SOFTVERSKO INŽENJERSTVO

školska 2024/2025 godina

Vežba 5: Sekvencijalni dijagrami

Sekvencijalni dijagrami (engl. *Sequence Diagrams*) su **interaktivni UML dijagrami** koji omogućuju detaljno modeliranje događaja unutar funkcionalnosti sistema. Ovi dijagrami prikazuju kako objekti međusobno komuniciraju kroz razmenu poruka u vremenskom toku, pružajući vizualnu predstavu o tome kako se akcije i procesi odvijaju unutar sistema.

Svaka poruka koja se šalje između objekata označava specifičnu akciju, kao što su metode, funkcije ili procesi koji se pokreću, i sve to u datoj vremenskoj liniji. Sekvencijalni dijagrami se često koriste kako bi se dobio uvid u međusobne interakcije između sistema i korisnika ili između različitih komponenata unutar sistema.

Za razliku od **Use Case dijagrama**, koji se fokusira na to šta korisnik želi da postigne u okviru sistema, sekvencijalni dijagrami prikazuju kako se ta funkcionalnost implementira u praksi, korak po korak. To znači da ne samo da pokazuju koji su to ciljevi korisnika, već i način na koji se ti ciljevi ostvaruju kroz konkretne interakcije među objektima.

Na ovaj način, sekvencijalni dijagrami služe kao ključni alat za analizu i detaljno planiranje interakcija između različitih entiteta sistema, pomažući u identifikaciji potencijalnih problema i optimizaciji procesa.

Koriste se za:

- **Analizu toka komunikacije među komponentama sistema**
  Prikazuju kako komponente sistema međusobno razmenjuju podatke, što je esencijalno za razumevanje kako se funkcionalnosti izvode.
- **Detaljnu specifikaciju procesa ili slučajeva upotrebe**
  Omogućavaju razumevanje specifičnih koraka koji se preduzimaju tokom izvršenja određene funkcionalnosti ili scenarija.
- **Prikaz ponašanja objekata u određenom scenariju**
  Ilustruju kako objekti ili entiteti u sistemu reaguju na određene ulazne podatke i kako dolazi do promene stanja sistema.

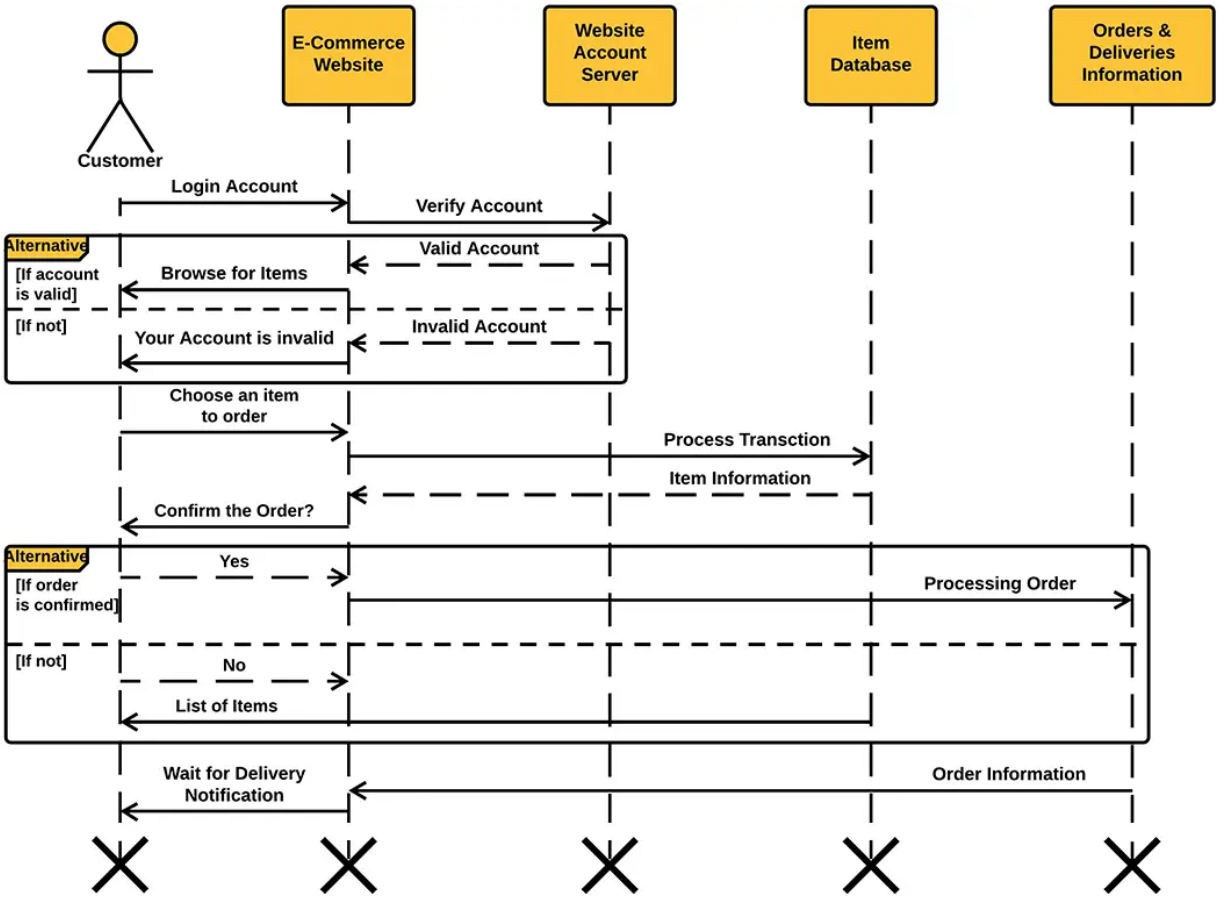
🎯 Ciljevi sekvencijalnih dijagrama

- **Vizualizacija toka izvršavanja funkcionalnosti**
  Prikazuju konkretan redosled akcija unutar sistema, kao što je proces prijave korisnika, kroz jasno definisane poruke između objekata.

- **Modelovanje međusobne komunikacije objekata**
  Dijagram detaljno prikazuje kako objekti međusobno komuniciraju, koje poruke šalju i kako se te poruke razmenjuju u vremenskom okviru.

- **Specifikacija dinamičkog ponašanja Sistema**
  Sekvencijalni dijagrami se razlikuju od klasnih dijagrama po tome što prikazuju kako objekti međusobno deluju u određenom vremenskom periodu, a ne samo statičku strukturu sistema.

- **Podrška implementaciji**
  Dijagrami pružaju jasnu sliku programerima o tome kako se objekti ponašaju i kako sarađuju prilikom izvršavanja funkcionalnosti, što im pomaže u implementaciji i debagovanju sistema.



🔧 Glavni elementi sekvencijalnog dijagrama

📌 1. Objekti (Učesnici u komunikaciji)

- Objekti su predstavljeni pravougaonicima sa nazivom objekta (npr. *Korisnik*, *AutentikacioniServis*, *BazaPodataka*).
- Na vrhu dijagrama su horizontalno raspoređeni, svaki objekat zauzima svoj prostor u dijagramu, a njihova interakcija se prikazuje kroz strelice.
- Svaki objekat ima svoju liniju života, tj. vertikalnu liniju koja se prostire nadole, što simbolizuje njegovu aktivnost tokom vremena i trajanje postojanja objekta u procesu.

📌 2. Poruke (Messages)

- Poruke predstavljaju komunikaciju između objekata, a u sekvencijalnim dijagramima su prikazane strelicama koje povezuju linije života objekata. Strelice mogu označavati različite vrste komunikacije:
    - **Pune strelice** označavaju **sinhroni poziv funkcije**, što znači da objekat koji šalje poruku mora čekati na odgovor pre nego što nastavi dalje. Na primer, to može biti metoda koja zahteva povratnu vrednost pre nego što objekat može da pređe na sledeći korak.
    - **Isprekidane strelice** označavaju **asinhroni poziv ili povratnu vrednost**, što znači da objekat može nastaviti sa radom bez čekanja na odgovor, a odgovor može doći kasnije.

📌 3. Linije života (Lifelines)

- Linije života su vertikalne isprekidane linije koje izlaze iz objekata i pokazuju vremenski okvir tokom kojeg objekat učestvuje u procesu.
- One predstavljaju trajanje objekta u toku interakcije, gde što je niže na liniji, to se radnja događa kasnije.
- Omogućuju praćenje toka događaja u vremenskoj sekvenci i jasnu ilustraciju kada i kako objekti komuniciraju.

📌 4. Aktivacija (Activation Bar)

- Aktivacija je prikazana kao uski pravougaonik na liniji života objekta. Ona predstavlja period kada objekat izvršava neku akciju ili metod, tj. kada je aktivan u procesu.
- Aktivacija pokazuje da objekat preduzima određene operacije, bilo da se radi o pozivu funkcije, obradi podataka ili čekanju na odgovor.
- Aktivacija je dosta važna za razumevanje tačno kada objekat preuzima odgovornost za izvršenje određenih zadataka i kada je on "aktivan" u procesu.

🔄 **Primer toka – Prijava korisnika**

Prikažimo kako izgleda sekvencijalni dijagram za jednostavan slučaj prijavljivanja u sistem.

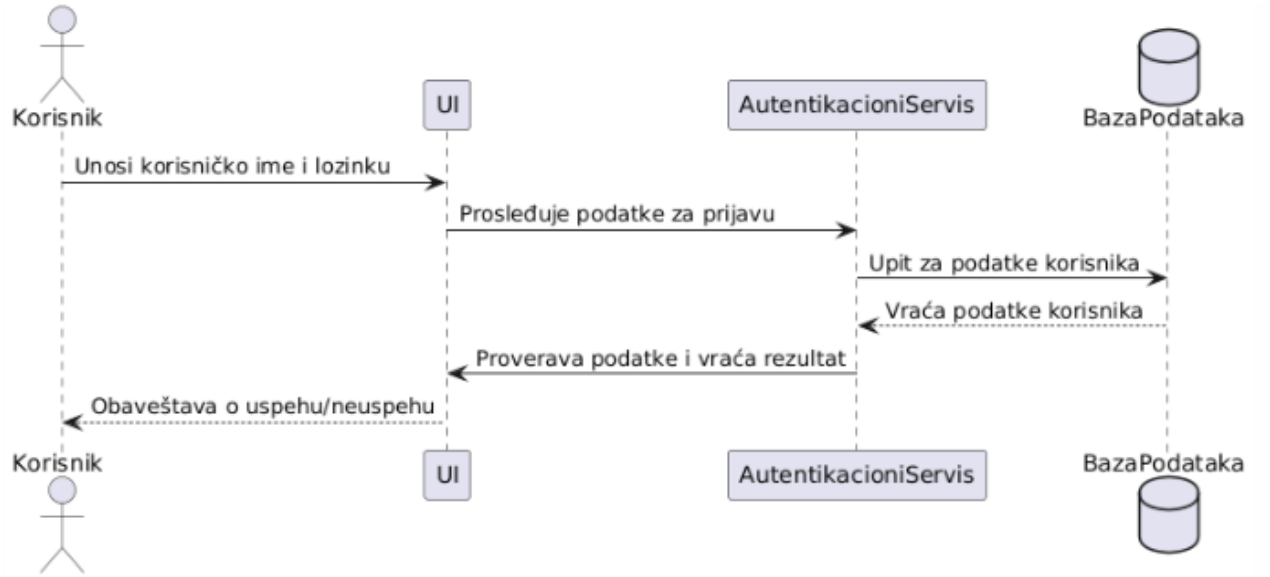
Učesnici:

- Korisnik (akter)
- UI (Interfejs)
- AutentikacioniServis
- BazaPodataka

Sekvenca:

1. Korisnik unosi korisničko ime i lozinku → šalje UI zahtev za prijavu
2. UI prosleđuje podatke autentikacionom servisu
3. Autentikacioni servis šalje upit bazi podataka
4. Baza vraća podatke korisnika
5. Autentikacioni servis proverava podatke i vraća rezultat UI-ju
6. UI obaveštava korisnika o uspehu/neuspehu

🖼️ Svaka strelica pokazuje interakciju; redosled je od vrha ka dnu – kako se radnja odvija.



Tok prijave korisnika prikazuje standardni proces autentifikacije, gde svaki korak zavisi od prethodnog, počevši od unosa podataka do obaveštavanja korisnika o rezultatu. Ovaj dijagram jasno ilustruje kako se korisnički podaci unose, šalju na proveru prema bazi podataka, te kako se na osnovu tih podataka donosi odluka o pristupu sistemu.

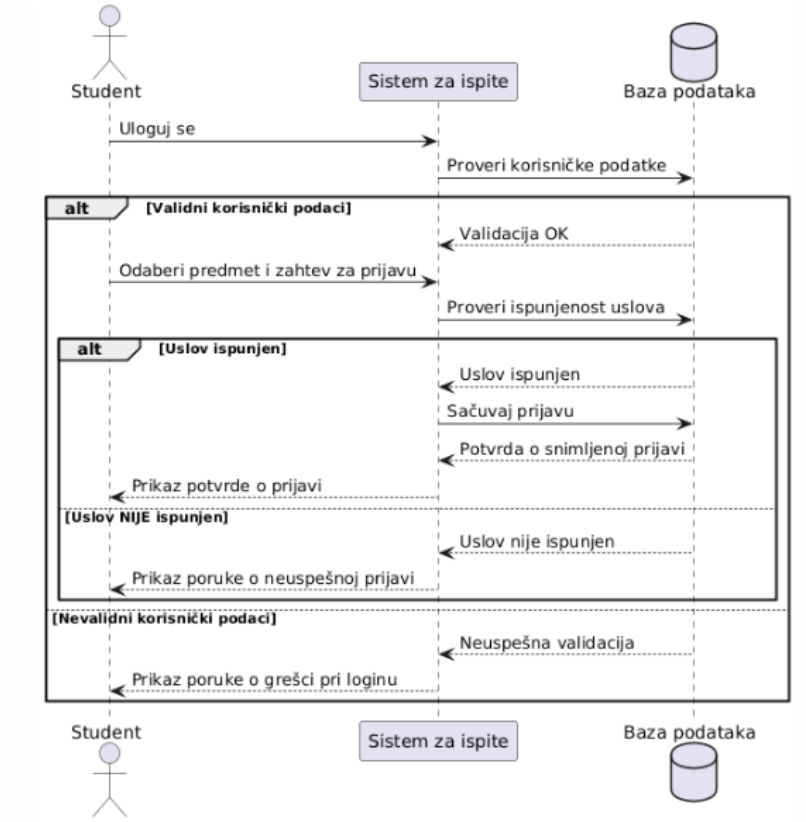
Takođe, prikazuje kako autentikacioni servis vrši verifikaciju podataka i odgovara korisniku putem interfejsa, što omogućava sigurnost i preciznost u procesu autentifikacije.

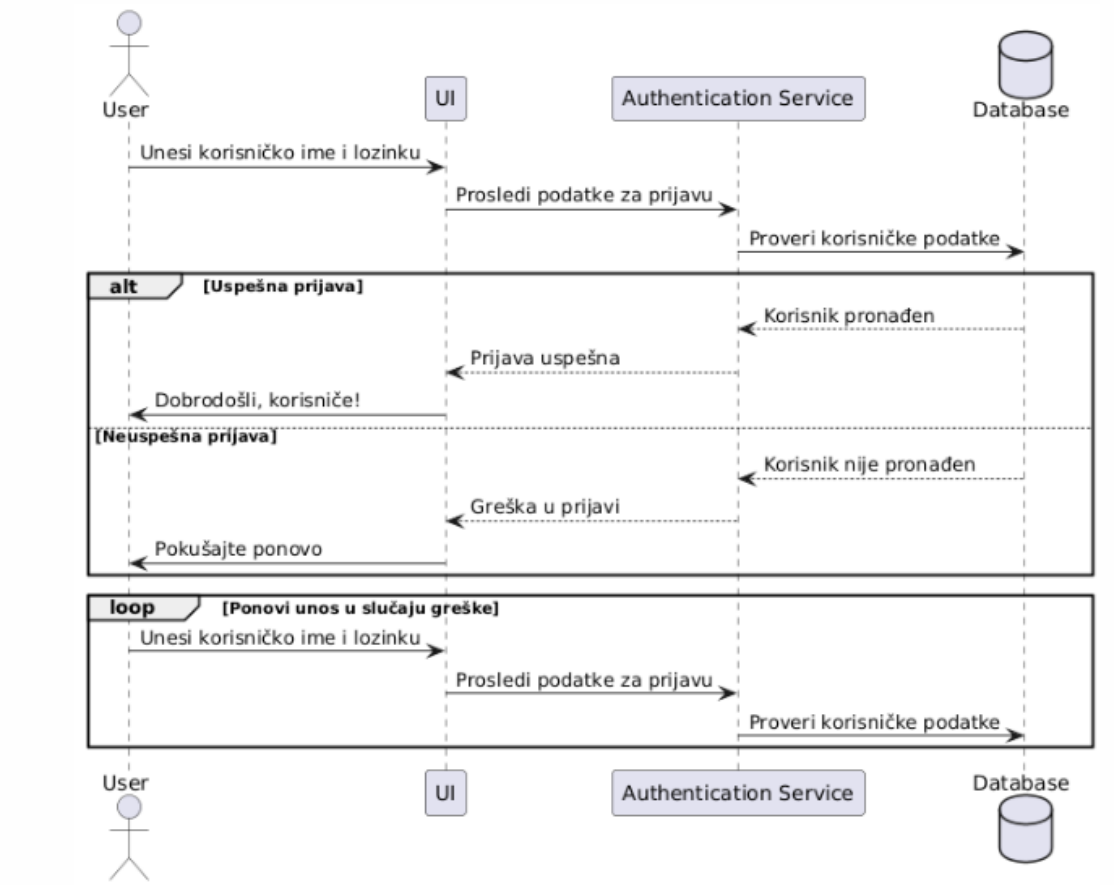
✅ Primeri dobre prakse

1. **Korišćenje jasnih i preciznih naziva poruka**
   - Imena metoda/akcija treba da budu kratka ali opisna (npr. validirajKorisnika(), vratiPodatke())
2. **Ne pretrpavati dijagram nepotrebnim objektima**
   - Prikazivati samo one objekte koji aktivno učestvuju u datoj funkcionalnosti
3. **Hronološki redosled**
   - Poruke treba da budu poređane u redosledu kako se događaju – sekvencijalni dijagram je vremenski orijentisan.
4. **Pravilno korišćenje sinhronih i asinhronih poziva**
   - Ako se očekuje povratna vrednost → koristi punu strelicu (sinhroni poziv)
   - Ako se poziv ne čeka → koristi strelicu bez povratne (asinhrono)

🔶 Petlje i grananja

- UML omogućuje korišćenje **kontrolnih struktura**:
  - loop: ponavljanje (npr. provera svih stavki u korpi)
  - alt: alternativni tok (npr. uspešna vs. neuspešna prijava)
  - opt: opcioni tok (npr. slanje emaila ako korisnik traži)





🔶 Uništavanje objekata

- Kraj linije života može biti označen velikim "X" → znači da objekat više ne postoji (npr. konekcija sa bazom je zatvorena)

🛠️ Praktična primena u razvoju softvera

1. **Specifikacija složenih procesa**
   Sekvencijalni dijagrami pomažu timu da razjasni redosled događaja i komunikaciju između komponenti sistema, čime se smanjuje mogućnost grešaka.
2. **Razrada Use Case-ova**
   Svaki Use Case može imati svoj dijagram koji prikazuje detaljan tok interakcije i logiku sistema, olakšavajući razumevanje funkcionalnosti.
3. **Pregled grešaka i optimizacija**
   Dijagrami otkrivaju nepotrebnu komunikaciju, duplikate i loše definisane korake, što pomaže u optimizaciji i poboljšanju sistema.