Dijagrami UML-a

Marina Ivašić-Kos, dr. sc. Mile Pavlić, Patrizia Poščić

Odsjek za informatiku

Filozofski fakultet u Rijeci

marinai@pefri.hr,

ris.pavlic@ri.tel.hr

patrizia@pefri.hr

1. Uvod

- Unified Modeling Language-UML je jezik za specifikaciju, vizualizaciju, konstruiranje i dokumentiranje programskog sustava, poslovnog sustava i drugih ne programskih sustava.

- Može se koristiti za poslovno modeliranje, softversko modeliranje u svim fazama razvoja i za sve tipove sustava, te opće modeliranje bilo koje konstrukcije koja ima statičku strukturu i dinamičko ponašanje.

- Analiziraju se dijagrami UML-a verzije 1.3.

2. Pogledi i dijagrami

- Složenom sustavu najbolje je pristupiti putem skupa malih gotovo neovisnih **pogleda** na model koji pokazuje posebni aspekt sustava, fokusirajući relevantne detalje, a pritom ignorirajući ostale.

- **Pogled** je apstrakcija koja se sastoji od dijagrama.
- **Dijagram**:
  - graf koji pokazuje simbole elemenata modela sređene tako da ilustriraju pojedini dio ili aspekt sustava
  - dovoljan za jednostavnu komunikaciju
  - mora učestvovati u upotpunjavanju drugih dijagrama
  - povezan sa drugim dijagramima i pogledima tako da je sustav kompletno opisan kroz sve poglede zajedno.

- Izbor koji će se dijagrami koristiti ovisi o pristupu problemu i odgovarajućem rješenju.

Obzirom na pogled na model, UML definira slijedeće dijagrame:

- ✓ pogled na primjenu.....dijagram korištenja (*use case diagram*)
- ✓ pogled na statiku.....dijagram klasa (*class diagram*)
- ✓ pogled na ponašanje.....dijagrami ponašanja:
  - dijagram karte stanja (*statechart diagram*)
  - dijagram aktivnosti (*activity diagram*)
- ✓ pogled na interakciju.....dijagrami interakcije:
  - dijagram slijeda (*sequence diagram*)
  - dijagram suradnje (*collaboration diagram*)
- ✓ pogled na implementaciju..... dijagrami implementacije:
  - dijagram komponenata (*component diagram*)
  - dijagram rasporeda (*deployment diagram*)

2.1 Dijagrami statične strukture

- Dijagram se smatra statičnim ako je opisana struktura uvijek valjana u svakoj točki životnog ciklusa sustava.

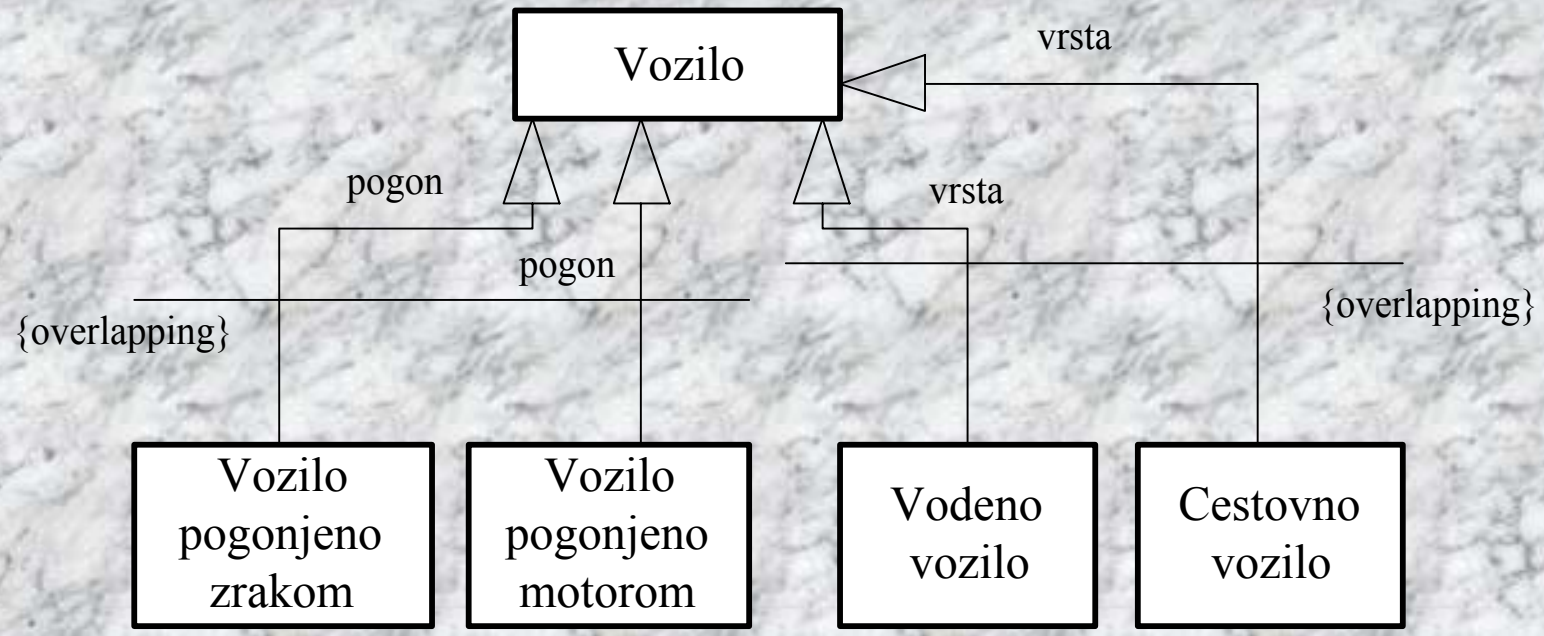
- Statičnu strukturu modela čine:
  - entiteti koji postoje,
  - unutarnju strukturu entiteta,
  - veze sa drugim entitetima.

2.1.1 Dijagram klasa

- pokazuje statičku strukturu sustavu definirajući elemente klasifikatora (klase, sučelja, paketi, veze, instance,..) i statične odnose među njima.

- na dijagramu klasa prikazana je unutarnja struktura klasa koju definiraju atributi i operacije.

Sustav najčešće ima brojne dijagrame klasa jer nisu sve klase uključene u jedan dijagram klase. Isto tako neka klasa može, kada je potrebno, sudjelovati u nekoliko dijagrama klasa.



Prikaz generalizacije sa ograničenjima i diskiminatorima na dijagramu klasa

2.1.2 Dijagram objekata

- graf instanci koji uključuje objekte i vrijednosti podataka

- instanca ili primjer dijagrama klase (umjesto aktualnih klasa, prikazuje primjere klasa, moguću snimku stanja sustava u nekom (dijelu) vremena)

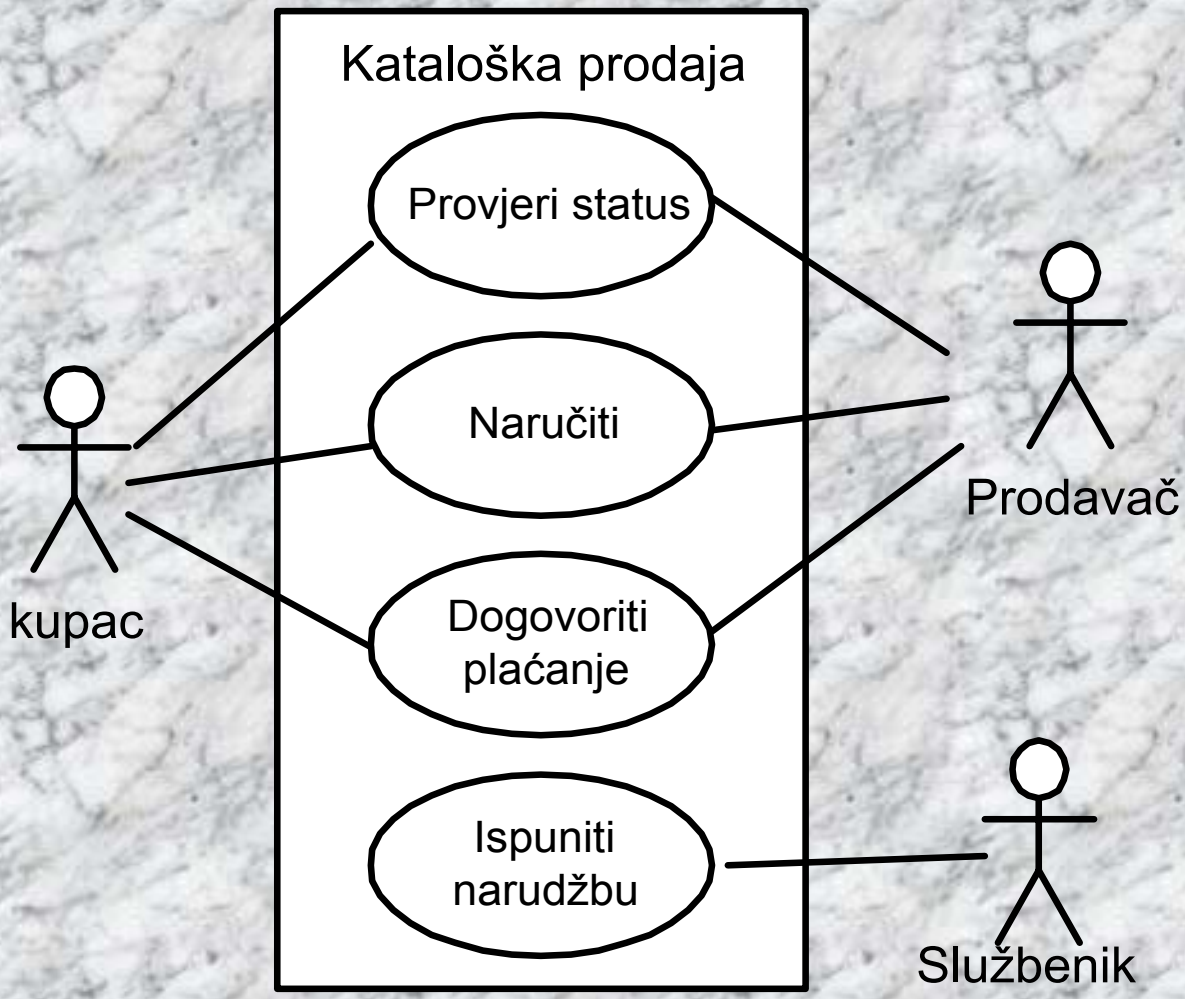
Dijagrami objekata nemaju važnost kao što je imaju dijagrami klase, ali korisni su za «pojašnjavanje» kompleksnih dijagrama klasa prikazujući mogući primjer dijagrama klasa, kao i prikaz suradnje između skupa objekata.

2.2 Dijagram korištenja

- funkcionalnost koju sustav omogućuje
- primarne komponente:
  - način korištenja
  - učesnici.

- **Način korištenja**:
  - definira stvarne zahtjeve i opisuje funkcionalnost sustava ili klasifikatora iz perspektive učesnika
  - mora rukovati cijelom funkcijom, od inicijalizacije koju je pokrenuo učesnik, do funkcionalnost i vrijednost koju je zahtjevao
  - opisuje kako učesnici mogu koristiti sustav ali ne kako je ta funkcionalnost podržana unutar sustava.

- **Učesnik: -** vanjski entitet (čovjek-korisnik sustava, drugi sustav, hardverski uređaja) koji ima interesa u interakciji sa sustavom.



Dijagram načina korištenja

2.3 Dinamičko modeliranje

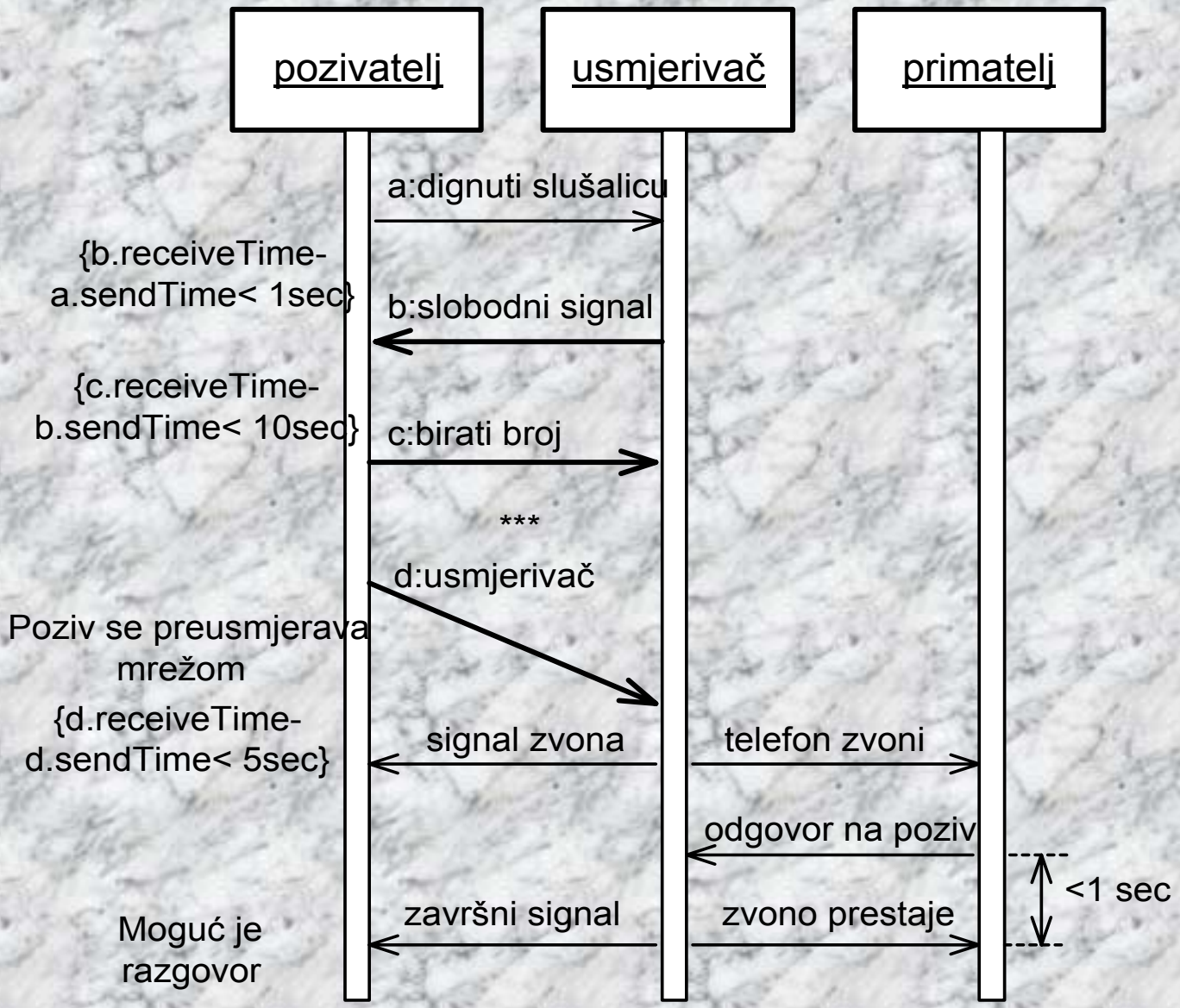
Objekti komuniciraju jedni sa drugima unutar sustava međusobno razmjenjujući poruke i tipične operacije.

Specifični oblik komunikacije između objekata kojom se generirala neka funkcija - **interakcija**.

2.3.1 Dijagram slijeda

- prikazuje eksplicitni slijed poticaja i služi za specifikaciju interakcije i komunikacije objekata u realnom vremenu.
- ima dvije dimenzije:
  - vertikalnu - predstavlja vrijeme
  - horizontalnu - prikazuje različite instance.

- **Forme** dijagrama slijeda:
  - **opća** forma - opisuje sve moguće sljedove
  - forma **instance** - opisuje određen slijed u suglasnosti sa općom formom.



Dijagram slijeda sa konkurentnim objektima

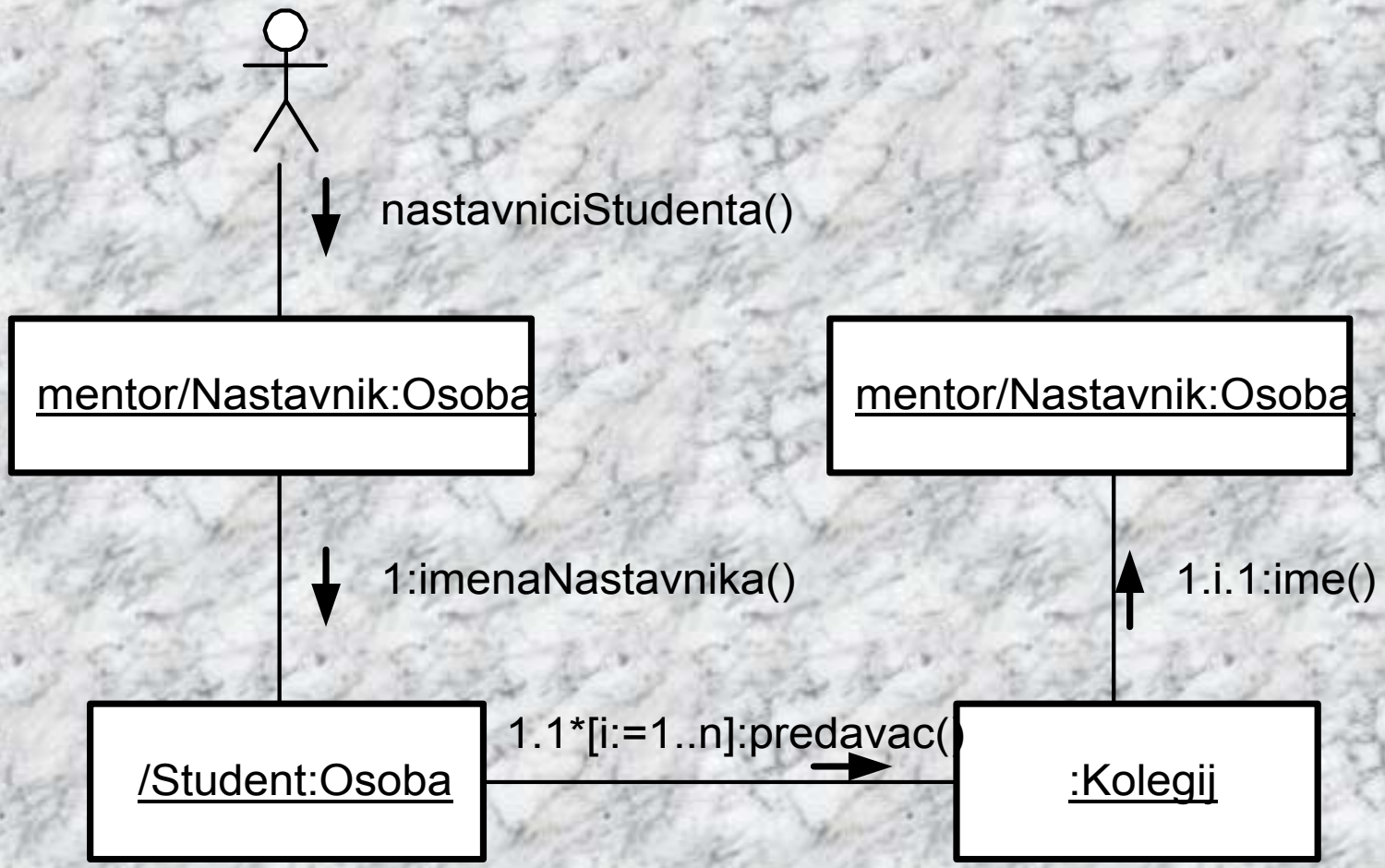
2.3.2 Dijagram suradnje

- Sadrži skup djelomično složenih poruka tako da svaka specificira jednu **komunikaciju** - poticaj koji će uzrokovati poziv operacije, signalizaciju, kreiranje ili uništenje instance i uloge koje će pritom imati pošiljatelj i primatelj

Razine apstrakcije:

- **specijalizacijska** (prikazuje interakciju učesnika sa različitim ulogama i njihove međusobne veza smislene za dani skup ciljeva; uloge klasifikatora i asocijacija i njihovu strukturu definiranu u odgovarajućoj suradnji)
- razina **instance** (uloge učesnika imaju instance u međusobnoj interakciji - šalju informacije i poticaje)

Strelica iznad linije veze ili uloge asocijacije prikazuje smjer protoka informacije i poticaj koji slijedi u danom smjeru



Dijagram suradnje na razini instance

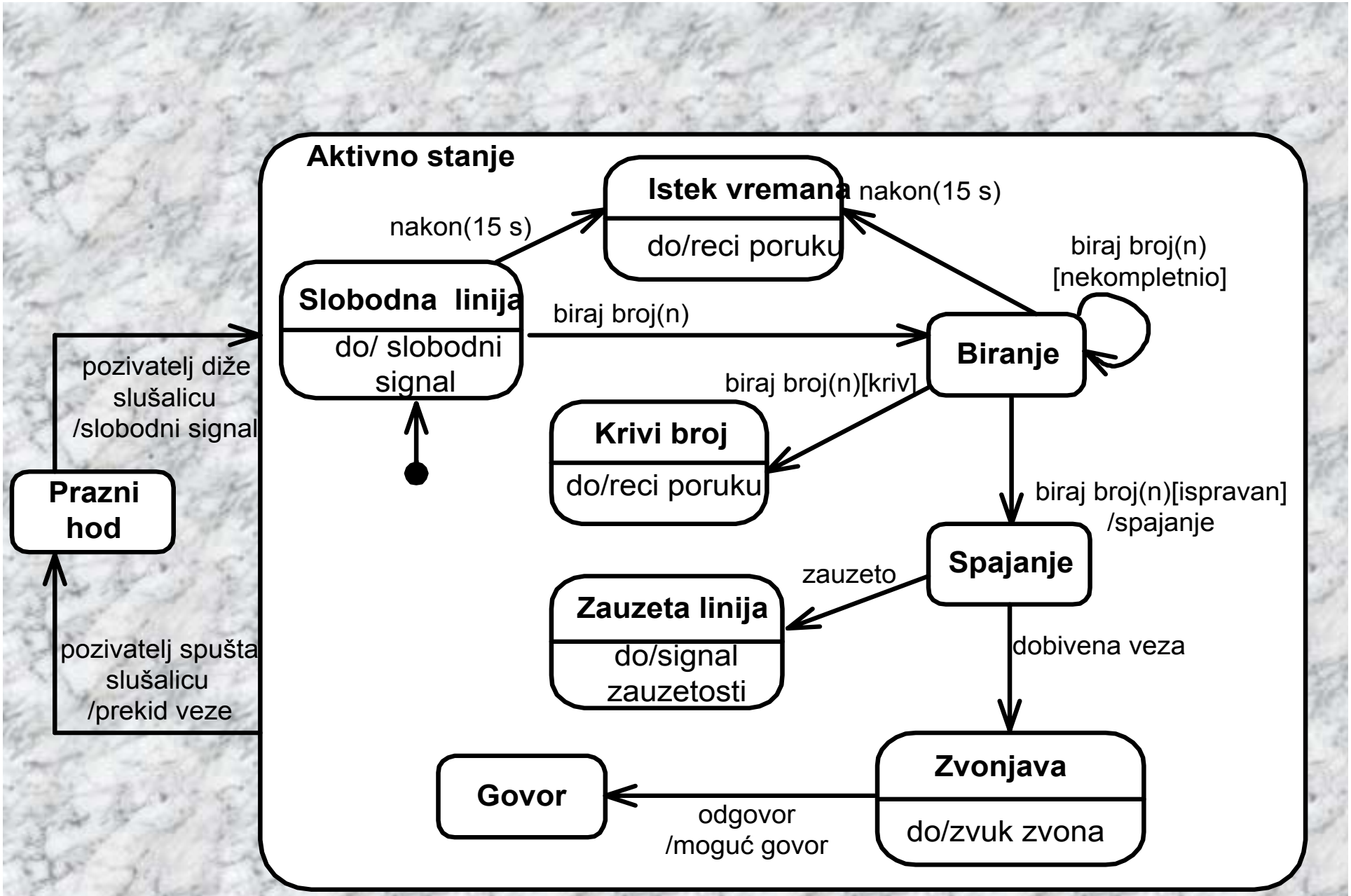
2.3.3 Dijagrami stanja

- graf koji prikazuje mehanizam stanja

- prikaz dinamičkog ponašanja elementa modela (klasa, način korištenja, učesnik, …).

- opisuje mogući slijed stanja i akcija kroz koje element može prolaziti za vrijeme života, a koje su rezultat reakcije na primljenu instancu događaja (signali, pozivi operacija)

- prikazuje sva moguća stanja koja objekt klase može imati i događaje koje prouzrokuje kada se stanje mijenja



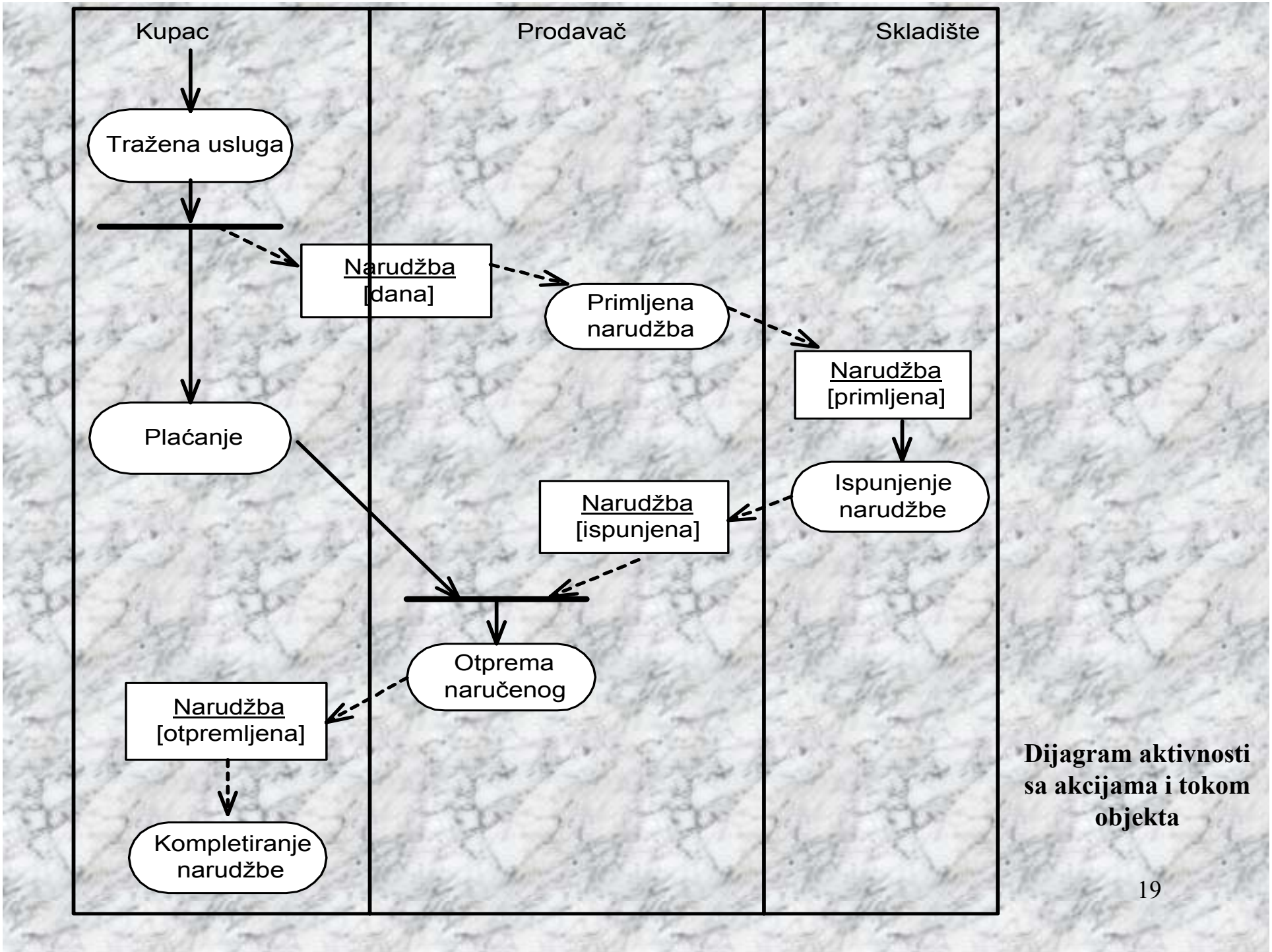
Dijagram stanja

2.3.4 Dijagram aktivnosti

- specijalni slučaj dijagrama stanja u kojem su sva stanja akcije ili podakcije, a promjena je inicirana završetkom akcije ili podakcije u izvornom stanju.

- koristi se:
  - u slučajevima gdje svi ili većina događaja prikazuje završetak interno generiranih akcija,
  - za opisivanje aktivnosti koje se izvode u nekoj operaciji,
  - za opis način korištenja
  - za opis interakcija.

Stanje objekta može biti prikazano u uglatim zagradama ispod imena objekta kao npr.[planirano], [kupljeno], [ispunjeno], itd.



Dijagram aktivnosti sa akcijama i tokom objekta

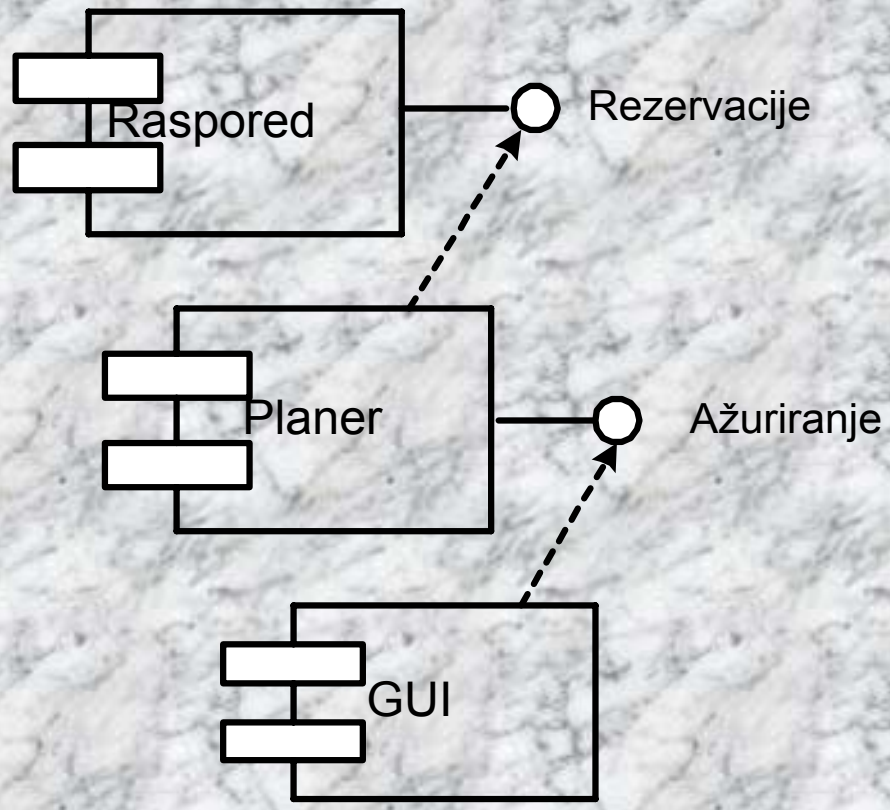
2.4 Implementacijski dijagrami

Dijagrami implementacije prikazuju strukturu programskog koda i strukturu implementacije u realnom vremenau.

2.4.1 Dijagram komponenata

- prikazuje:
  - zavisnost programskih komponenata (komponente programskog i binarnog koda i izvršne komponente)
  - fizičku strukturu koda u terminima kodnih komponenti (komponente u vremenu prevođenja, u vremenu povezivanja, vremenu izvršavanja ili u nekoliko od navedenih vremena).

- graf komponenti povezanih zavisnim vezama kako bi se omogućila lakša analiza reakcije ostalih komponenti na promjene u jednoj komponenti



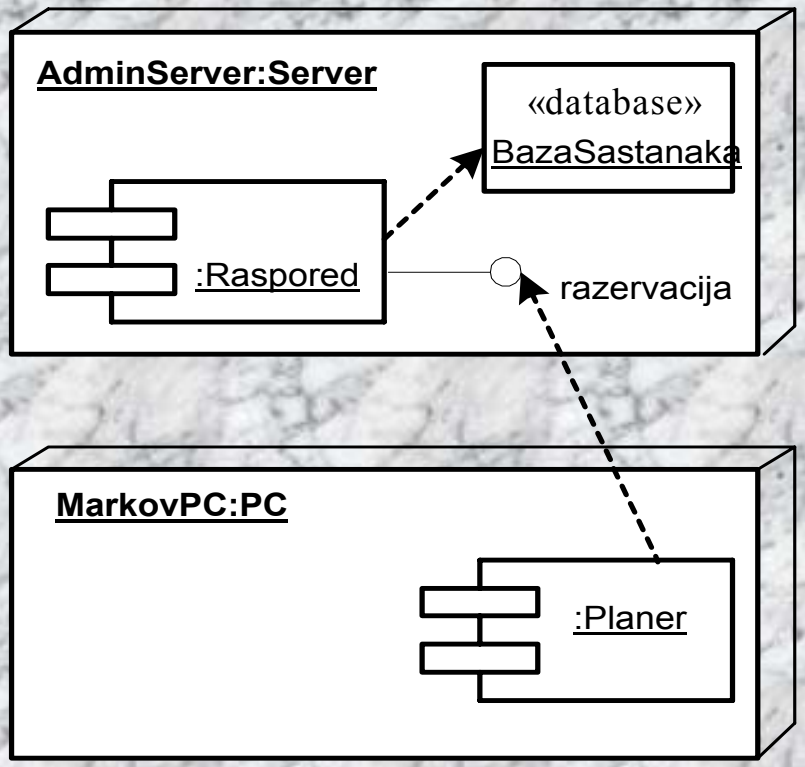
Dijagram komponenata

2.4.2 Dijagram rasporeda

- graf čvorova povezanih komunikacijom
- čvor:
  - fizički objekt koji prikazuje izvor procesa, a ima memoriju i sposobnost procesiranja
  - prikazuju uređaje, ljudske izvore i mehaničko procesiranje (alokacijom izvršnih komponenti i objekata prikazuje se koja se programska jedinica izvodi na kojem uređaju.)

- prikazuje:
  - fizičku arhitekturu elemenata procesa u realnom vremenu,
  - arhitekturu programskih komponenti koje postoje u realnom vremenu- očitovanje jedinica koda u realnom vremenu i objekata koji na njima žive.

Komponente koje ne postoje u realnom vremenu pokazuju se na dijagramu komponenata !



Dijagram rasporeda

3. Zaključak

- raznolikost dijagrama UML-a i relativna ortogonalnost pogleda, omogućuje modeliranje jednostavnih, ali i vrlo složenih aplikacija.
- dijagrami omogućuju višestruki pogled na sustav koji se analizira
- dijagrami su međusobno povezani, a njihova interakcija olakšava verifikaciju dijagrama i povećava kvalitetu softvera.

- od 1997. kada je UML izabran za standard OMG-a, razvijeno je mnogo CASE alata koji ga podržavaju
- u njegovo testiranje, poboljšanje, unaprijeđenje i nadogradnju uključili su se i partneri OMG-a
- UML predstavlja budućnost -trend njegovog razvoja i upotrebe će se nastaviti