Projektovanje softvera

Dijagrami interakcije

Uvod

- Interakcija je ponašanje koje obuhvata skup poruka koje se razmenjuju između skupa učesnika u nekom kontekstu sa nekom namenom
  - kontekst – saradnja (kolaboracija) učesnika
  - učesnici (prototipski objekti) igraju odgovarajuće uloge u saradnji
- UML 2.1: Interakcija je jedinca ponašanja koja se fokusira na uočljivu razmenu informacija između povezivih elemenata
- Interakcija se koristi za modelovanje dinamičkih aspekata modela
- Objektni dijagram je reprezentacija statičkih aspekata komunikacije
  - prezentiraju se samo objekti i veze između objekata u jednom trenutku
- Interakcija uvodi dinamički aspekat komunikacije
  - prezentira se i sekvenca poruka koje razmenjuju učesnici

Kontekst interakcije

- Kontekst – sistem ili podsistem kao celina
  - interakcije su u saradnji objekata koji postoje u sistemu ili podsistemu
  - primer – sistem za e-trgovinu:
    sarađuju objekti na strani klijenta sa objektima na strani servera
- Kontekst – operacija
  - interakcije su među objektima koji implementiraju operaciju
  - tekući objekat, parametri operacije, lokalni i globalni objekti
    mogu da interaguju da bi izvršili algoritam operacije
- Kontekst – klasa
  - atributi klase mogu da sarađuju međusobno kao i sa drugim objektima
    (globalnim, lokalnim i parametrima operacija)
  - interakcija se koristi da opiše semantiku klase
- Kontekst – slučaj korišćenja
  - interakcija reprezentuje scenario za slučaj korišćenja

Poruka

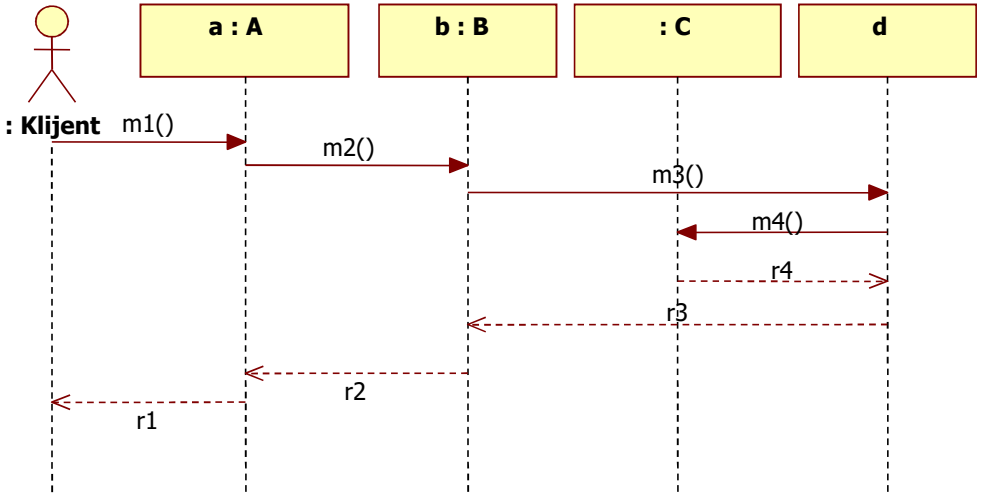
- Poruka je specifikacija komunikacije između objekata koja prenosi informaciju, iza koje se očekuje da sledi aktivnost
  - aktivnost se dešava na strani prijema poruke
  - slanje poruke je samo stimulus za aktivnost
- Poruka može da bude:
  - asinhrona (slanje signala)
  - sinhrona (poziv operacije)
- Poruka se prikazuje kao strelica na dijagramu interakcije
  - razne vrste strelica odgovaraju raznim vrstama poruka
  - na dijagramu sekvence – poruke su linije između linija života
  - na dijagramu komunikacije – poruke su strelice pored konektora

Vrste dijagrama interakcije

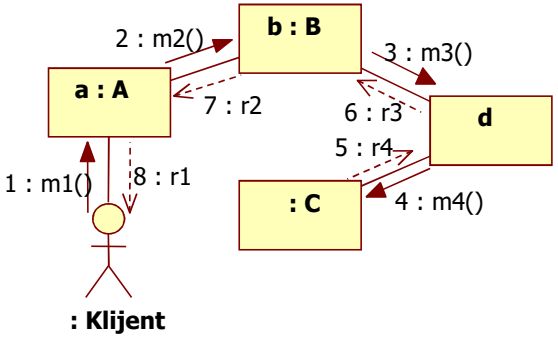
- UML 2 – 4 vrste dijagrama interakcije
- Dijagrami sekvence naglašavaju vremensko uređenje interakcije
- Dijagrami komunikacije
  naglašavaju strukturu veza između učesnika u interakciji
  - u UML-u 1 nazivali su se dijagramima saradnje (kolaboracije)
  - vizuelizuju na različit način skoro iste informacije kao d. sekvence
- Dijagrami pregleda interakcije (UML 2)
  kombinuju dijagram aktivnosti sa dijagramima sekvence
  - u okviru toka kontrole, blokovi (čvorovi) se opisuju interakcijama
- Vremenski dijagrami (UML 2)
  prikazuju promenu stanja učesnika u vremenu
  - promena stanja se dešava kao posledica
    - prijema poruka (stimulusa) i
    - dešavanja događaja

Dijagrami sekvence i komunikcije

Dijagram sekvence

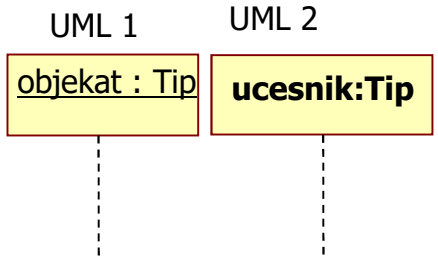


Dijagram komunikacije



Učesnici i njihove linije života

- Linija života (*lifeline*)
  - reprezent jednog učesnika (entiteta, objekta) u interakciji
  - notacija (UML 2): naziv učesnika se ne podvlači



- Učesnici u interakciji:
  - UML1: konkretni primerci (objekti) u konkretnom stanju
    - označavaju realnu stvar nekog tipa u tekućem stanju
    - na primer, konkretan objekat `o1` klase `Osoba` u tekućem stanju; `o1:Osoba`
  - UML 2: prototipske stvari - predstavnici konkretnih stvari
    - označavaju proizvoljnu stvar nekog tipa
    - na primer, `o` kao referenca na objekat klase `Osoba`; `o:Osoba`
- U interakciji mogu da se pojave i "primerci" apstraktnih klasa i interfejsa
  - takvi primerci ne označavaju konkretne stvari (nemogući su primerci apstraktnih klasa i interfejsa)
  - predstavljaju prototipske stvari – uloge učesnika (to su stvarni primerci potklasa)

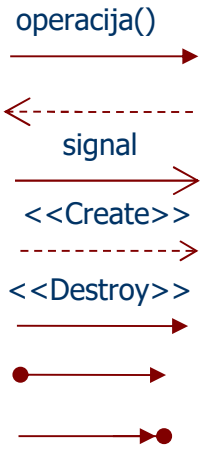
Konektori

- Na d. objekata se prikazuju primerci (objekti), a na d.interakcije – učesnici
- Na d. objekata se prikazuju veze, a na d.komunikacije – konektori
- Veza – strukturna sprega između objekata – primerak asocijacije
- Konektor – komunikaciona putanja između učesnika
  - ne mora da bude primerak asocijacije, može da se ostvaruje kao zavisnost
  - može da bude privremen, kao i veza

- Ukrasi načina pristupa drugoj strani konektora (UML 2 UG)
  - specificiraju način na koji učesnik koji šalje poruku vidi drugu stranu
  - tekstualni ukrasi koji se pišu na udaljenom kraju konektora (kod primaoca)
  - konkretni ukrasi:
    - `{association}` – drugoj strani se pristupa preko primerka asocijacije - veze
    - `{self}` – učesnik sam sebi može da šalje poruku
    - `{global}` – druga strane je u nekom okružujućem dosegu imena
    - `{local}` – druga strana je u lokalnom dosegu imena
    - `{parameter}` – druga strana je argument operacije

Slanje i prijem poruke (1)

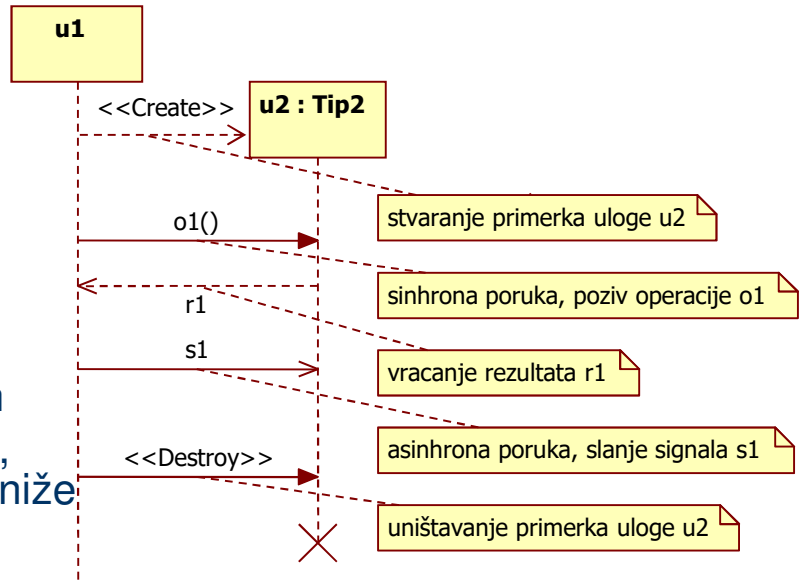
- Prijem jedne poruke može da se smatra pojavom jednog događaja
- Kada se pošalje i primi poruka, sledi aktivnost na strani prijema
  - izvršenje naredbe koja predstavlja apstrakciju metoda kod sinhrone operacije
- Postoji više različitih poruka zahteva i jedna poruka odgovora
- UML predviđa sledeće vrste poruka:
  - poziv (*call*) – pokreće operaciju uloge primaoca
  - odgovor (*reply*) – vraća vrednost pozivaocu
  - slanje (*send*) – asinhrono se šalje signal primaocu
  - stvaranje (*create*) – stvara se objekat (primerak uloge)
  - uništavanje (*destroy*) – uništava se objekat
  - pronađena poruka (*found*) – poznat primalac, slanje nije opisano
  - izgubljena poruka (*lost*) – poznat pošiljalac, prijem neodređen



Slanje i prijem poruke (2)

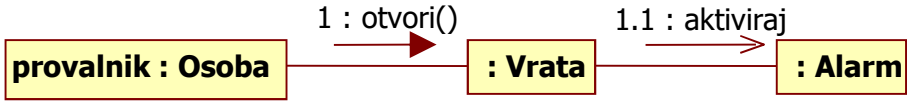
- Kod poruke vrste poziva (*call*) podrazumeva se "sinhronost":
  - pozivalac ne napreduje dok pozvani objekat ne završi obradu poziva (aktivnost)
  - cela sekvenca ugnežđenih poziva se završava pre nego što se spoljašnji nivo izvršenja nastavi
  - koristi se za:
    - proceduralne pozive u jednoj niti
    - pozive za randevu (npr. Ada) u konkurentnom okruženju

- Primer raznih vrsta poruka na dijagramu sekvence
- Poruke su načelno horizontalne
  - podrazumeva se da je stimulus trenutan
  - ako stimulus putuje (kroz mrežu) i kasni, poruka može da bude crtana i ukoso naniže



Sekvenciranje poruka

- Unutar toka kontrole neke niti – poruke su uređene u vremensku sekvencu
- Na dijagramima sekvence
  - sekvenca se modeluje implicitnoređanjem poruka odozgo-naniže
- Na dijagramima komunikacije
  - sekvenca se modeluje rednim brojem ispred dvotačke i imena
- Notacija:
  - proceduralni (ugnežđeni) tok kontrole se prikazuje strelicama sa popunjenom glavom
    - redni brojevi poruka imaju hijerarhijsku strukturu (nivoi hijerarhije se razdvajaju tačkom)
    - primer: 2.1.3:op()
  - ravni (*flat*) tok kontrole se prikazuje strelicama sa otvorenom glavom (asinhrone poruke)
    - redni brojevi poruka nemaju hijerarhijsku strukturu
    - primer: 5: s
- Primer kombinovane sekvence



Sekvenciranje poruka u nitima

- Identifikacija niti se piše iza rednog broja poruke kojom se pokreće aktivnost u kojoj se vrši konkurentno grananje:
  - primer:

    1a.5:uradi() →

    - aktivnost pokrenuta 1. porukom ima konkurentno grananje, pa se posmatra 5. poruka u niti a
  - primer:

    3b.5.2:dohvati() →

    - aktivnost pokrenuta 3. porukom se konkurentno grana, reč je o 2. poruci u okviru aktivnosti pokrenute 5. porukom u niti b

Sintaksa poruke zahteva

- Na poruci zahteva osim imena može da se prikaže i lista ulaznih i ulazno-izlaznih argumenta
- Argumenti mogu da se navode poziciono ili imenovano
  - ako se navode poziciono (navodi se samo vrednost), znak – je džoker za argument sa prozvoljnom vrednosšću
  - ako se navode imenovano (ime_parametra=vrednost), nedostajući argumenti imaju prozvoljnu vrednost
  - ne može da se kombinuje poziciono i imenovano navođenje argumenata
- Primer:
  - operacija: `+trazi(in ime: String, in pol: Pol)`
  - `trazi("Saša",-)` – poziciono navođenje argumenata, nebitan pol
  - `trazi(ime="Saša")` – imenovano navođenje parametara, nebitan pol

Sintaksa poruke odgovora

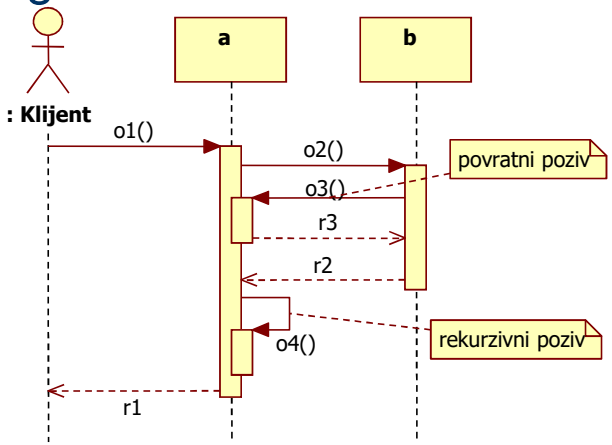
- Na poruci odgovora može da se prikaže i
  - lista izlaznih i ulazno-izlaznih argumenta
  - vraćena vrednost
  - pridruživanje vraćene vrednosti cilju
    - cilj je po pravilu (ne mora da bude) atribut klase pošiljaoca
    - cilj može da se koristi kao argument neke naredne poruke
- Primer:
  - operacija:
    `+traziIme(in matBroj:String, return ime:String):String`
  - poruka odgovora:
    `trazeni.ime=trazi():imeNadjeneOsobe`
  - kombinovana poruka zahteva sa odgovorom (nije po standardu):
    `trazeni.ime=trazi(matBroj="1234567890123"):imeNadjeneOsobe`

Životni vek učesnika i veza

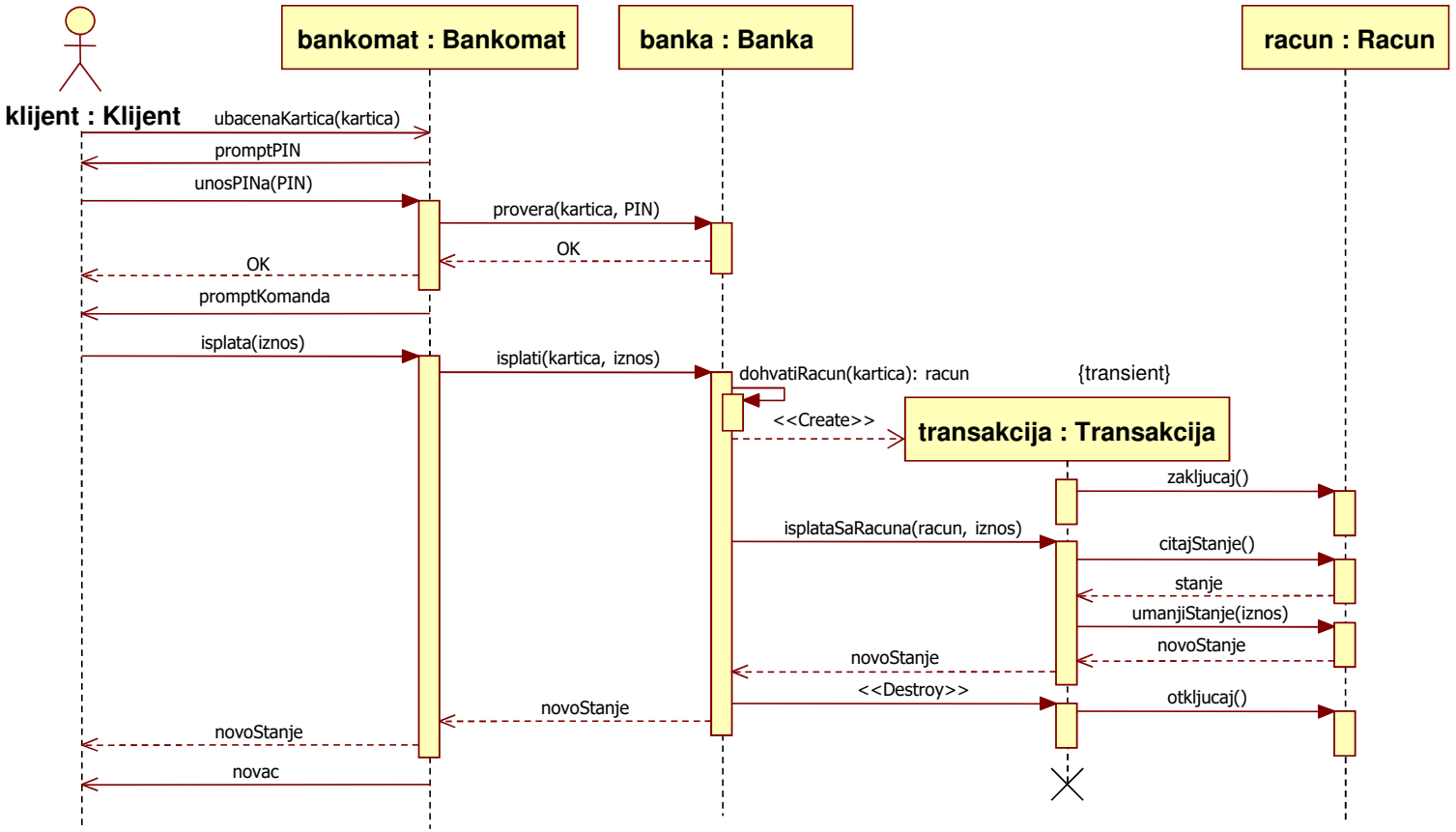
- Nekad se životni vek učesnika ne poklapa sa trajanjem interakcije
- Slično važi za veze koje se uspostavljaju preko konketora
- Učesnici i veze mogu da nastaju i nestaju u toku interakcije
- Ograničenja životnog veka koja mogu da se označe na učesniku i/ili konektoru preko kojeg se ostvaruje veza (UML 1 UG)
  - `{new}` – uloga/veza se kreira za vreme izvršenja interakcije
  - `{destroyed}` – uloga/veza se uništava pre završetka interakcije
  - `{transient}` – uloga/veza se kreira i uništava za vreme interakcije
- Ima smisla na dijagramima komunikacije, na dijagramu sekvence se predstavlja grafički
- Promena stanja ili uloge učesnika na dijagramu interakcije može da se naznači njegovom replikacijom (UML 1 UG)
  - na dijagramu sekvence: ponavljanje učesnika u novoj ulozi ili stanju na istoj (vertikalnoj) liniji života
  - na dijagramu komunikacije: isti učesnik u novoj ulozi ili stanju se povezuje porukom `<<become>>`

Specifikacija izvršenja

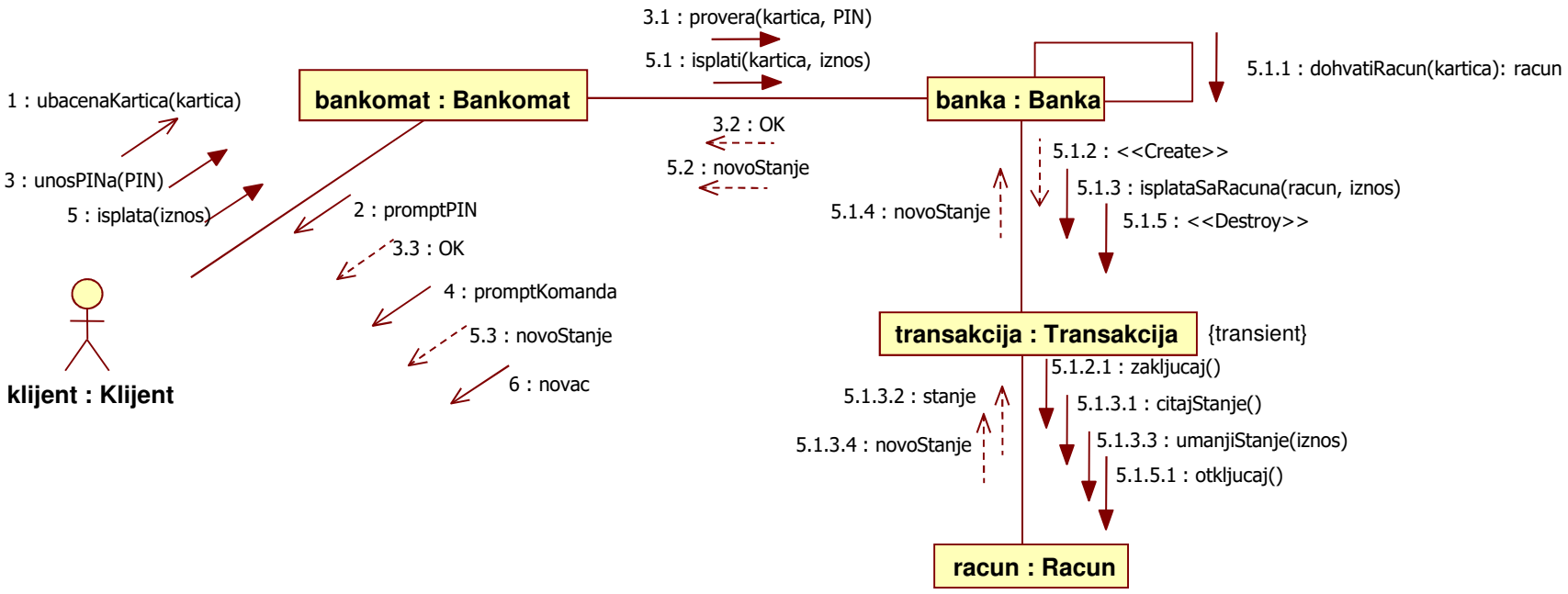
- Raniji termini: događanje izvršenja, fokus kontrole
- Može da se naznači samo na dijagramima sekvence
- Grafička notacija:
  - uski pravougaonik na liniji života
  - definiše period u kojem uloga obavlja aktivnost izazvanu porukom
- Moguće je i ugnežđivanje iz sledećih razloga:
  - (A) zbog rekurzije
    (poziva iste ili druge sopstvene operacije)
  - (B) zbog povratnog poziva (*call back*)
    od pozvanog objekta



Primer Bankomat – d. sekvence



Primer Bankomat – d. komunikacije

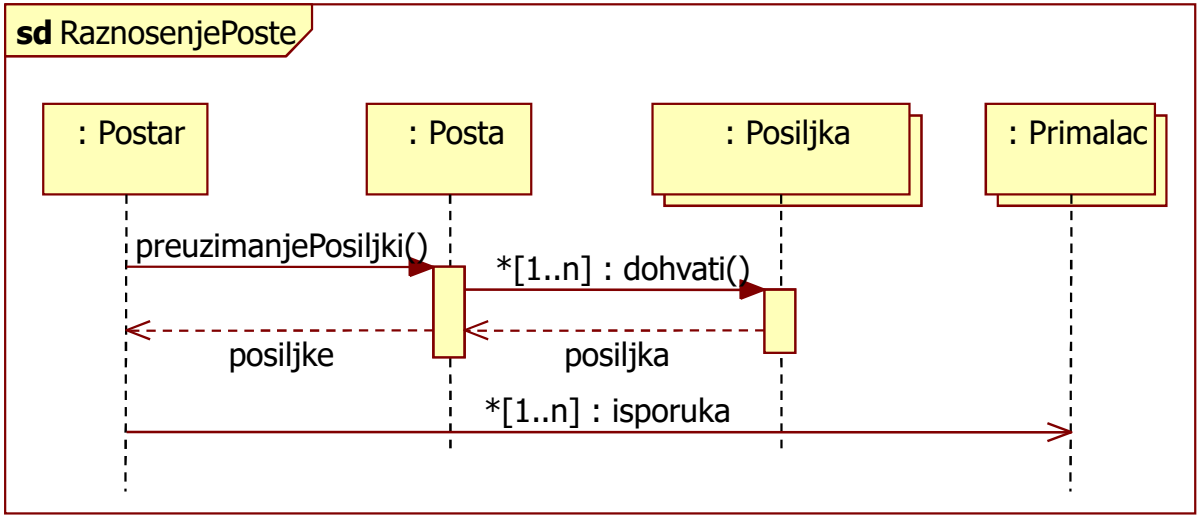


Iteracije i grananje

- Izrazi za iteracije i grananje se pišu iza broja za sekvenciranje poruke, na bilo kom nivou ugnežđenja
- Izraz za sekvencijalne iteracije:
  - brojač ponavljanja `*[i:=1..n]` ili uslov `*[a>b]` ili samo `*`
  - poruka sa r.br. ispred izraza se ponavlja u skladu sa izrazom
- Oznaka za paralelne iteracije:
  - `*||`
- Izraz za grananje:
  - `[x>0]`
  - dve ili više poruka u sekvenci mogu da imaju isti redni broj, ali onda moraju da imaju disjunktne uslove

Fragment (okvir) interakcije

- Fragment interakcije je najopštija jedinica interakcije
- Opisuje deo interakcije i konceptualno je isti kao i sama interakcija
- Notacija: okvir sa jezičkom naniže u kojem stoji `sd` <ime interakcije>



- Višestruki primerci (multiobjekti) – UML1, zastareli u UML2

Operatori kombinovanih fragmenata

- Opšti
  - `sd` - dijagram sekvence ili komunikacije (uokviruje ceo dijagram)
  - `neg` - fragment prikazuje pogrešnu (negativnu) interakciju
  - `ref` - fragment je referenca na interakciju definisanu na drugom dijagramu
- Kontrola sekvencijalnog toka
  - `opt` - opcioni fragment – izvršava se samo ako je ispunjen uslov
  - `alt` - alternativni izbor između više fragmenata
  - `loop` - fragment se izvršava više puta (petlja)
  - `break` - fragment se izvršava umesto ostatka okružujućeg fragmenta
- Kontrola paralelnih tokova
  - `par` - paralelno se izvršavaju fragmenti
  - `region` - kritični region – u fragmentu ne može da se istovremeno izvršava više niti

Primer operatora (1)

- Distribucija porudžbina
  - neka uloga tipa `:Porudzbina` ima operaciju `isporuka()`

```
procedure isporuka()
    foreach (stavka)
        if (vrednost<=1000)
            redovniDistributer.isporuci()
        else
            specijalniDistributer.isporuci()
        endif
    endfor
    if (potrebnaPotvrda) kurir.potvrdi() end if
end procedure
```

  - neka akter `:Prodavac` pokreće isporuku porudžbine signalom `isporuka`
  - neka je cela komunikacija asinhrona

Primer operatora (2)