

Projektovanje softvera

Dijagrami interakcije



Uvod

- Interakcija je ponašanje koje obuhvata skup poruka koje se razmenjuju između skupa objekata u nekom kontekstu sa nekom namenom
- UML 2.0: interakcija je specifikacija slanja stimulusa između objekata sa ciljem obavljanja nekog zadatka
 - definiše se u kontekstu neke saradnje (kolaboracije)
- Interakcija se koristi za modeliranje dinamičkih aspekata modela
- Objektni dijagram je reprezentacija statičkih aspekata komunikacije
 - prezentiraju se samo objekti i veze između objekata u jednom trenutku
- Interakcija uvodi dinamički aspekt komunikacije
 - prezentira se i sekvenca poruka koje razmenjuju uloge

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 interagovati da izvrše algoritam operacije
- Kontekst – klasa
 - atributi klase mogu sarađivati 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

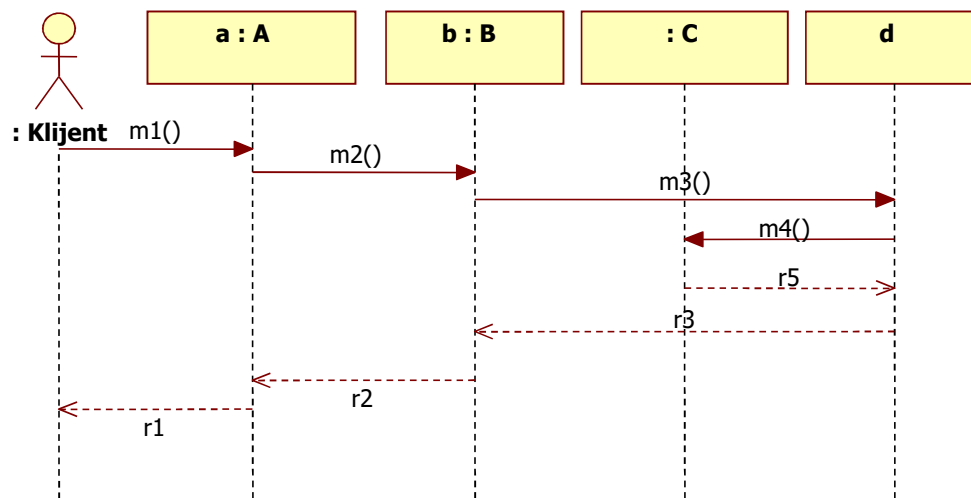
- Poruka je specifikacija komunikacije između objekata koja prenosi informaciju, iza koje se očekuje da sledi aktivnost
- Slanje poruke uložni označava samo stimulus za aktivnost
- Poruka može biti
 - asinhrona (slanje signala)
 - sinhrona (poziv operacije)
- Poruka se prikazuje kao strelica na dijagramu 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

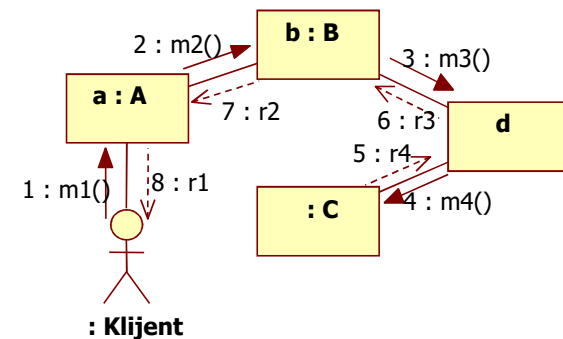
- UML 2 – 4 vrste dijagrama interakcije
- Dijagrami sekvence naglašavaju vremensko uređenje interakcije
- Dijagrami komunikacije naglašavaju strukturu veza između učesnika u interakciji
 - u UML-u 1 nazivali su se dijagramima saradnje (kolaboracije)
 - vizuelizuju na različit način praktično iste informacije 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 prikazuju promenu stanja objekta (ili uloge) u vremenu
 - promena stanja se dešava kao posledica
 - prijema poruka (stimulusa) i
 - dešavanja događaja

Dijagrami sekvence i komunikacije

Dijagram sekvence

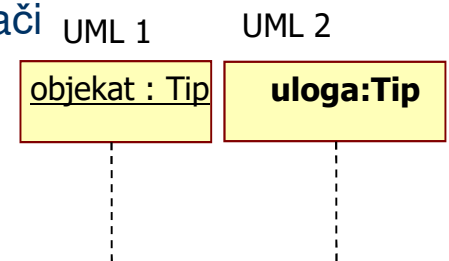


Dijagram komunikacije



Uloge i njihove linije života

- Linija života (*lifeline*)
 - reprezent jednog učesnika (entiteta, objekta) u interakciji
 - notacija (UML 2): naziv uloge (na liniji života) se ne podvlači
- Učesnici u jednoj interakciji mogu biti:
 - UML1: objekti - konkretni primerci u konkretnom stanju
 - označavaju realnu stvar nekog tipa u tekućem stanju
 - na primer, konkretan objekat osobe klase `Osoba`
 - UML 2: uloge - prototipske stvari predstavnici konkretnih stvari
 - označavaju proizvoljnu stvar nekog tipa
 - na primer, referenca na objekat klase `Osoba`: `Osoba o`;
- U interakciji se mogu pojaviti “primerci” apstraktnih klasa i interfejsa
 - takvi primerci ne označavaju konkretne stvari (nemogući su primerci apstraktnih klasa i interfejsa)
 - predstavljaju prototipske stvari – uloge (to su primerci potklasa)



Konektori

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

uloga1 : Klasa1

uloga2 : Klasa2

- Ukrasi načina pristupa drugoj strani konektora
 - specificiraju način na koji uloga koja šalje poruku vidi drugu stranu
 - tekstualni ukrasi koji se pišu na udaljenom kraju konektora (kod primaoca)
 - konkretni ukrasi:
 - `{association}` – objektu uloge se pristupa preko primerka asocijacije - veze
 - `{self}` – objekat uloge sam sebi može da šalje poruku
 - `{global}` – uloga je u nekom okružujućem doseg imena
 - `{local}` – uloga je u lokalnom doseg imena
 - `{parameter}` – uloga je argument operacije

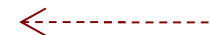
Dijagrami interakcije

25.10.2018.

Slanje i prijem poruke (1)

- Prijem jedne poruke se može smatrati pojavom jednog događaja
- Kada se pošalje i primi poruka, sledi aktivnost na strani prijema
 - izvršenje naredbe koja predstavlja apstrakciju metoda kod sinhronne operacije
- UML predviđa sledeće vrste poruka:
 - poziv (*call*) – pokreće operaciju uloge primaoca
 - povratak (*return*) – vraća vrednost pozivaocu
 - slanje (*send*) – asinhrono se šalje signal primaocu
 - kreiranje (*create*) – kreira se objekat (primerak uloge)
 - uništavanje (*destroy*) – uništava se objekat
 - pronađena poruka (*found*) – poznat primalac, slanje nije opisano
 - izgubljena poruka (*lost*) – poznat pošiljalac, prijem neodređen

operacija()



signal



<<create>>

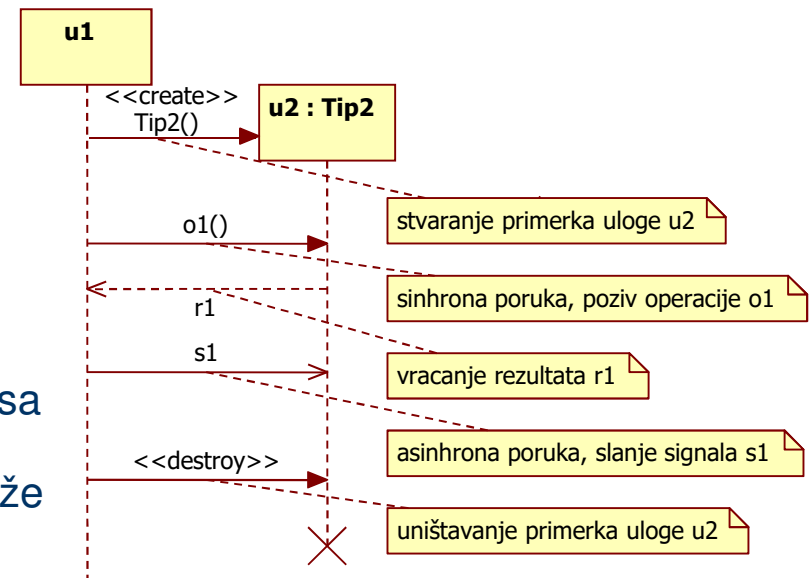


<<destroy>>



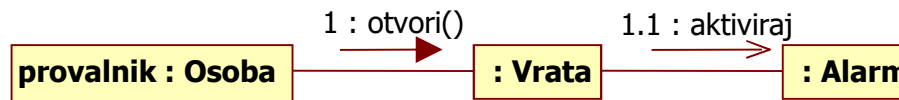
Slanje i prijem poruke (2)

- Kod poruke vrste poziva (*call*) podrazumeva se "sinhronost":
 - pozivalac ne napreduje dok pozvani objekat ne završi obradu poziva
 - cela sekvenca ugneždenih 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 višeprocenom okruženju
- Primer raznih vrsta poruka na dijagramu sekvence:
- Poruke su horizontalne
 - podrazumeva se atomičnost stimulusa
 - ako se ne može smatrati atomičnim, poruka može biti crtana i ukoso naniže



Sekvenciranje poruka

- Unutar toka kontrole neke niti – poruke su uređene u vremensku sekvencu
- Na dijagramima sekvence
 - sekvenca se modelira implicitno ređanjem poruka odozgo-naniže
- Na dijagramima komunikacije
 - sekvenca se modelira rednim brojem ispred imena
- Grafička 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 običnim strelicama (asinhrono poruke - signali)
 - 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

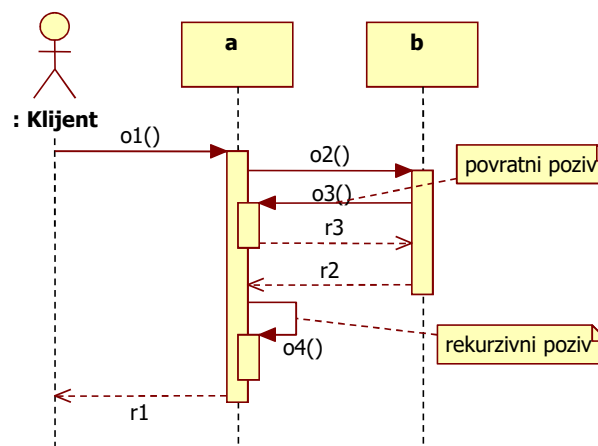
- Na poruci se osim imena mogu prikazati i
 - njeni (stvarni) argumenti
 - vraćena vrednost
 - pridruživanje vraćene vrednosti promenljivoj
 - vrednost se može koristiti kao argument neke naredne poruke
 - promenljiva je po pravilu (ne mora biti) atribut klase pošiljaoca
 - primer: `1.2:prosekGod=uspeh(godStud):srednjaOcena`
- Argumenti se mogu pisati i sa imenom parametra
 - primer: `5:trazenaOsoba=trazi(ime="Petar Petrović")`

Životni vek objekata i veza

- Po nekad se životni vek objekta ili veze ne poklapa sa trajanjem interakcije
- Objekti i veze mogu nastajati i nestajati u toku interakcije
- Sledeća ograničenja se mogu pripisati objektu i/ili vezi (UML 1)
 - {new} – objekat/veza se kreira za vreme izvršenja interakcije
 - {destroyed} – objekat/veza se uništava pre završetka interakcije
 - {transient} – objekat/veza se kreira i uništava za vreme interakcije
- Promena stanja ili uloge objekta na dijagramu interakcije se naznači njegovom replikacijom (UML 1)
 - na dijagramu sekvence sve varijante jednog objekta se smeštaju na istu vertikalnu liniju
 - na dijagramu komunikacije varijante se povezuju porukom <<become>>

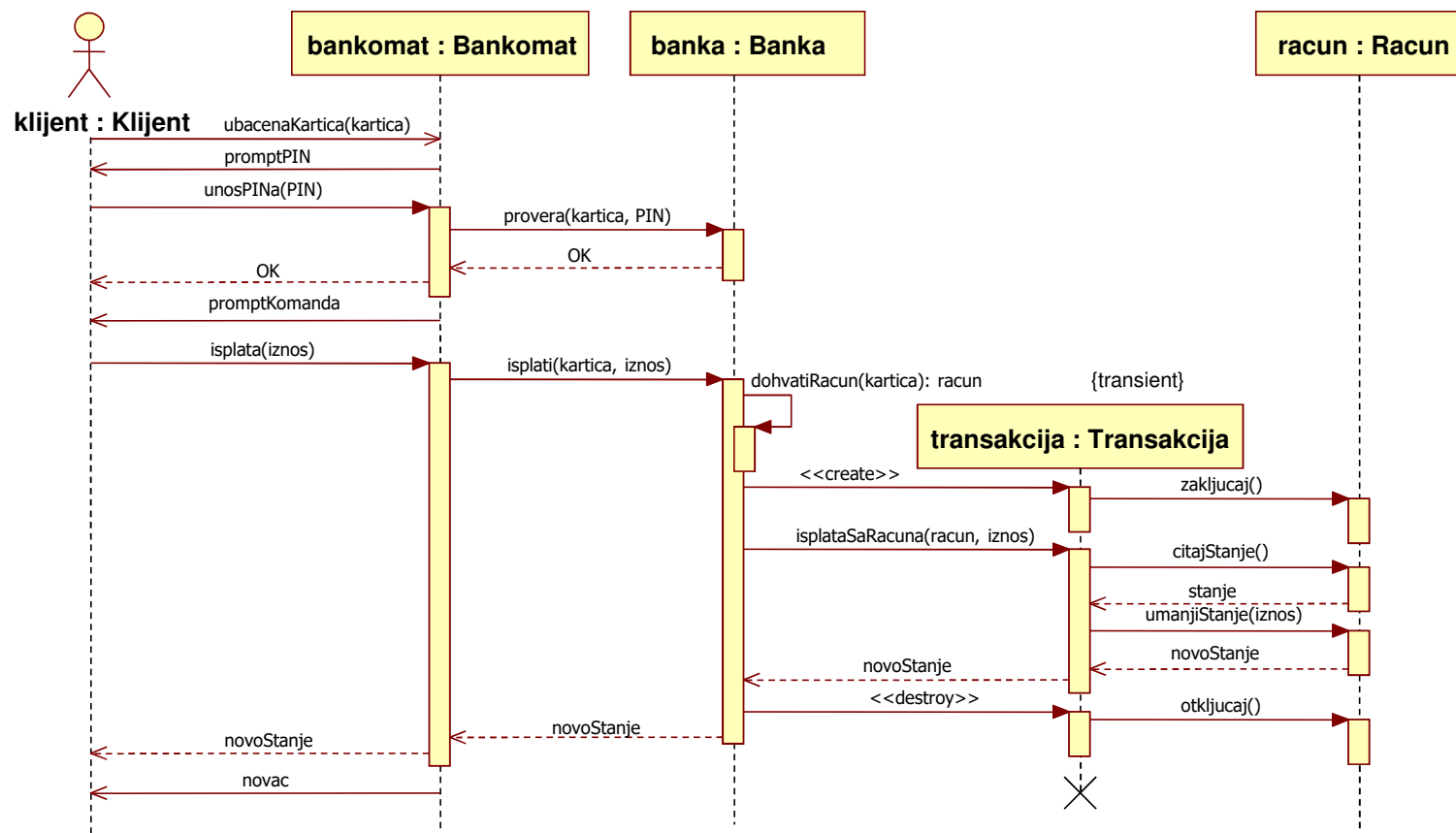
Događanje izvršenja (fokus kontrole)

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

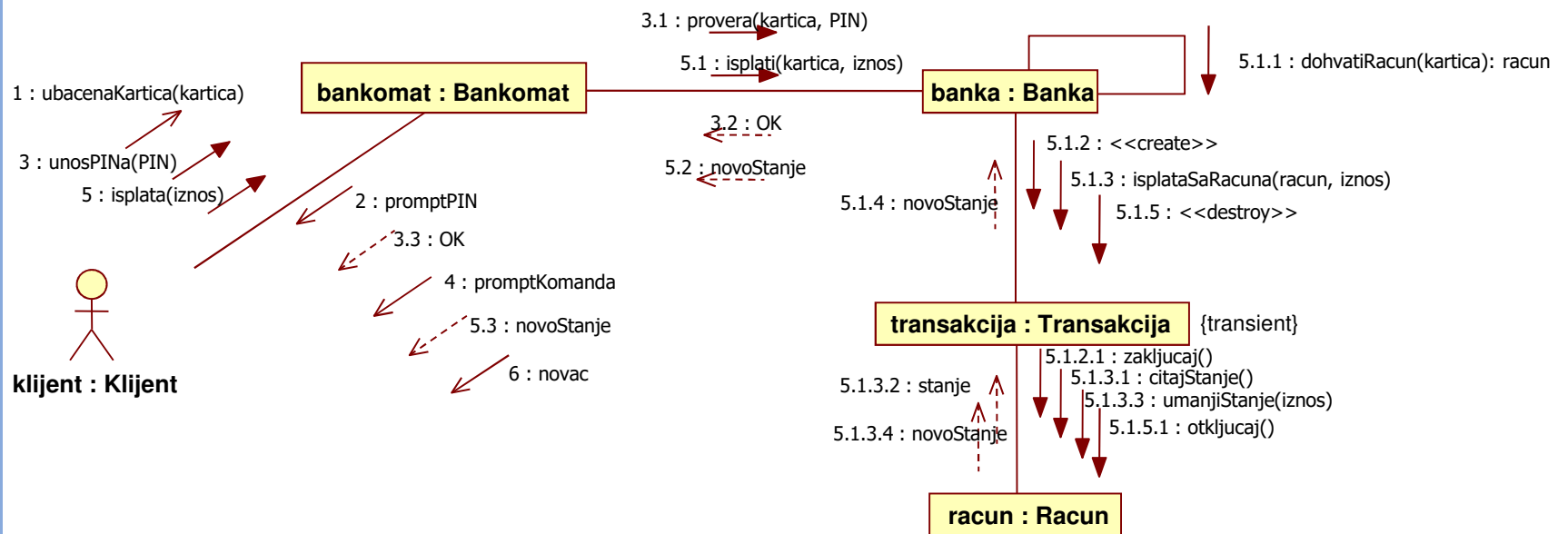


Dijagrami interakcije

Primer Bankomat – d. sekvence



Primer Bankomat – d. komunikacije

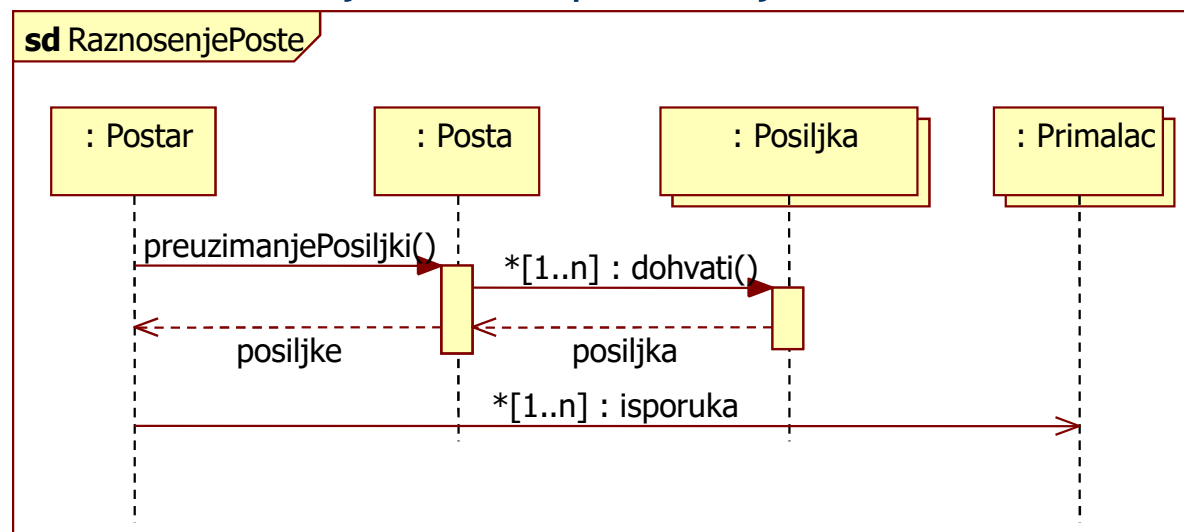


Iteracije i grananje

- Izrazi za iteracije i grananje se pišu iza broja za sekvenciranje poruke, na bilo kom nivou ugnežđenja
- Izraz za sekvencijalne iteracije:
 - brojač ponavljanja * $[i := 1 \dots n]$ 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 imati isti redni broj, ali onda moraju imati disjunktne uslove

Fragment (okvir) interakcije

- Fragment interakcije je najopštija jedinica interakcije
- Opisuje deo interakcije i konceptualno je isti kao i sama interakcija



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

Operatori kombinovanih fragmenata

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

Primer operatora (1)

- Distribucija porudžbina

- neka uloga tipa `:Porudzbina` ima operaciju `slanje()`

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

- neka akter `:Prodavac` pokreće `slanje` porudžbina signalom
- neka je cela komunikacija asinhrona

Primer operatora (2)

