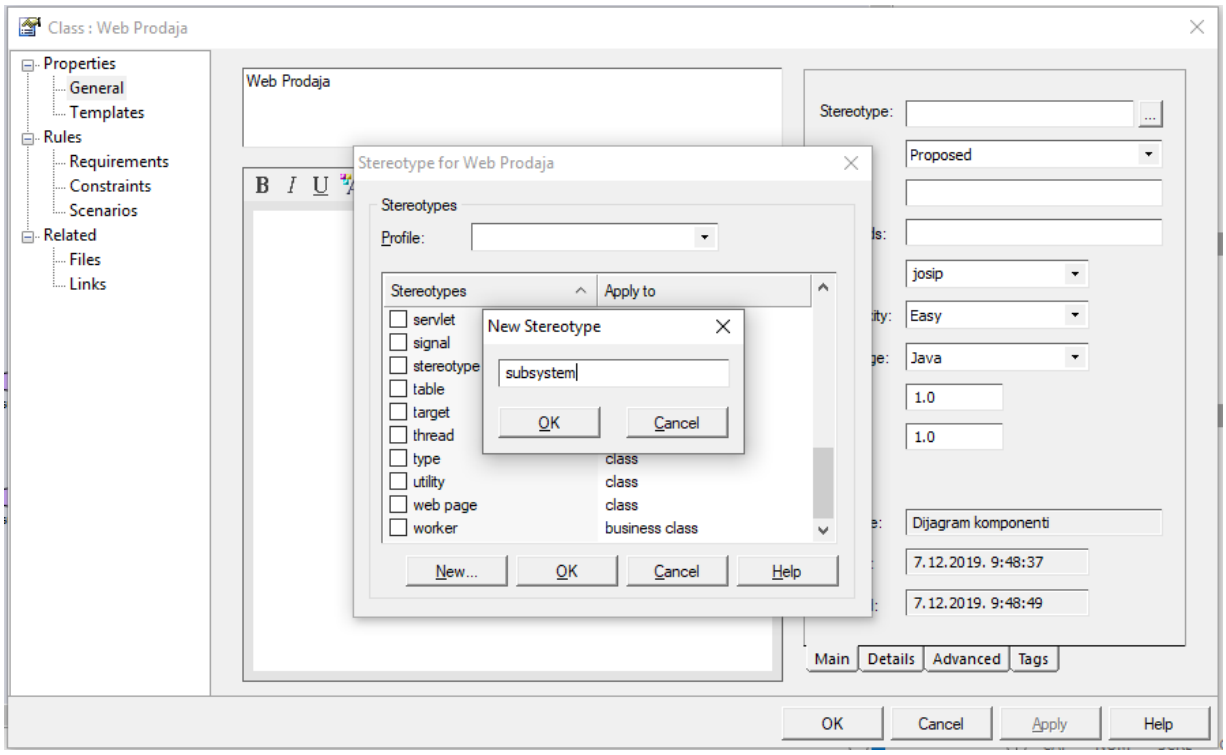


Slika 4.2.72 Povezivanje sučelja podsustava na dijagramu komponenti

Primjer konačnog oblika dijagrama s unutarnjim i vanjskim sučeljima prikazan je u uvodno elaboriranom primjeru sustava, Slika 2.10.8.

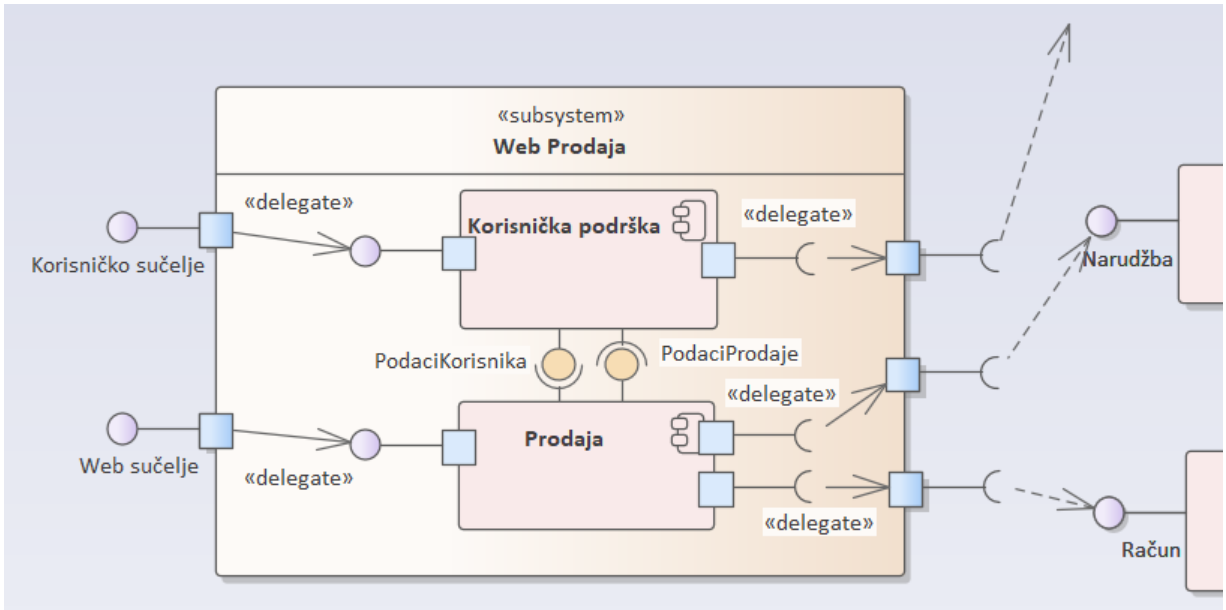
Definicija stereotipa

Neka komponenta se može sažeto dodatno opisati na način da se označi nekim stereotipom kao posebnom vrstom notacije. Stereotip se može definirati za skoro bilo koji element dijagrama pa tako i za složenu komponentu ili izdvojeni podsustav. Izborom stavke izbornika *Stereotype:* moguće je izabrati ili definirati stereotip komponente. Primjerice, za podsustav 'Web Prodaja' se definira stereotip <<*subsystem*>> da bi se posebno naznačila narav izdvojenog podsustava, Slika 4.2.73.



Slika 4.2.73 Definicija novog stereotipa na dijagramu komponenti

Ako se za podsustav web-prodaje definiraju sve pristupne točke, sučelja, povezivanja vanjskih i unutarnjih sučelja i njihova povezivanja s vanjskim komponentama te potrebni stereotipovi, podsustav 'Web Prodaja' zadobiva novi izgled unutar dijagrama komponenti, Slika 4.2.74.

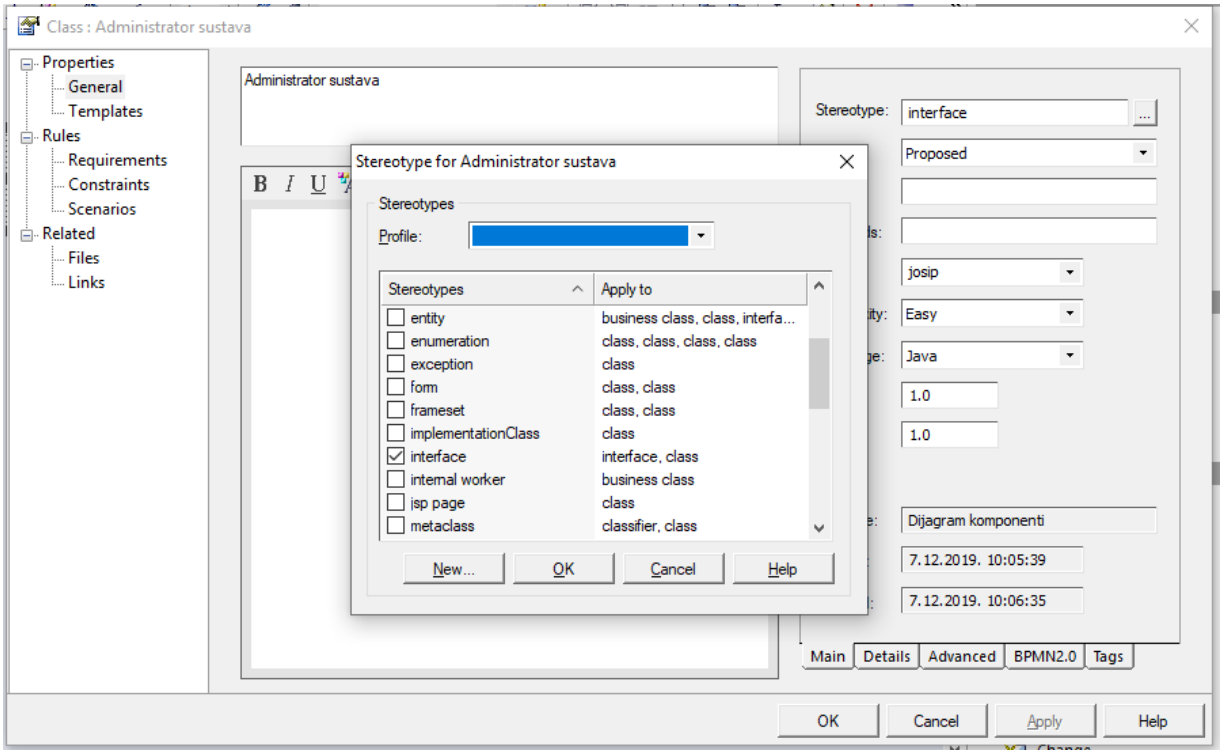


Slika 4.2.74 Upotpunjeni prikaz podsustava 'Web Prodaja' na dijagramu komponenti

Osim stereotipa <<*subsystem*>> za označavanje podsustava upotrijebljen je i stereotip <<*delegate*>> kao oznaka povezivanja unutarnjih i vanjskih sučelja.

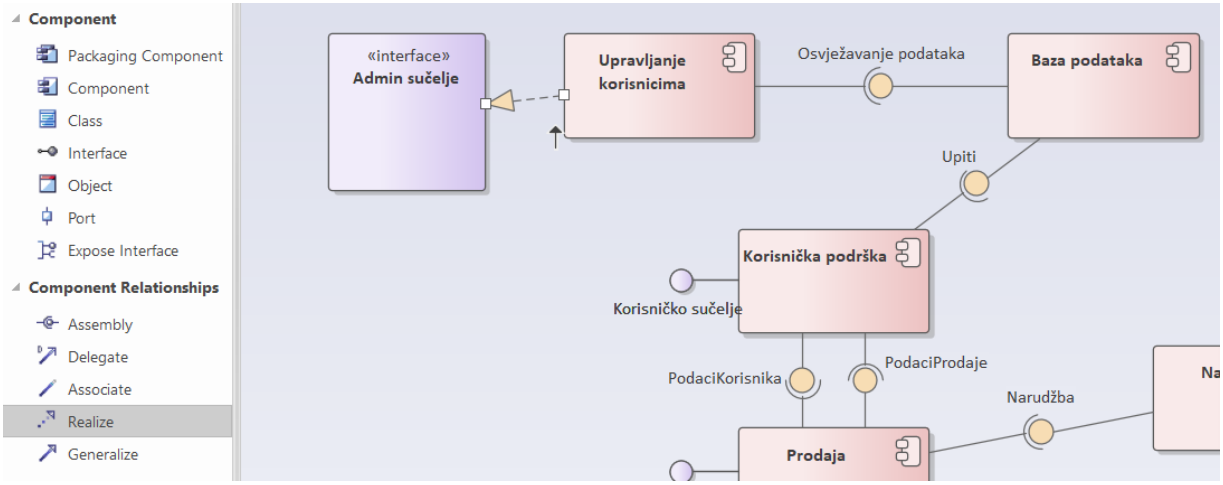
Uporaba proširene notacije sučelja

Prošireni oblik sučelja umeće se kao novi element dijagrama vrste *Interface*, ili vrste *Class* uz dodatnu definiciju stereotipa *interface*, iz alatnog okvira *Component*, Slika 4.2.75.



Slika 4.2.75 Postavljanje stereotipa za prošireni prikaz sučelja na dijagramu komponenti

Naprimjer, ako se prethodno definirano 'Admin sučelje' (Slika 2.10.6) želi prikazati u proširenoj notaciji onda se definira element dijagrama vrste *Interface*, ili vrste *Class* uz stereotip *interface*, te se postavlja na dijagramu umjesto sažetog oblika elementa ostvarenog sučelja, Slika 4.2.76. Novodefiniranom sučelju je postavljen naziv 'Administrator sustava' te ga ostvaruje komponenta 'Upravljanje korisnicima', kao što je bilo u prethodnom slučaju dijagrama.

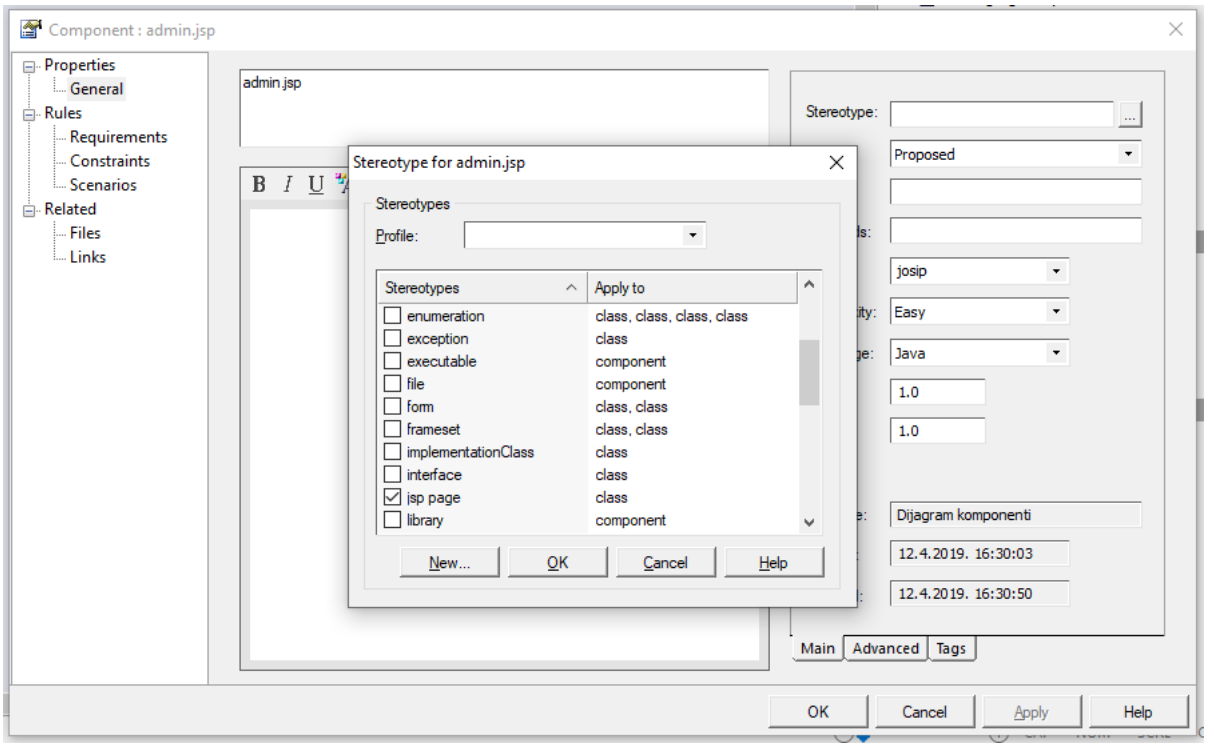


Slika 4.2.76 Proširenи prikaz sučelja na dijagramu komponenti

Od komponente 'Upravljanje korisnicima' prema sučelju 'Administrator sustava' se povlači veza vrste *Realize* čime se upravo govori da ta komponenta ostvaruje navedeno sučelje. Na sličan način se i zahtijevano sučelje prikazuje u proširenom obliku gdje se veza vrste 'Dependency' povlači od komponente koja traži neko sučelje prema samom sučelju. Komponenta u tom slučaju „ovisi“ o nekom sučelju.

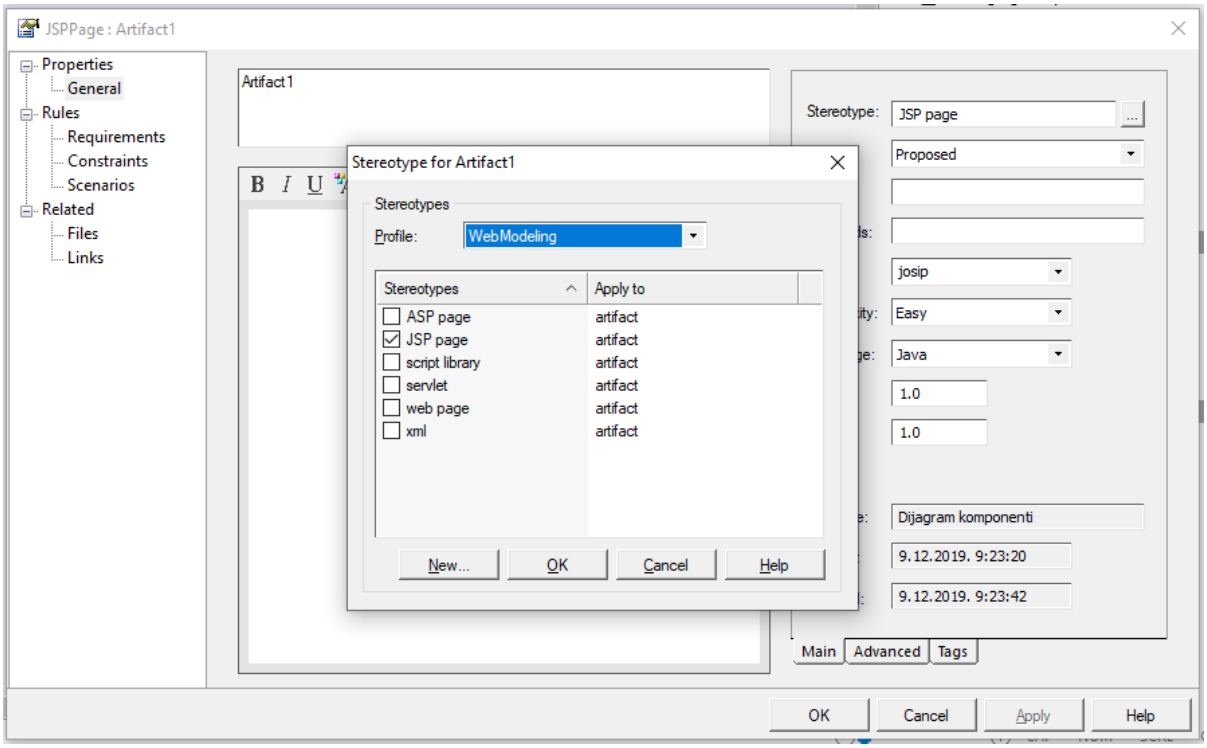
Umetanje artefakata

Ako se komponente u dijagramu gledaju i prikazuju na razini artefakata, umjesto stavke *Component* iz istoimenog alatnog okvira, može se izabrati stavka *Artifact* iz istog okvira. S obzirom na mnoštvo postojećih različitih formata i namjena datoteka, uputno je definirati stereotip za svaku od njih. To se može napraviti i ako se kao polazna komponenta definicije uzme vrsta komponente *Component* (Slika 4.2.77) i ako se kao polazna točka definicije uzme vrsta komponente *Artifact* (Slika 4.2.78).



Slika 4.2.77 Definiranje stereotipa za komponentu vrste Component

Razlika je u skupu predefiniranih stereotipova za te dvije vrste komponenti, a po želji se može definirati i novi stereotip.

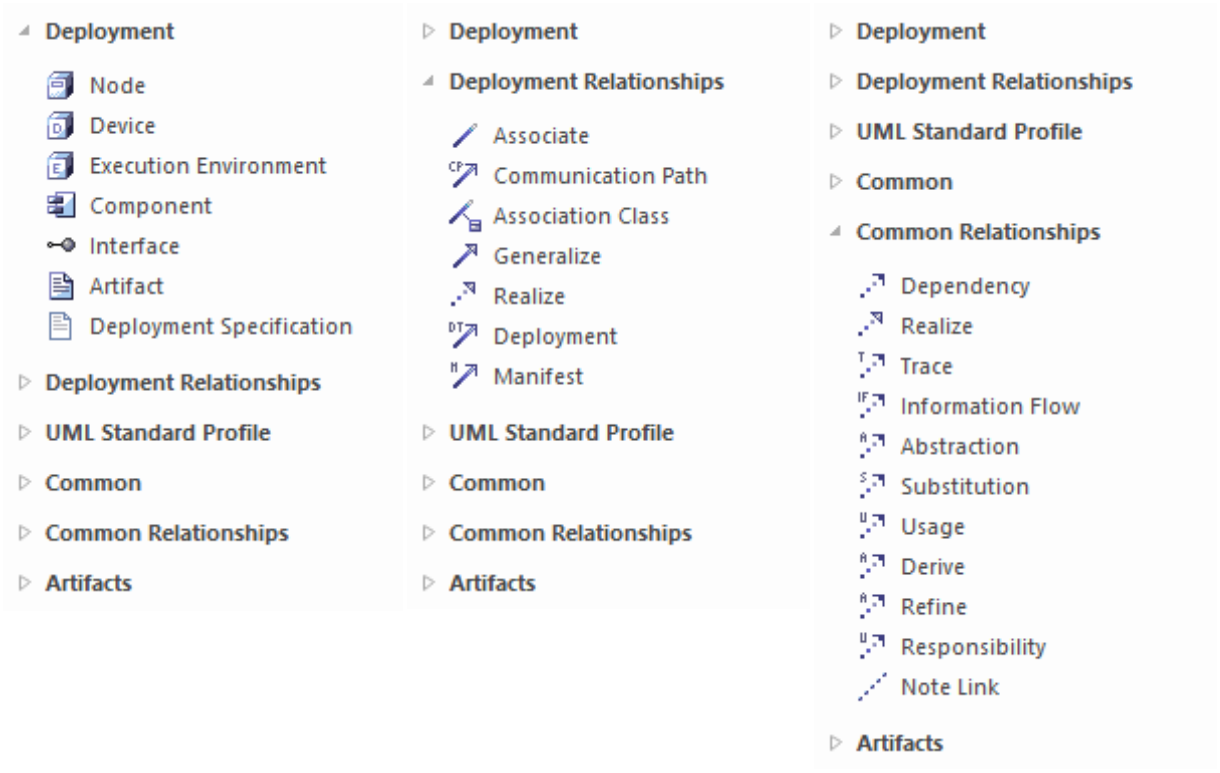


Slika 4.2.78 Definiranje stereotipa za komponentu vrste Artifact

Vrsta komponente *Component* po smislu više teži logičkoj komponenti ili općenitijoj vrsti komponente što je barem jednu razinu apstrakcije više nego komponenta *Artifact* kod koje se više misli na postojanje fizičkog zapisa komponente u memoriji računala.

Dijagram razmještaja

Alatni okviri vezani uz dijagram razmještaja sadrže više mogućih elemenata (a), odnosa između elemenata (b) te fizičkog i logičkog razmještaja sustava uz mogućnost umetanja i nekih općih vrsta veza (c), Slika 4.2.79.



(a) *(b)* *(c)*

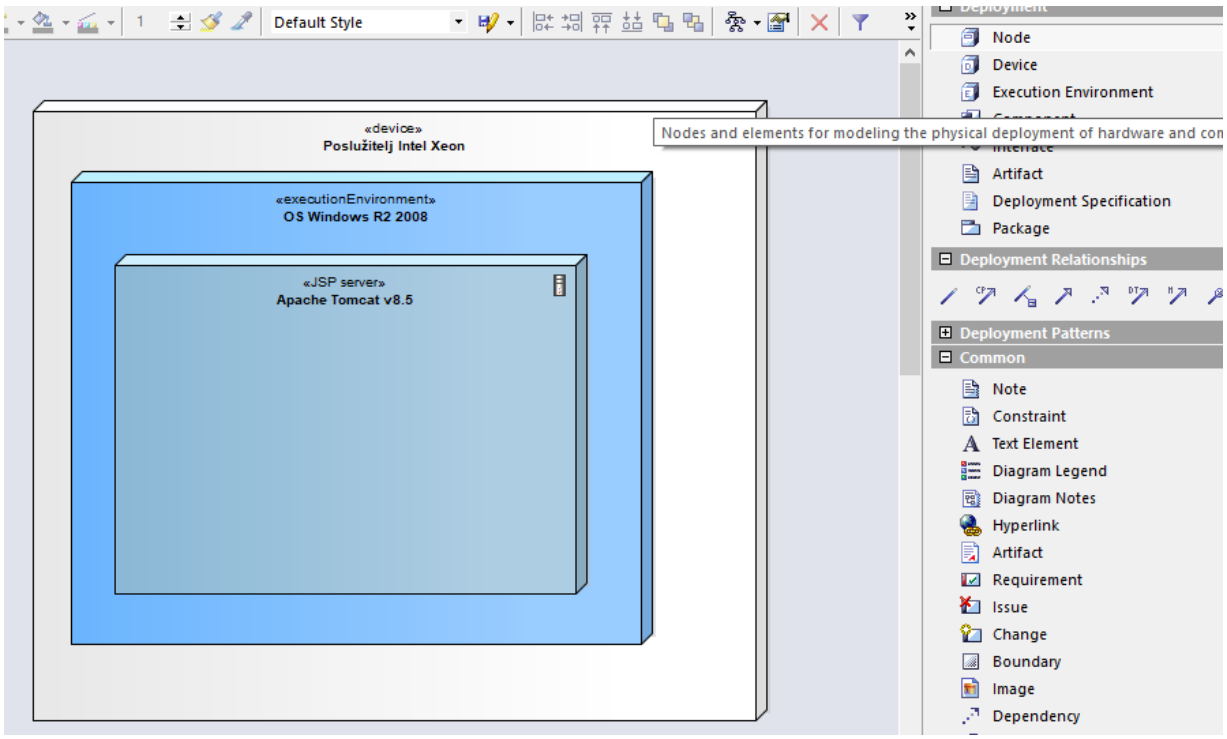
Slika 4.2.79 Alatni okviri dijagrama razmještaja

Alatni okvir *Deployment* pritom nudi predefinirane vrste čvorova kao što su obični čvor (*Node*), uređaj (*Device*) i okolina izvršavanja (*Execution Environment*).

Umetanje i hijerarhija čvorova

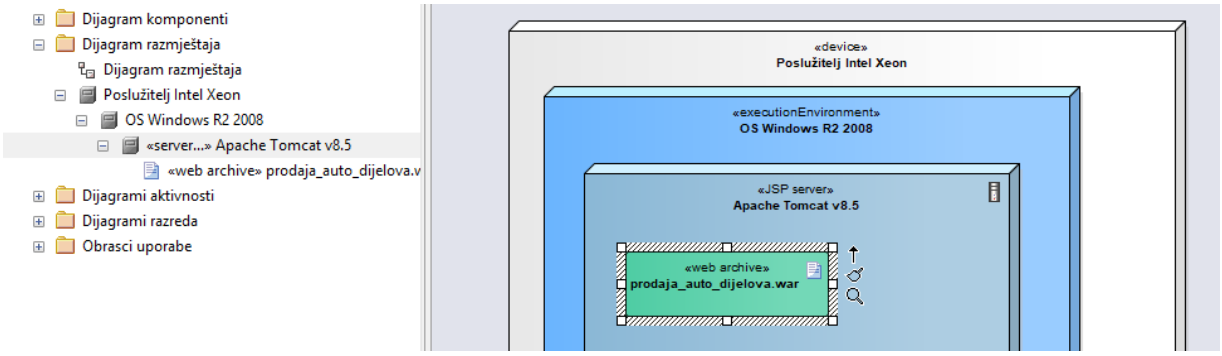
Postavljanje navedenih fizičkih dijelova sustava se svodi na izbor jedne od triju stavki iz alatnog okvira *Deployment*: *Node*, *Device* i *Execution Environment* i njihovu pravilnom smještanju unutar dijagrama. Preostale stavke iz istog alatnog okvira uglavnom su istovjetnog značenja vezanog uz oblikovanje dijagrama komponenti: *Component*, *Interface*, *Artifact* i *Package*, jer se, u pravilu, logičke komponente smještaju unutar prije definiranih fizičkih čvorova.

U primjeru hijerarhije ugniježdenih čvorova prikazuje se kako su međusobno jedan u drugome sadržani: poslužiteljsko računalo kao uređaj, operacijski sustav kao okolina izvršavanja i poslužiteljska implementacija, Slika 4.2.80.



Slika 4.2.80 Umetanje čvorova u dijagram razmještaja

Hijerarhija čvorova i komponenti jasno je vidljiva u projektnom navigatoru, Slika 4.2.81.

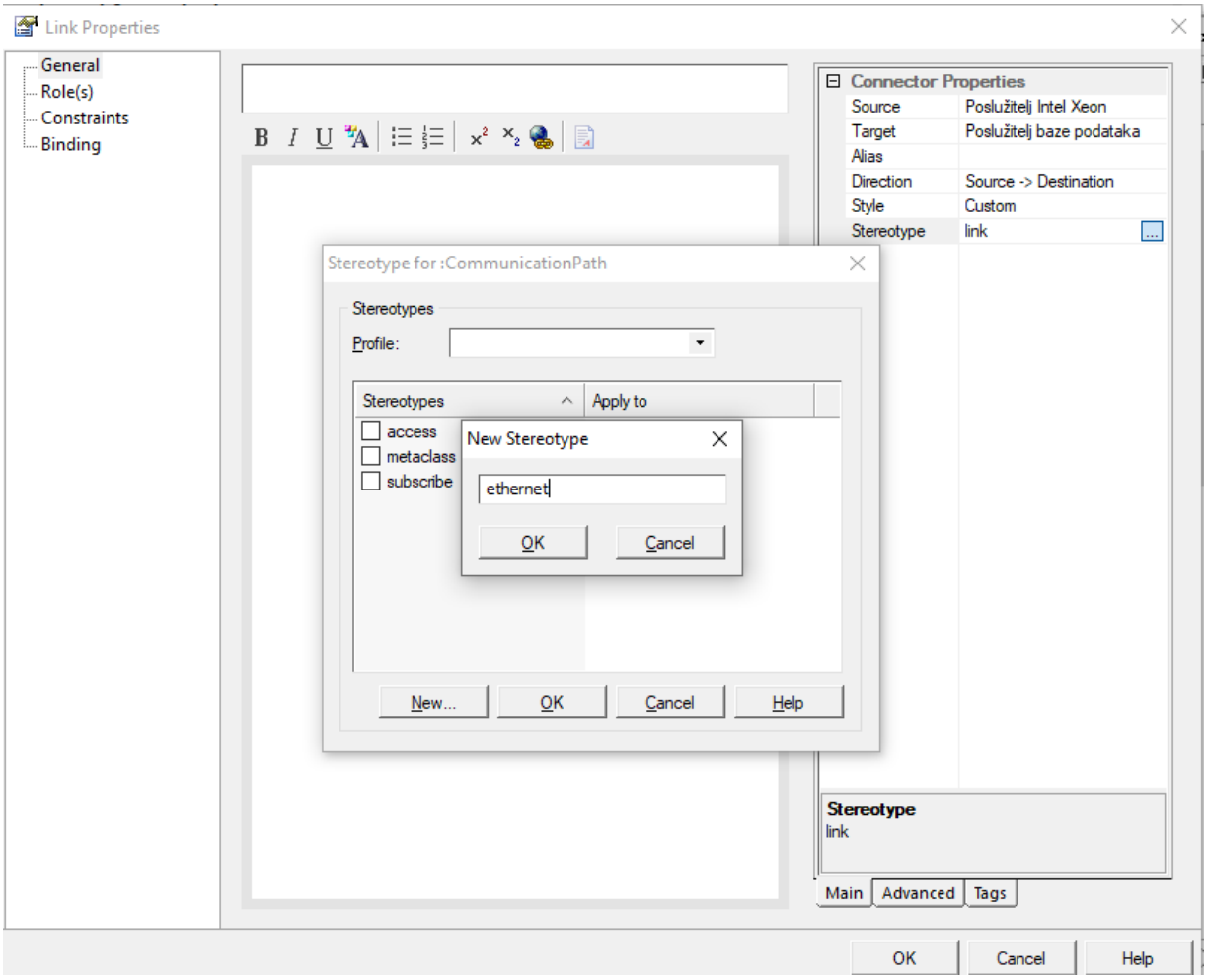


Slika 4.2.81 Prikaz hijerarhije čvorova i komponenti na dijagramu razmještaja

U tom kontekstu programsko ostvarenje sustava prodaje auto-dijelova fizički se nalazi unutar okoline izvršavanja kao datoteka web arhive te se pokreće na web poslužitelju.

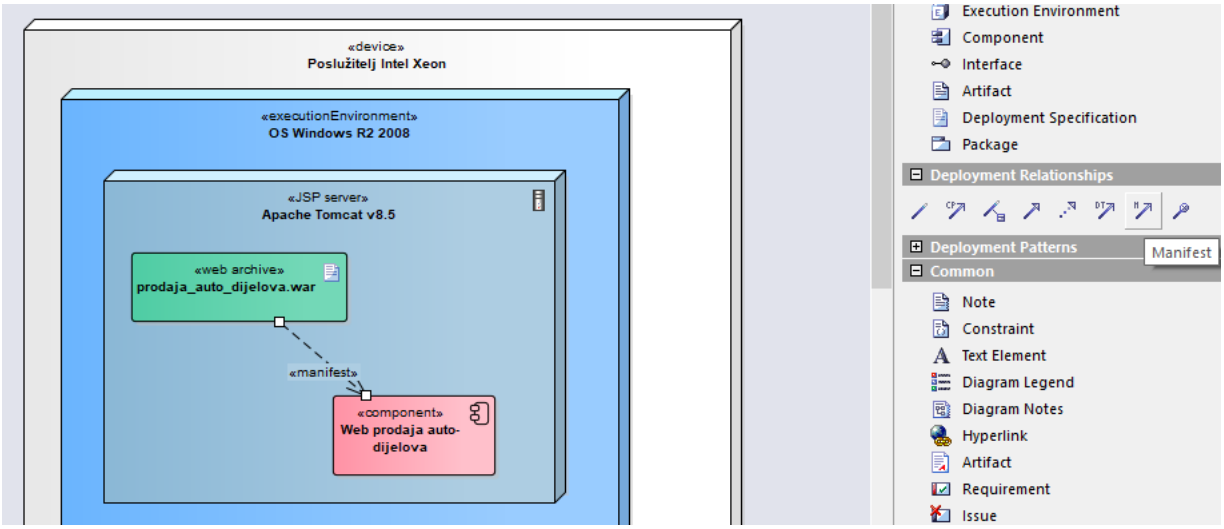
Umetanje veza između čvorova

Umetanjem veze vrste *Communication Path* iz alatnog okvira *Deployment Relationships* prikazuje se povezanost čvorova aplikacijskog poslužitelja i poslužitelja baze podataka, Slika 4.2.82. U ovom konkretnom slučaju se izabire i postavlja opis te veze poopćenim stereotipom *ethernet*, ali je moguće i specifičnije opisati tu vezu ovisno o njezinim konkretnim postavkama.



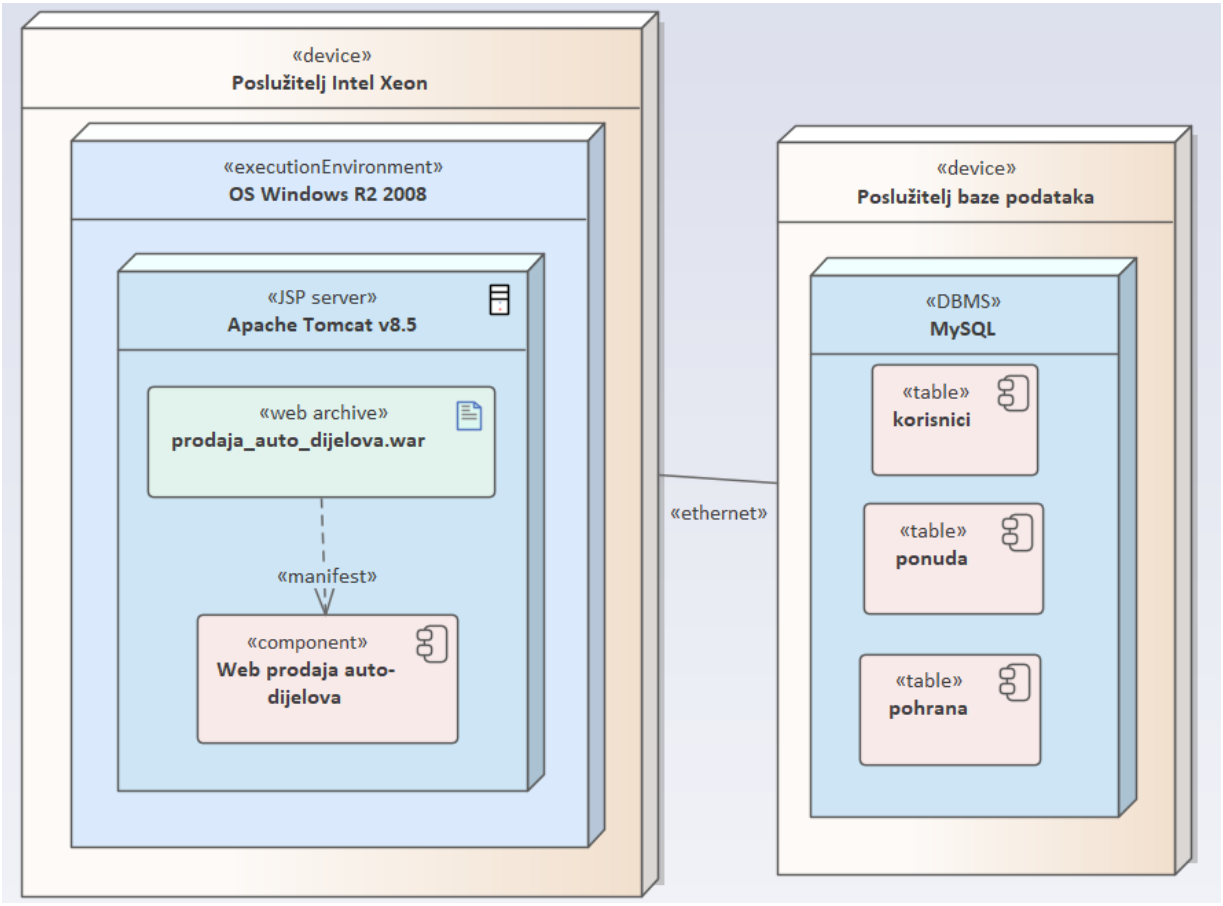
Slika 4.2.82 Definicija komunikacijske veze na dijagramu razmještaja

Definicija odnosa između komponenti koje se nalaze unutar čvorova radi se izborom odgovarajuće veze, npr. vrste *Manifest* koja je na raspolaganju unutar istog alatnog okvira kao specifičnosti odnosa između komponenti, Slika 4.2.83.



Slika 4.2.83 Smještaj i prikaz artefakta komponente dijagrama razmještaja

Slika 4.2.84 prikazuje rezultat umetanja navedenih veza u relevantnom dijelu dijagrama razmještaja.

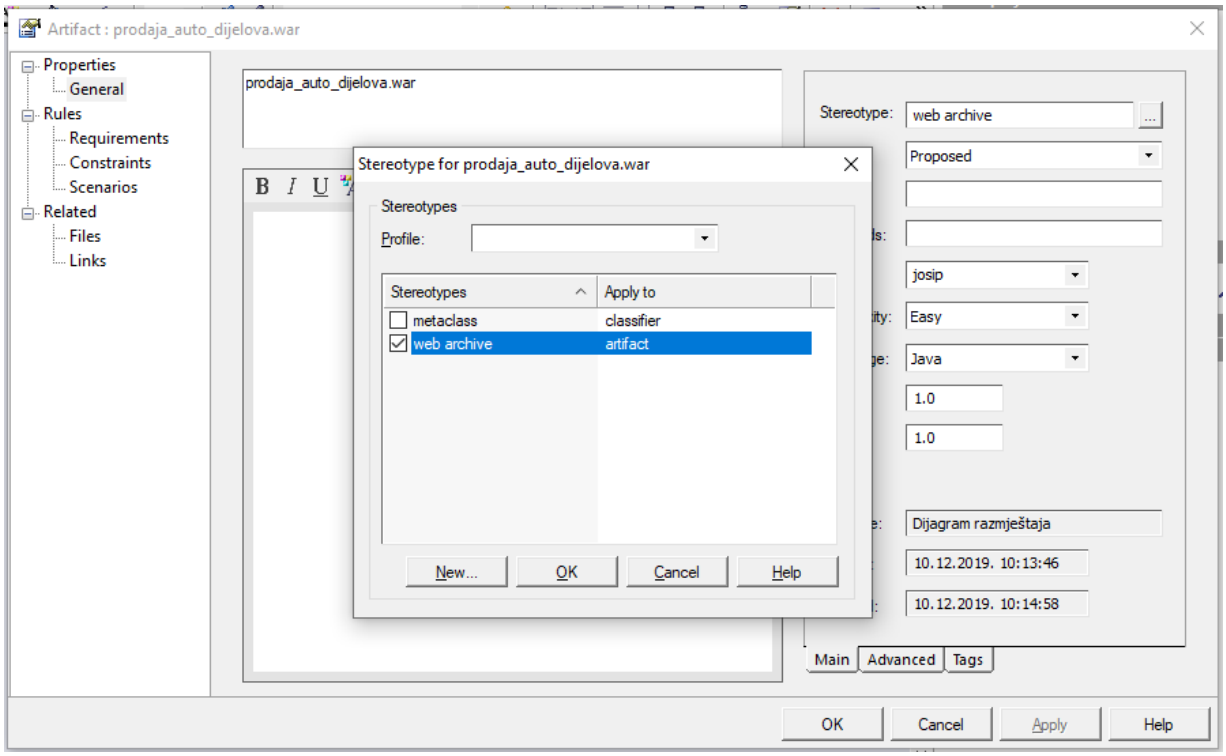


Slika 4.2.84 Primjer povezivanja aplikacijskog dijela i baze podataka na dijagrama razmještaja

Slika 2.11.1 prikazuje primjer cjelokupnog dijagrama razmještaja sa svim spomenutim čvorovima i vezama.

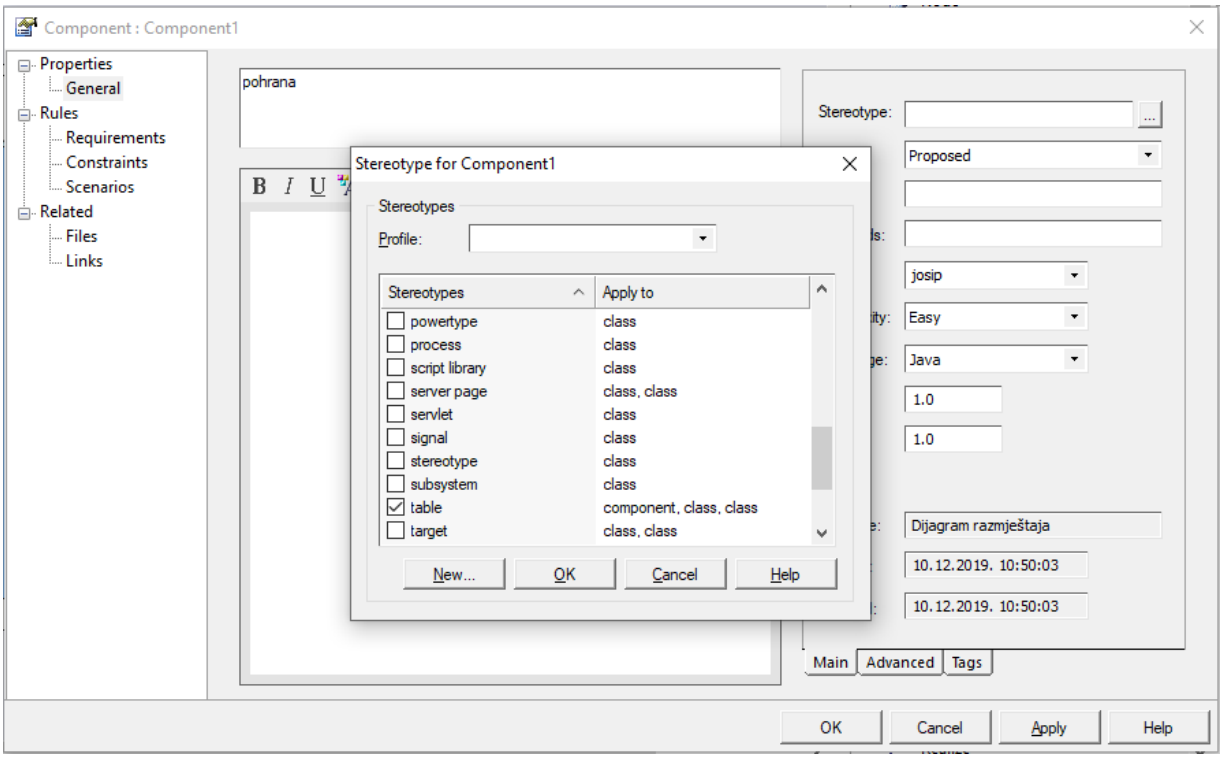
Definicije stereotipova

Umetanje stereotipa već je usput prikazano kao definicija osobitosti ili svojstva nekog odnosa ili veze između čvorova, Slika 4.2.82. Isto tako se to može primijeniti i na sam čvor ili komponentu, Slika 4.2.85.



Slika 4.2.85 Definicija stereotipa komponente na dijagramu razmještaja

Tablica baze podataka kao logička komponenta se može jasnije naznačiti uporabom stereotipa, Slika 4.2.86.

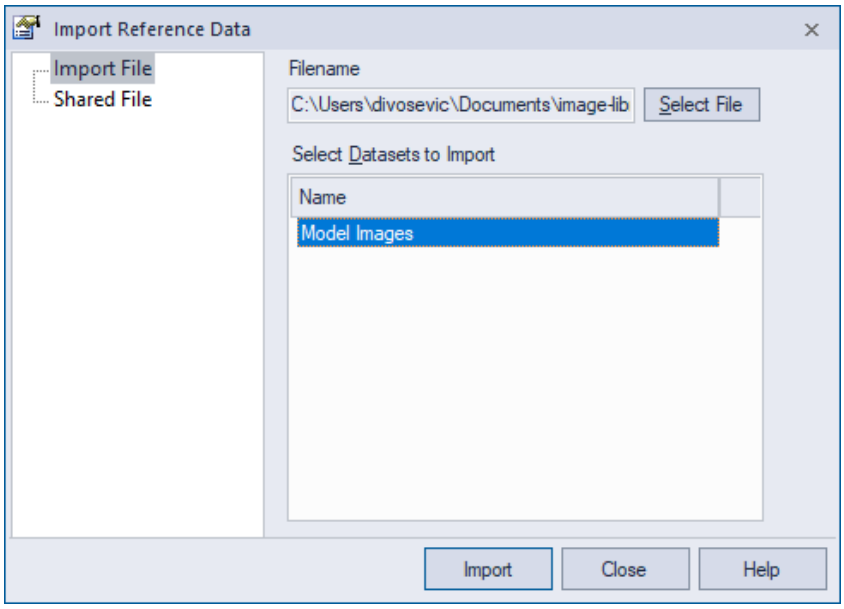


Slika 4.2.86 Definicija stereotipa komponente baze podataka na dijagramu razmještaja

Dijagram sa svojim čvorovima i komponentama je jasniji za razumijevanje kada se koriste odgovarajući stereotipovi pri umetanju komponenti i čvorova, u ovom slučaju za opis vrste sustava upravljanja bazom podataka i elemenata baze podataka, Slika 4.2.84.

Uvoz biblioteke simbola

Moguće je uvesti skup grafičkih simbola koji na dijagramima komponenti i razmještaja mogu slikovitije približiti namjenu pojedinih komponenti i čvorova. Izborom stavke *Import Reference Data* unutar glavnog izbornika otvara se dijaloški okvir *Import Reference Data*. U njemu je moguće izborom stavke *Import Data* i izborom datoteke biblioteke komponenti ([32]) dohvatiti XML datoteku grafičkih prikaza komponenti. U okviru *Select Datasets to Import* potom je potrebno izabrati stavku koja se nudi za uvoz te to potvrditi klikom na gumb *Import*, Slika 4.2.87.



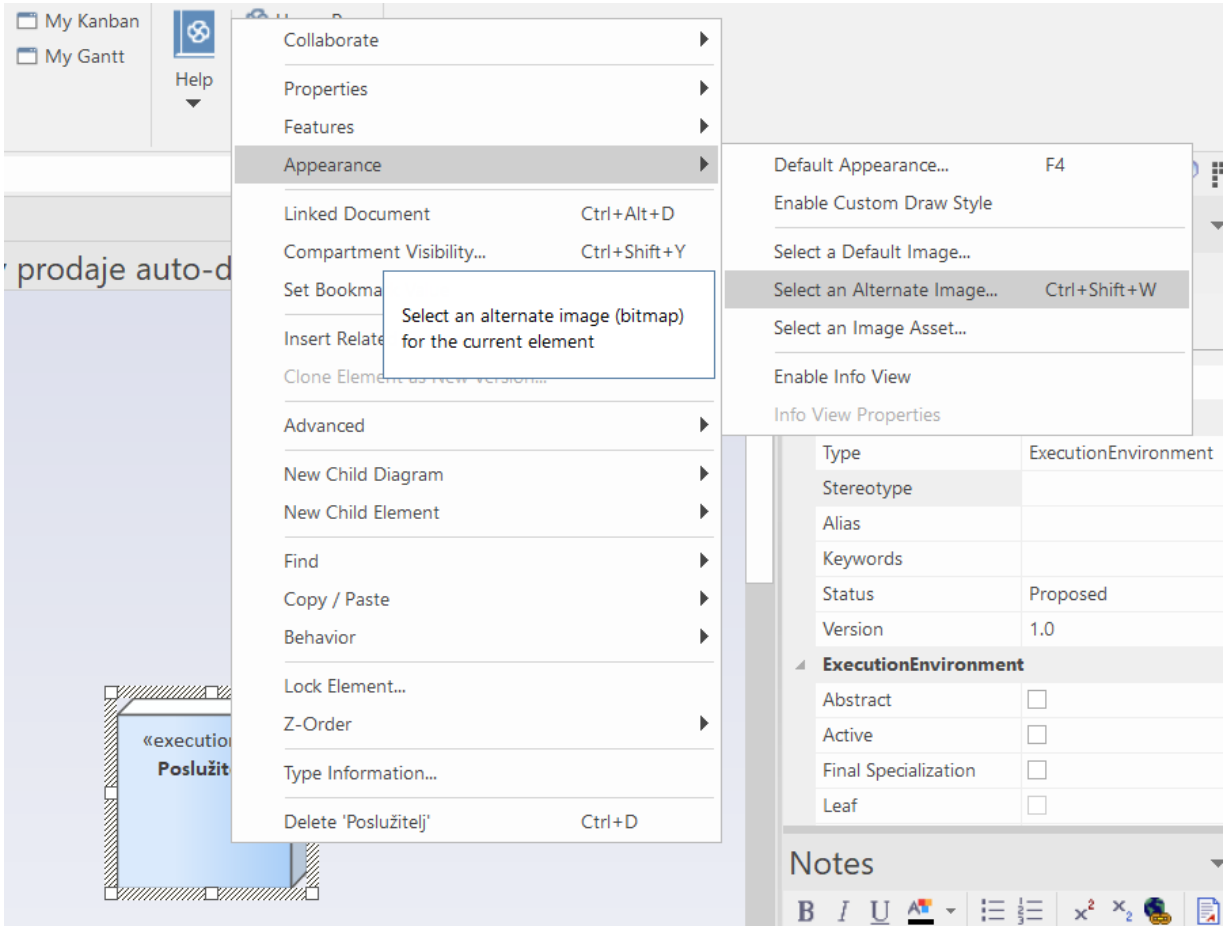
Slika 4.2.87 Uvoz datoteke simbola

Primjer uporabe uvezenih elemenata biblioteke je prikaz čvorova na dijagramu razmještaja, Slika 4.2.88.



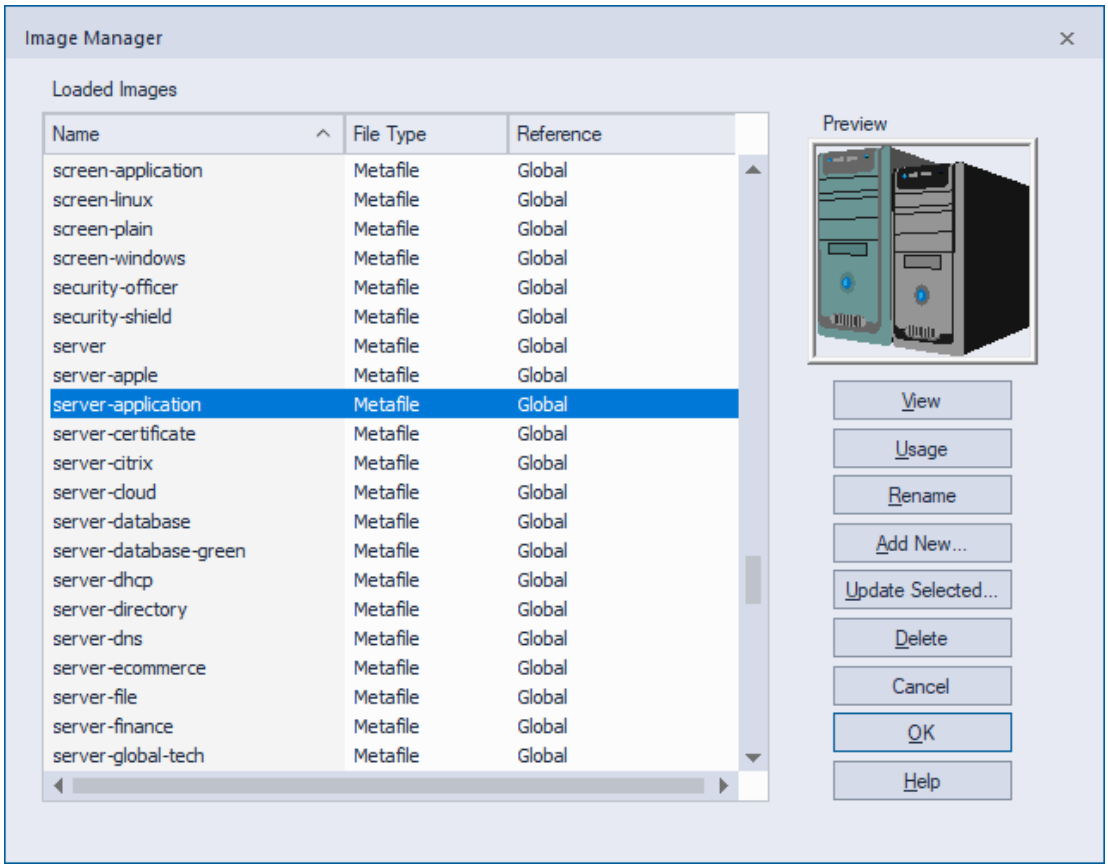
Slika 4.2.88 Jednostavni prikaz čvorova na dijagramu razmješaja

Desnim klikom na čvor i izborom *Appearance* → *Select an Alternate Image* iz potom otvorenog izbornika otvara se prozor *Image Manager* čiji okvir *Loaded Images* nudi popis stavki s grafičkim simbolima, Slika 4.2.89.



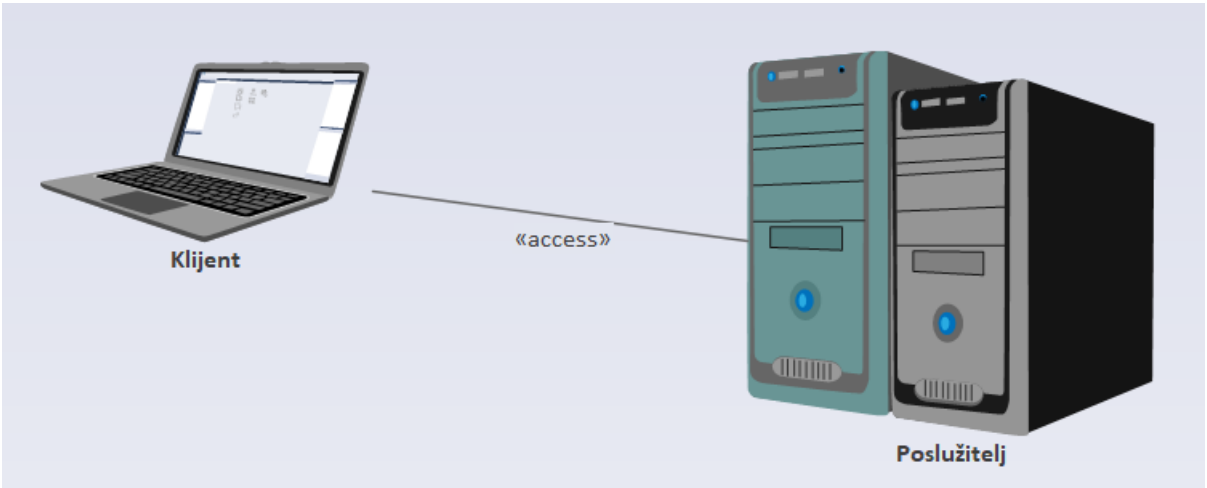
Slika 4.2.89 Izbor simbola iz uvezene biblioteke

Izborom i potvrdom odgovarajućeg simbola izabrani čvor poprima njegov izgled, Slika 4.2.90.



Slika 4.2.90 Pregled dostupnih simbola biblioteke

Nakon što se to ponovi i za ostale elemente dijagrama, sadržaj dijagram poprima prepoznatljivi oblik, Slika 4.2.91.



Slika 4.2.91 Simbolički prikaz čvorova na dijagramu razmještaja

Samo po sebi je jasno da naprednija grafička notacija može doprinijeti boljoj razumljivosti i prihvaćenosti dijagrama što je najpogodnije upotrijebiti u prikazu artefakata i čvorova u dijagramima komponenti i razmještaja.

5. Zaključak

Primjer sustava prodaje auto-dijelova opisan je putem modeliranja jezikom UML sve od funkcionalnih zahtjeva do ostvarenja programskih modula. *Enterprise Architect* predlaže više različitih UML modela koji su se u ovom primjeru dokumentacije jednim dijelom slijedili.

Model slučajeva uporabe uključuje pojedinačne dijagrame slučajeva uporabe s detaljnim prikazom ili dijagrame slijeda za prikaz više slučajeva uporabe. Dijagrami aktivnosti pomažu u pogledu na procese koji se izmjenjuju pri uporabi sustava.

Model razreda domene opisuje podatkovni model i domenu sustava te se funkcionalnošću generiranja programskog koda iz oblikovanih dijagrama dobiva okosnica programskog koda za daljnji razvoj. Model razreda na razini ostvarenja se temelji na razvijenom kodu programske potpore, djelomično se može dobiti uvozom tog programskog koda. Daljnjim profinjavanjem modela razreda na razini ostvarenja dobiva se svrsishodan i precizan model opisa arhitekture programske potpore.

Model komponenti i model razmještaja u konačnici prikazuju razvijene logičke komponente sustava te iste komponente u kombinaciji s fizičkih komponentama sustava.

Općenito se može reći da dijagrami slučajeva uporabe daju pogled na sustav izvana bez naznačene vremenske komponente ili uređenja, a dijagrami slijeda detaljnije pojašnjavaju komunikaciju između aktora ili objekata sustava kod nekog slučaja uporabe. Dijagrami aktivnosti prikazuju tokove aktivnosti u poslovnom procesu s jasnim vremenskih uređenjem procesa. Dijagram razreda čini okosnicu arhitekture sustava i na razini podatkovnog modela i na razini arhitekture programske potpore, dok dijagrami komponenti i razmještaja prikazuju arhitekturu programske potpore na razini stvarnih komponenti koje postoje, logičkih i fizičkih.

Idejno, modelno zasnovan razvoj posebno je izazovan u pogledu određene standardizacije, dobivanja precizne specifikacije i generiranja proizvoda s više razine opisa. Osim jezika i notacije UML, postoje i drugi standardi. kao što su BPMN za opis poslovnih procesa i SysML kao produžetak jezika UML za opis sustava.

Alat *Enterprise Architect* nudi jednu sveobuhvatnu okolinu oblikovanja ideje od početne zamisli do opisa krajnjeg rezultata te je u ovom dokumentu dotaknut samo djelić funkcionalnosti iz cjeline koja pretpostavlja određeni smjer razmišljanja koji bi pokrio sve vidove razvoja programske potpore. Osim njega, postoje i drugi, sveobuhvatniji programski alati kao što su Visual Paradigm i alati iz lepeze IBM Rational koji osim mogućnosti modeliranja UML-om teže pokrivanju cjelokupnog poslovnog modela i procesa razvoja. Uz sve njih, dostupni su i brojni alati koji su specifično namijenjeni zadacima modeliranja UML-om.

6. Literatura

1. Objektno orijentirani razvoj programa, https://moj.tvz.hr/, (11.09.2023.)
2. Objektno orijentirani razvoj programa, https://lms-2020.tvz.hr/course/view.php?id=627, (06.09.2023.)
3. Softversko inženjerstvo, http://web.studenti.math.pmf.unizg.hr/~manger/si/, (11.09.2023.)
4. Programsko inženjerstvo, https://www.fer.unizg.hr/predmet/proinz, (11.09.2023.)
5. OMG Standards Development Organization, https://www.omg.org/, (11.09.2023.)
6. OMG Unified Modeling Language, https://www.omg.org/spec/UML/2.5.1/PDF (11.09.2023.)
7. Ian Sommerville, Software Engineering, 10th Edition, Pearson, 2016.
8. Software Engineering, Tenth Edition, https://software-engineering-book.com/, (11.09.2023.)
9. Robert Manger, Softversko inženjerstvo, Element, 2016.
10. Alan Jović, Nikolina Frid: Procesi programskog inženjerstva, 5. izdanje, Sveučilište u Zagrebu, FER ZEMRIS, 2022. https://www.fer.unizg.hr/_download/repository/Procesi_programskog_inzenjerstva_3_izdanje[1].pdf (11.09.2023.)
11. Steve Tockey, How to Engineer Software, https://www.construx.com/books/how-to-engineer-software/, (11.09.2023.)
12. Alan Jović, Marko Horvat, Igor Grudenić, UML-dijagrami, zbirka primjera i riješenih zadataka, Graphis, 2014.
13. Timothy C. Lethbridge and Robert Laganière, Object-Oriented Software Engineering: Practical Software Development using UML and Java Second Edition, https://www.site.uottawa.ca/school/research/lloseng/ (11.09.2023.)
14. Bran Selic, The pragmatics of model-driven development. IEEE Software 20(5), 19-25, October 2003.
15. Bran Selic, Model-Driven Development: Its Essence and Opportunities, Ninth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC 2006), 24-26 April 2006, Gyeongju, Korea
16. MDA – The Architecture of Choice for a Changing World, https://www.omg.org/mda/, (11.09.2023.)
17. SparxSystems MDA Overview, https://sparxsystems.com/downloads/whitepapers/MDAOverview.pdf, White Paper, (11.09.2023.),
18. Ivar Jacobson International, http://ivarjacobson.com/, (11.09.2023.)
19. Rational Unified Process, https://web.archive.org/web/20040402113344/http://www-306.ibm.com/software/awdtools/rup/, (11.09.2023.)
20. IEEE Recommended Practice for Software Requirements Specifications, http://www.cse.msu.edu/~cse870/IEEEXplore-SRS-template.pdf (11.09.2023.)
21. Ammar Botonjić, Unified Modeling Language (UML): Use Case Diagram, Završni rad, Sveučilište Jurja Dobrile u Puli, 2015.

22. Katarina Đaić , Najčešće korišteni UML dijagrami u programskom inženjerstvu, Završni rad, Sveučilište Jurja Dobrile u Puli, 2017.
23. Edmond Krusha, Enterprise Architect, Verzije 11 i 12, https://bib.irb.hr/datoteka/838138.EA_Studentski_prirunik_-_TVZ_EA_Verzija_12-280315.pdf (12.10.2022.)
24. Željko Kovačević, Edmond Krusha, Programsko inženjerstvo i informacijski sustavi, Tehničko veleučilište u Zagrebu, 2020., https://www.bib.irb.hr/1096475/download/1096475.Programsko_inenjerstvo_i_informacijski_sustavi.pdf (11.09.2023.)
25. The Unified Modeling Language, https://www.uml-diagrams.org/ (11.09.2023.)
26. Martina Seidl, Marion Scholz, Christian Huemer, Gerti Kappel, UML@Classroom, http://www.uml.ac.at/, (11.09.2023.)
27. SparxSystems Enterprise Architect, https://sparxsystems.com/products/ea/, (11.09.2023.)
28. Enterprise Architect 15.2 User Guide, https://sparxsystems.com/enterprise_architect_user_guide/15.2/index/index.html (11.09.2023.)
29. SparxSystems Tutorials, https://sparxsystems.com/resources/tutorials/index.html, (11.09.2023.)
30. SparxSystems Enterprise Architect Tutorial, https://sparxsystems.com/enterprise_architect_user_guide/15.2/user_interface/quick_start.html, (11.09.2023.)
31. SparxSystems Code Engineering, https://sparxsystems.com/enterprise-architect/code-engineering/code-engineering.html, (11.09. 2023.)
32. SparxSystems, Image Library Download, https://sparxsystems.com/resources/image_library.html, (11.09.2023.)
33. IBM Rational Software Architect documentation, https://www.ibm.com/docs/en/rational-soft-arch, (11.09.2023.)
34. Visual Paradigm, https://www.visual-paradigm.com/ (11.09.2023.)
35. MBSE and SysML, https://www.visual-paradigm.com/guide/sysml/mbse-and-sysml/, (11.09.2023.)
36. Steve Tockey - Software Engineering 101, https://www.youtube.com/watch?v=wdOavmxmye4 (11.09.2023.)
37. Steve Tockey - How to Engineer Software, Part 1: Semantic Models, https://www.youtube.com/watch?v=LqMhha360XM (11.09.2023.)
38. Steve Tockey - How to Engineer Software, Part 2: A Deeper Dive into Semantic Models, https://www.youtube.com/watch?v=FjGh8YMo_tU (11.09.2023.)
39. Steve Tockey - How to Engineer Software, Part 3: Designing and Coding from Semantic Models, https://www.youtube.com/watch?v=8uJg-qfeqVM (11.09.2023.)
40. Spring MVC CRUD using MongoDB Tutorial, https://examples.javacodegeeks.com/enterprise-java/spring/mvc/spring-mvc-crud-using-mongodb-tutorial/, (11.09.2023.)
41. Modeling Languages, https://modeling-languages.com/, (11.09.2023.)

42. Nikolina Frid, Alan Jović, UML zadaci, Primjeri modeliranja UML dijagramima, , Sveučilište u Zagrebu, FER ZEMRIS, 2023., https://www.fer.unizg.hr/_download/repository/UML_zadaci_za_vjezbu_-_nadopuna_sveucilisnog_prirucnika[1].pdf, (11.09.2023.)

7. Dodaci

7.1. Generirani programski kodovi

Generirani programski kodovi se odnose na sadržaj poglavlja 4.2 gdje se prilikom opisa rada s dijagramom razreda navodi mogućnost dobivanja programskog koda iz dijagrama.
Ovdje se konkretno za dijagram razreda korisničkog modela iz poglavlja 2.7 prikazuje rezultat generiranja koda.

Dijagram razreda korisničkog modela

Pobrojavanje 'RazinaPristupa' (datoteka *RazinaPristupa.java*)

```
public enum RazinaPristupa {
    ADMIN,
    TRGOVAC,
    KUPAC
}
```

Razred 'Korisnik' (datoteka *Korisnik.java*)

```
private class Korisnik {

    private String eposta;
    private String id;
    private String ime;
    private String korisnickoIme;
    private String prezime;
    private String zaporka;
    public RazinaPristupa m_RazinaPristupa;
    public BankovnaKartica m_BankovnaKartica;

    public Korisnik(){

    }

    public void finalize() throws Throwable {

    }
    public String getEposta(){
        return eposta;
    }
```

```
public String getId(){
    return id;
}

public String getIme(){
    return ime;
}

public String getKorisnickoIme(){
    return korisnickoIme;
}

public String getPrezime(){
    return prezime;
}

public String getZaporka(){
    return zaporka;
}

/**
 *
 * @param newVal
 */
public void setEposta(String newVal){
    eposta = newVal;
}

/**
 *
 * @param newVal
 */
public void setId(String newVal){
    id = newVal;
}

/**
 *
 * @param newVal
 */
public void setIme(String newVal){
    ime = newVal;
}

/**
 *
 * @param newVal
 */
public void setKorisnickoIme(String newVal){
```

```
        korisnickoIme = newVal;
    }

    /**
     *
     * @param newVal
     */
    public void setPrezime(String newVal){
        prezime = newVal;
    }

    /**
     *
     * @param newVal
     */
    public void setZaporka(String newVal){
        zaporka = newVal;
    }
}//end Korisnik
```

7.2. Primjeri ispitnih zadataka

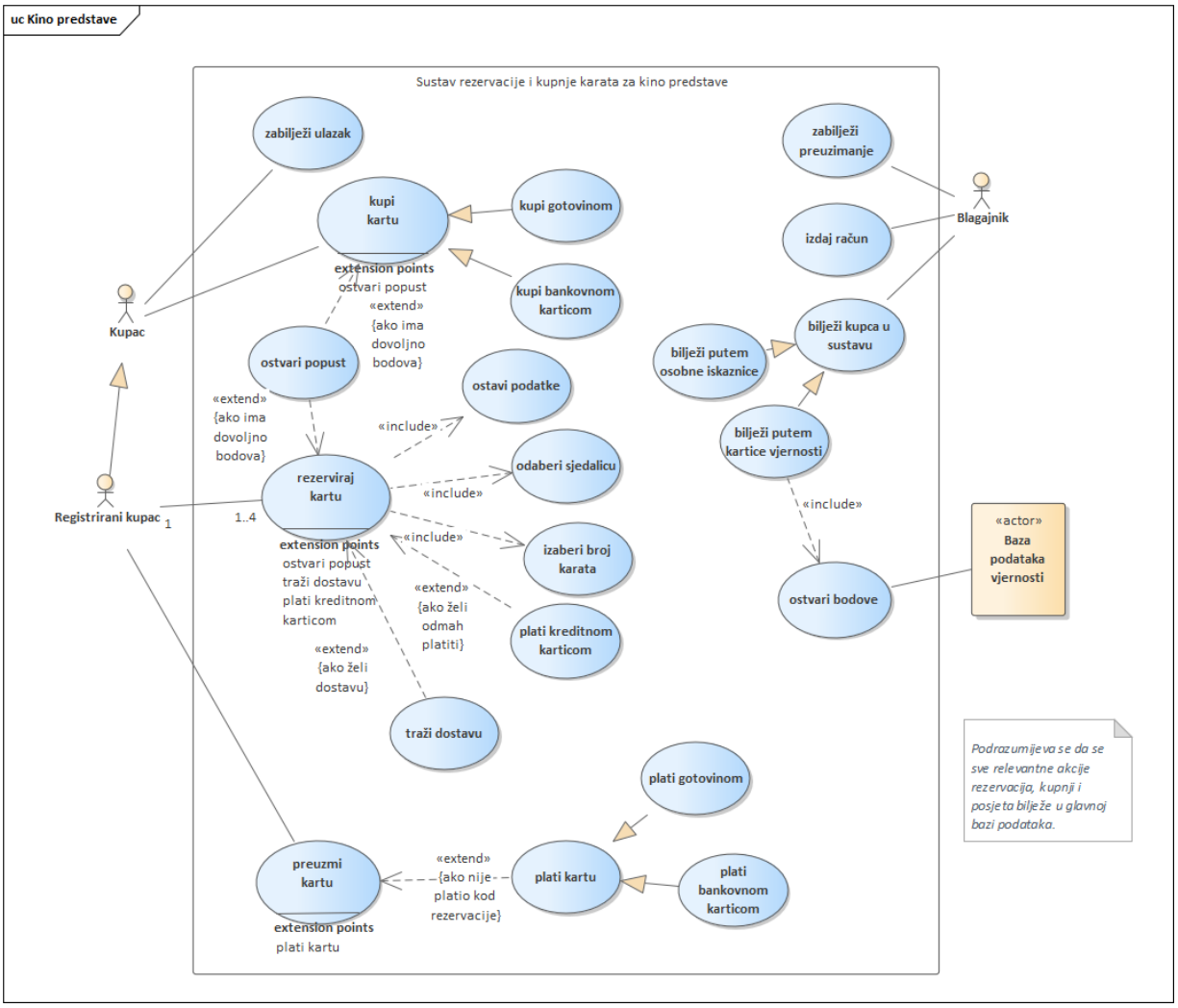
U ovom poglavlju se ukratko opisuju dva primjera ispitnih zadataka koji se sastoje u oblikovanju dijagrama slučajeva uporabe i dijagrama razreda. Mogu poslužiti, osim kao primjer oblikovanja dijagrama, i kao primjer za vježbu modeliranju UML-om u proizvoljno izabranom alatu programske potpore za notaciju UML.

Dijagram slučajeva uporabe

Tekst zadatka je sljedeći:

Dijagramom slučajeva uporabe modelirajte sustav rezervacije i kupnje karata/ulaznica za kino predstave. Bilo koji kupac može kupiti kartu za predstavu na ulazu u kino dvoranu na način da plati i preuzme kartu na blagajni. Registrirani kupac može rezervirati kartu putem usluge web sučelja. Pri rezervaciji karte on najprije mora ostaviti osobne podatke, odabrati brojeve sjedalica i potvrditi broj karata s ukupnom cijenom. Pritom može odmah i platiti kartu kreditnom karticom i tražiti dostavu karte na kućnu adresu. Inače, kartu može platiti kod podizanja na blagajni na licu mjesta. Karta se na blagajni može platiti gotovinom ili bankovnom karticom. Registrirani kupac može rezervirati najviše četiri karte odjednom. Na blagajni dvorane radi blagajnik koji bilježi da je karta preuzeta od strane kupca te kupcu uručuje račun. Ako kupac nije radio rezervaciju preko weba, onda blagajnik provlači njegovu osobnu iskaznicu kroz čitač koji bilježi njegove podatke u sustavu. Ako posjetitelj ima karticu vjernosti onda provlači tu karticu umjesto

osobne iskaznice, a na karticu vjernosti mu se bilježe bodovi vjernosti pri kupnji ulaznica. Bodovi se bilježe u posebnoj bazi podataka članova kluba vjernosti. Na temelju skupljenih bodova, posjetitelj može ostvariti popust kod rezervacije odnosno kupnje karte za predstavu. Pri ulasku u dvoranu kupci provlače karte kroz čitač koji bilježi ulazak posjetitelja u dvoranu. Podrazumijeva se da sustav kino multipleksa koji prikazuje predstave ima bazu podataka u koju se bilježe sve rezervacije, kupovine i posjete predstavama od strane korisnika.



Slika 7.2.1 Rješenje zadatka dijagram slučajeva uporabe

Provlačenja kartica nisu doslovno navedena kao posebni slučajevi uporabe jer nije naglasak na samim fizičkim akcijama već na učincima tih akcija, tj. bilježenju odgovarajućih podataka u sustavu.

Specijalizacije *kupi gotovinom* i *kupi bankovnom karticom* slučaja uporabe *kupi kartu* su bolje rješenje nego povezivanje tog slučaja na slučaj uporabe *plati kartu*.

Neki parovi slučajeva uporabe imaju semantičku povezanost ali nije potrebno to posebno prikazivati: plaćanje karte i izdavanje računa, preuzimanje karte i bilježenje preuzimanja od strane blagajnika.

Dijagram razreda

Tekst zadatka je sljedeći:

Kino multipleks ima više lokacija. Lokacija je određena s nazivom i adresom.

Na svakoj lokaciji postoji više dvorana. Svaka dvorana ima svoj naziv, broj sjedećih mjesta i raspored predstava. Predstava je obilježena cijenom, satom i filmom u to vrijeme.

Film ima naslov, ime glavnog glumca i glumice, trajanje, žanr i oznaku primjerenosti ili zabrane gledanja ovisno o minimalnoj dobi gledatelja. Oznaka primjerenosti može: U12, U15 ili U18.

Za svaku predstavu se izdaje neka količina ulaznica. Svaka ulaznica ima oznaku sjedećeg mjesta: kategoriju, red, broj sjedalice.

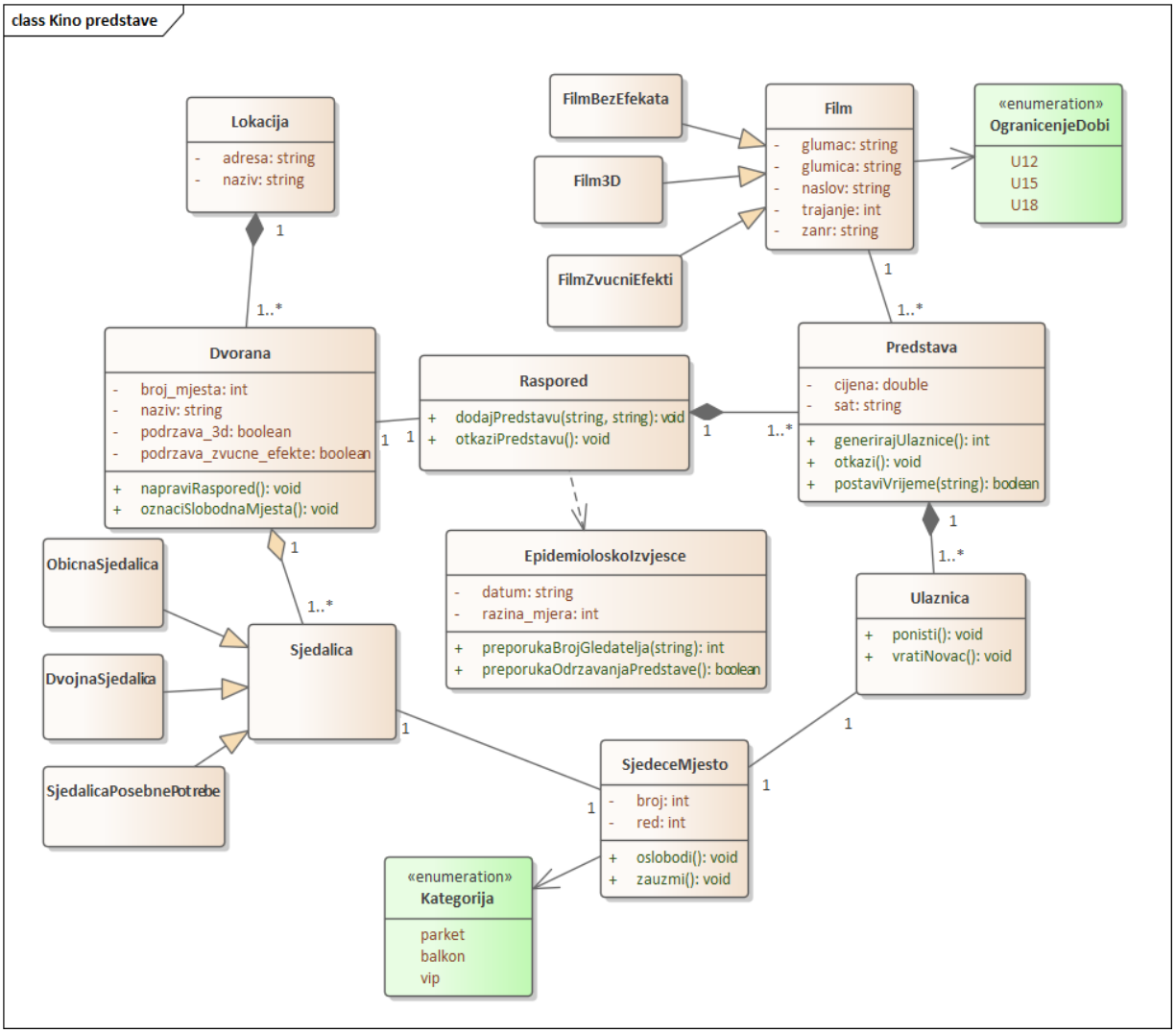
Kod dodavanja predstave u raspored generira se neki broj ulaznica po predstavi a vrijeme prikazivanja se može i promijeniti.

Ako se predstava otkaže zbog epidemiološke ili koje druge situacije ulaznice više ne vrijede, a novci se vraćaju te se poništavaju sve generirane ulaznice. Sjedeća mjesta se zauzimaju kod kupnje ili rezervacije ulaznica za pojedinu predstavu.

Dvorana ima neki broj sjedalica. Sjedalice su mobilne i mogu se premještati iz dvorane u dvoranu ovisno o potrebama. Dvorana je podijeljena prema kategorijama sjedenja kojih ima više. Kategorija može biti parket, balkon ili vip. Ako se dvorana prestane koristiti kategorije ostaju jer nisu ovisne o dvorani. Poseban entitet je epidemiološko izvješće o kojem ovisi svaka predstava (hoće li se održati ili neće). Izvješće sadrži datum i razinu epidemioloških mjera (cjelobrojna vrijednost) koji su na snazi za navedeni datum. Za svako izvješće mogu se dobiti posebne informacije o preporuci o održavanju predstave i o preporuci broja posjetitelja, ovisno o dvorani. U tom slučaju se za pojedinu dvoranu mogu označiti odnosno napraviti raspored dozvoljenih sjedalica za korištenje.

Sama sjedalica po vrsti može biti obična, dvojna ili za osobe s posebnim potrebama.

Filmovi koji se prikazuju prema načinu projekcije mogu biti obični, 3D ili s posebnim zvučnim efektima, što ovisi i o dvorani, tj. koliko pojedina dvorana podržava pojedine načine projekcije.



Slika 7.2.2 Rješenje zadatka dijagrama razreda

Ovisnost projekcije nekog filma o mogućnostima dvorane se modelira posredno: putem atributa *dvorane podrzava_3d* i *podrzava_zvucne_efekte*.

S obzirom na to da nisu navedeni posebni dodatni atributi za specijalizacije razreda *Sjedalica* i razreda *Film* moguće je primijeniti i pobrojavanja umjesto specijalizacija.

Ako bi se dodao poseban atribut *ponistena* za razred Ulaznica kao i poseban atribut otkazana za razred *Predstava* eventualno bi se mogle prikazati i obične agregacije umjesto kompozicija na razredima *Predstava* i *Raspored*.

Kad bi cijena ulaznice ovisila i o kategoriji sjedećeg mjesta onda bi razred *Ulaznica* također imao atribut cijena i eventualno metodu *izracunajCijenu()*. Ovako cijena ovisi samo o filmu i vremenu prikazivanja.