UNIVERZITET U BANJOJ LUCI
ELEKTROTEHNIČKI FAKULTET

Prof. dr Dražen Brđanin

PROJEKTOVANJE SOFTVERA
/dekompozicija sistema/

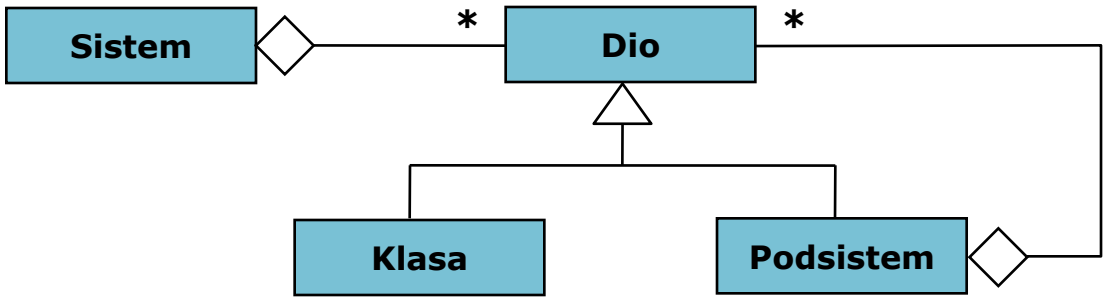
Banja Luka
2024.

2. Dekompozicija sistema

Dekompozicija sistema je postupak kojim se sistem reprezentuje kao kolekcija podsistema.

Podsistemi

- **Podsistem** je kolekcija blisko povezanih klasa, asocijacija, operacija, događaja i ograničenja.
- Podsistemi se u UML modeluju kao paketi.
- Podsistem je **zamjenljivi dio sistema sa dobro definisanim interfejsima**, a koji inkapsulira stanje i ponašanje svih sadržanih klasa.
- Podsistem tipično korespondira obimu posla kojim može biti dodijeljen jednom programeru ili razvojnom timu. Dekompozicijom sistema na (relativno nezavisne) podsisteme, više timova može istovremeno da razvija pojedine podsisteme uz minimalnu međusobnu komunikaciju.
- U slučaju kompleksnih sistema, dekompozicija može rekurzivno da se primjenjuje – podsistemi mogu da se dekomponuju na jednostavnije podsisteme.



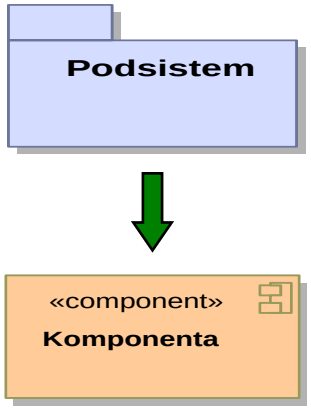
Podsistemi se različito realizuju u programskim jezicima:

Java (paketi)

C++ (ne podržava podsisteme)

2. Dekompozicija sistema

Podsistem = komponenta ciljnog softverskog sistema



- U kasnijim fazama, podsistemi se mapiraju u komponente sistema:
  - **logička** komponenta
    podsistem koji nema eksplicitni run-time ekvivalent
  - **fizička** komponenta
    podsistem koji ima eksplicitni run-time ekvivalent, npr. DB server

Proces dekompozicije

- **Iterativno-inkrementalni proces**:
  - iterativne revizije postojeće dekompozicije i inkrementalno adresiranje novog
  - **dijeljenje/spajanje** i **dodavanje/izbacivanje**
- **Inicijalna dekompozicija**:
  - zasnovana na funkcionalnom modelu
  - **slučaj upotrebe → podsistem** (dijelovi strukturnog modela koji pripadaju istom slučaju upotrebe pripadaju jednom podsistemu)
- **Naknadne transformacije**:
  - tehnike**: raslojavanje** i **particionisanje** (*layering/partitioning*)
  - arhitekturni stilovi (šabloni za dekompoziciju)
    - klijent-server, MVC, repozitorijum, 3-slojna, 4-slojna, SOA, ...
    - ciljna softverska arhitektura je konkretna instanca nekog arhitekturnog stila

2. Dekompozicija sistema (nastavak)

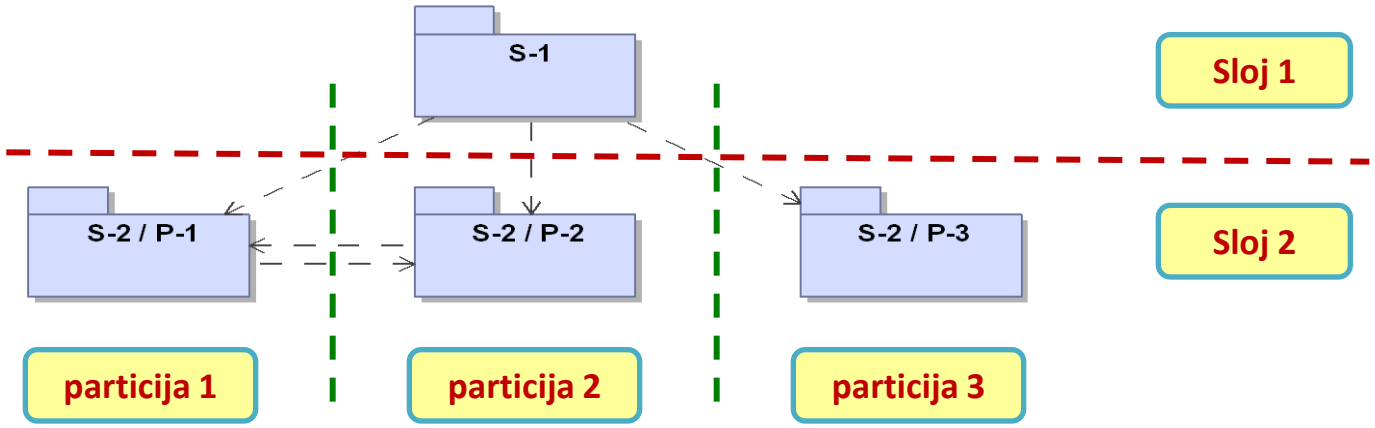
Tehnike za dekompoziciju (tehnike za smanjivanje stepena spregnutosti)

Vertikalna dekompozicija (raslojavanje)

- **Sloj** je podsistem koji obezbjeđuje servis drugom podsistemu, uz sljedeća ograničenja:
  - sloj zavisi samo od servisa nižih slojeva,
  - sloj nema znanje o višim slojevima.

Horizontalna dekompozicija (particionisanje)

- Sloj može horizontalno da se izdijeli na više manjih (nezavisnih) podsistema – **particije**
  - particije obezbjeđuju servise drugim particijama istog sloja
  - particije se često nazivaju **slabo spregnuti podsistemi** (*weakly coupled*)



2. Dekompozicija sistema (nastavak)

Veze između podsistema

Vertikalne veze (veze između slojeva)

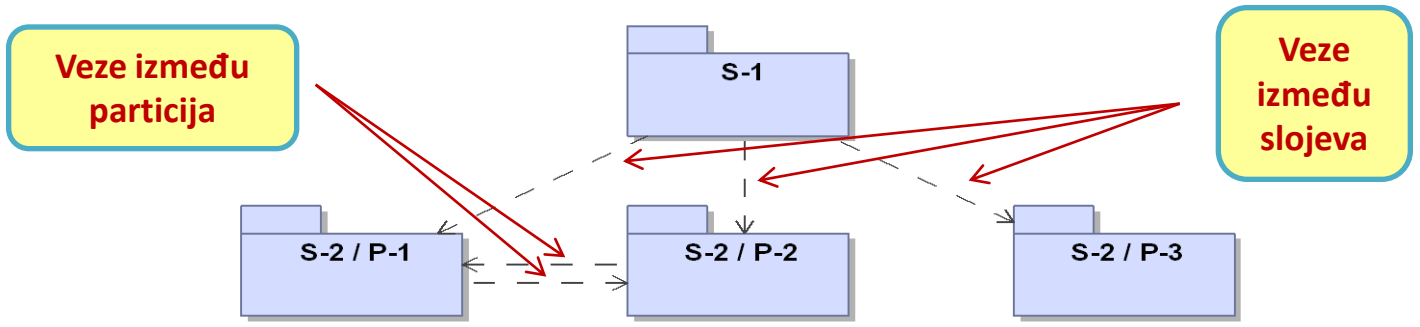
- **"compile time" zavisnosti**
  - sloj A **zavisi od** sloja B
  - npr. *import*, *build*
- **"run time" zavisnosti**
  - sloj A **poziva** sloj B
    (npr. web klijent poziva web server)
  - slojevi nisu procesorski čvorovi
    (SW/HW mapiranje je predmet naknadne analize)

Horizontalne veze (veze između particija)

- **"Peer-to-Peer" veze**
  - particije su ravnopravne i imaju uzajamno znanje jedna od drugoj
  - particija A poziva particiju B i particija B poziva particiju A

Neke heuristike za dekompoziciju

- Ukupan **broj podsistema**: **≤10**
  - više podsistema povećava stepen sprege u sistemu (više servisa)
- Ukupan **broj slojeva**: **≤5**
  - **dobar dizajn: 3 sloja**



2. Dekompozicija sistema (nastavak)

Osnovni cilj dekompozicije: REDUKCIJA SLOŽENOSTI

Mjere složenosti

Povezanost (*coherence*)

- mjera zavisnosti između klasa u podsistemu
- **visoka povezanost** (*high coherence*)
  - klase u podsistemu imaju slične uloge, izvršavaju slične zadatke, i uzajamno su povezane većim brojem asocijacija
  - postiže se ako je većina interakcija unutar podsistema, a ne preko granica podsistema
  - **Uvijek pitanje:**

    Da li dati podsistem stalno poziva neki servis drugog podsistema?

    Da: Objediniti podsisteme ako je moguće!
- **niska povezanost** (*low coherence*)
  - velik broj pomoćnih i uslužnih klasa u podsistemu, koje su povezane malim brojem asocijacija

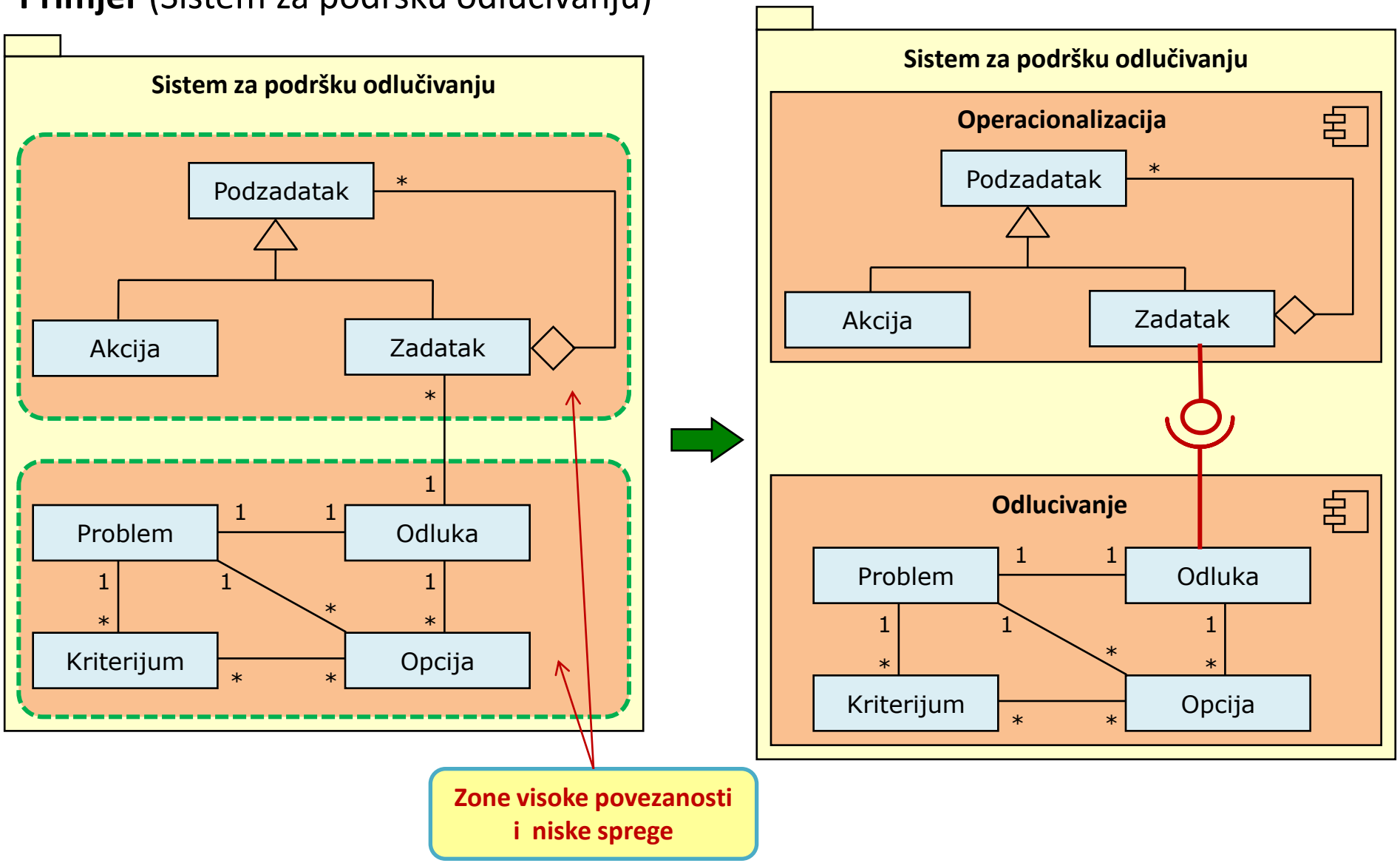
Sprega (*coupling*)

- mjera zavisnosti između podsistema
- **visoka sprega** (*high coupling*)
  - promjena u jednom podsistemu ima velik uticaj na drugi podsistem
- **niska sprega** (*low coupling*)
  - promjena u jednom podsistemu nema uticaj na drugi podsistem
  - postiže se ako pozivajuća klasa ne mora da zna ništa o implementaciji pozivane klase (princip skrivanja informacija)

Dobar dizajn

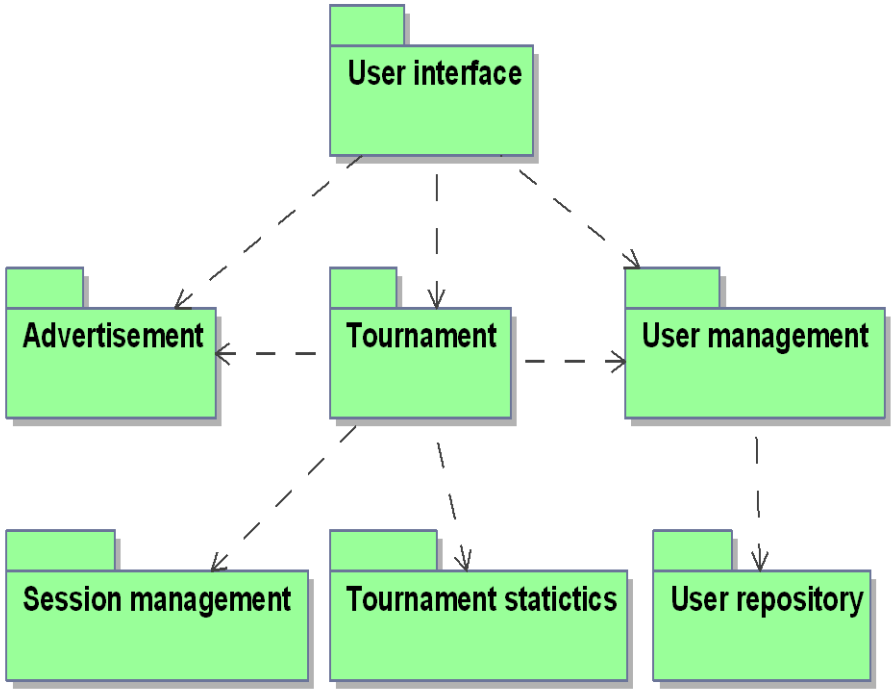
2. Dekompozicija sistema (nastavak)

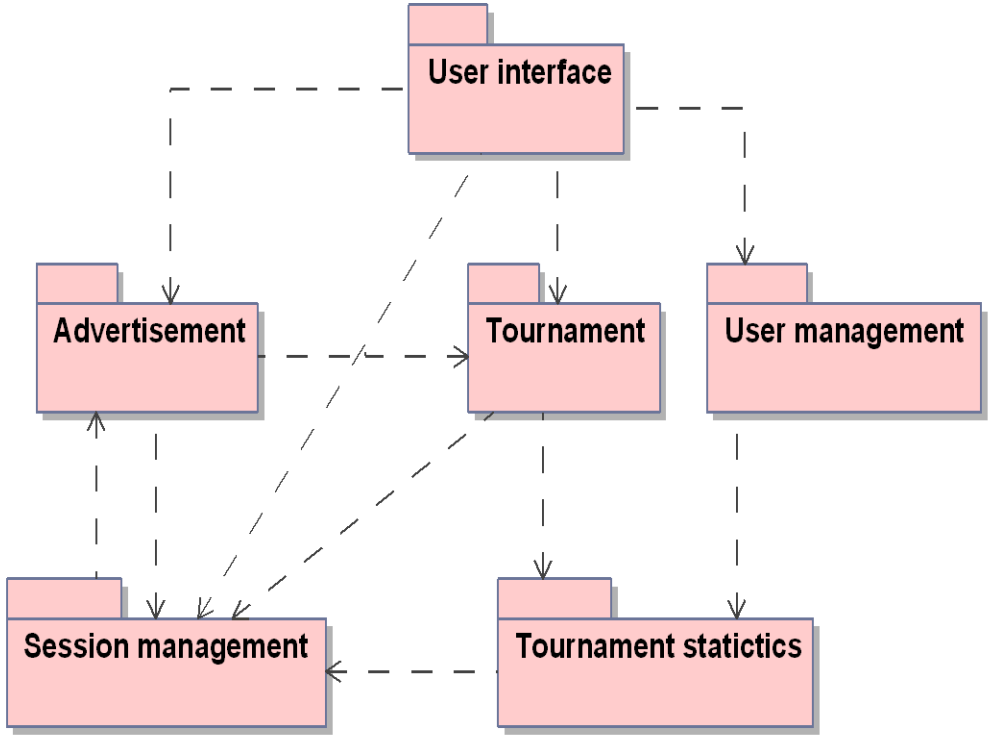
Primjer (Sistem za podršku odlučivanju)



2. Dekompozicija sistema (nastavak)

Primjeri dekompozicije





✓

- 3-slojna dekompozicija
- slojevi nizu uzajamno zavisni
- niska sprega podsistema

✘

- visoka sprega podsistema
- slojevi uzajamno zavisni
- ...

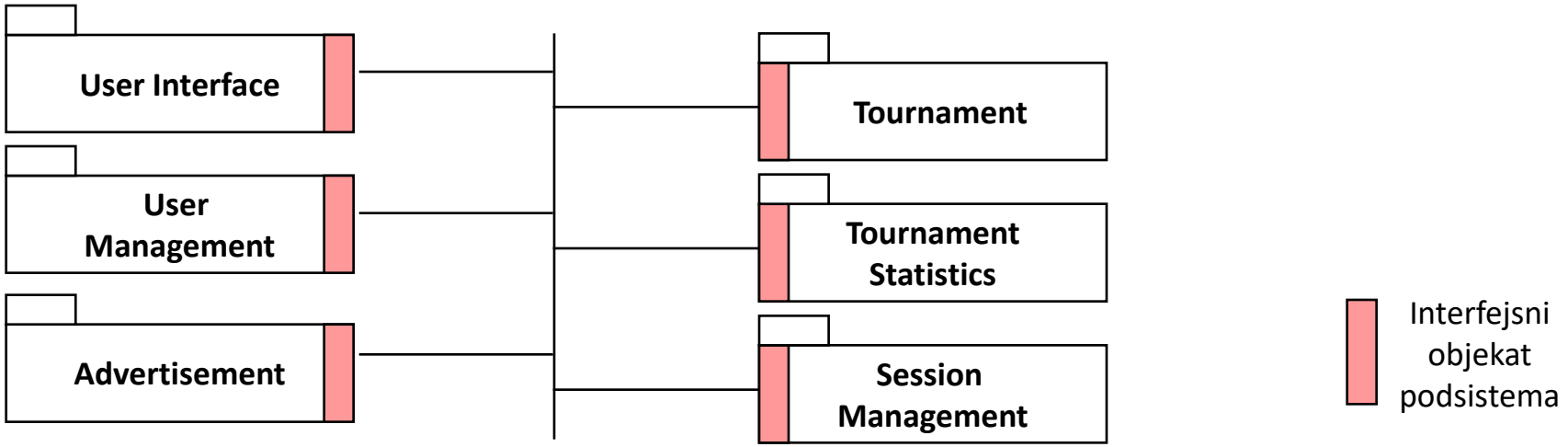
2. Dekompozicija sistema (nastavak)

Servisi

- **Servis** je kolekcija operacija kojom raspolaže neki podsistem.
- Servisi se definišu tokom projektovanja sistema, a rafiniraju kao interfejsi podsistema tokom detaljnog objektnog dizajna.
- Heuristika: slučajevi upotrebe često figurišu kao servisi podsistema

Interfejs podsistema

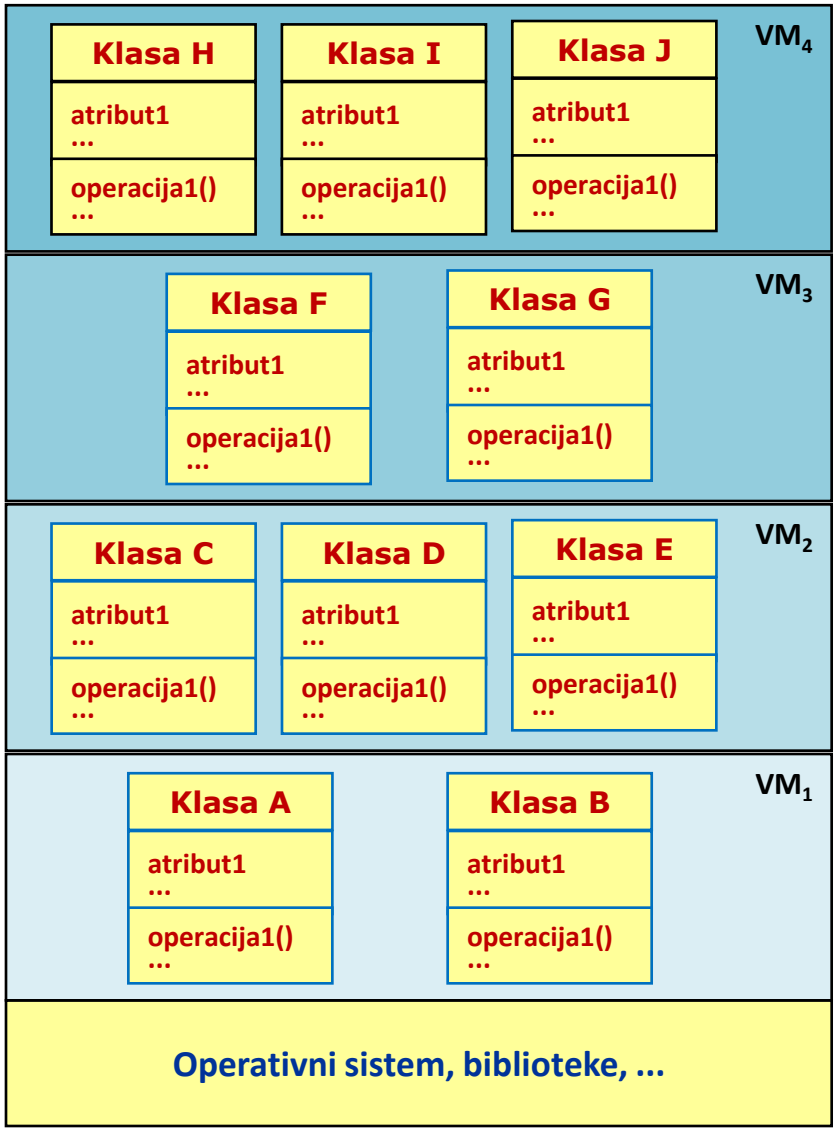
- **Interfejs podsistema** je skup operacija kojima se specifikuje interakcija i tok informacija između podsistema (ne unutar podsistema).
- **Dobar dizajn**: **svaki podsistem ima jedan interfejsni objekat koji reprezentuje sve servise kojima raspolaže dati podsistem (projektni obrazac FASADA)**
- **Dobar dizajn: sistem kao skup interfejsnih objekata**



2. Dekompozicija sistema (nastavak)

Virtuelna mašina ("nivo apstrakcije")

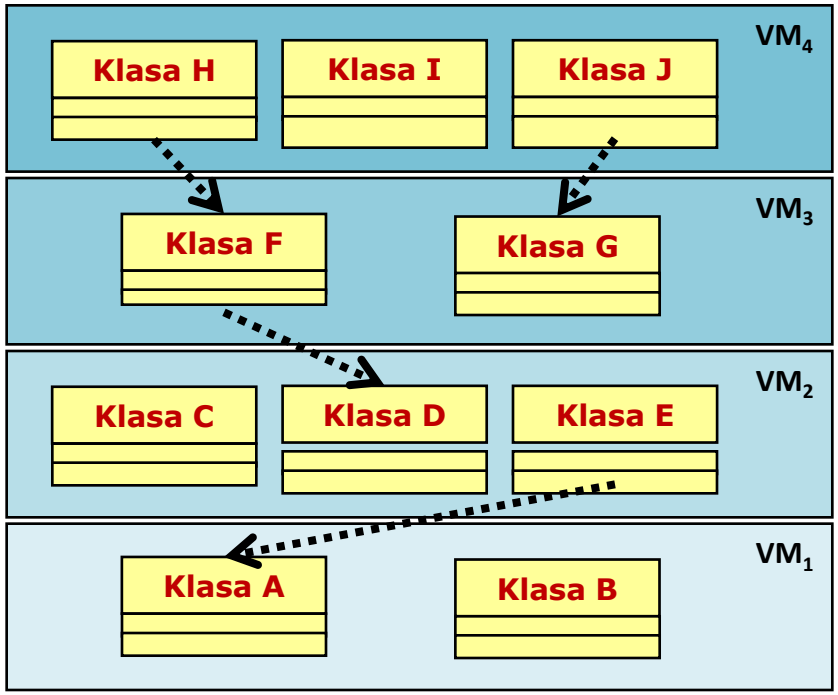
- **Virtuelna mašina** je podsistem koji je sa drugim virtuelnim mašinama (iznad i ispod) povezan vezama tipa "**obezbjeđuje servis za**".
- Slojevi i virtuelne mašine često se poistovjećuju.
- Virtuelna mašina je kolekcija klasa – **modul koji koristi servise virtuelnih mašina ispod i obezbjeđuje servise virtuelnim mašinama iznad.**
- Virtuelna mašina je ključni koncept u apstrakciji sistema ("out of date" za distribuirane sisteme, ali veoma pogodan za jednoprocesorske arhitekture)
- **Sistem kao hijerarhija virtuelnih mašina**, svaka virtuelna mašina koristi jezičke primitive koje obezbjeđuju niže virtuelne mašine



2. Dekompozicija sistema (nastavak)

Zatvorena arhitektura (*opaque layering*)

– Svaka VM može da poziva samo operacije iz VM neposredno ispod.

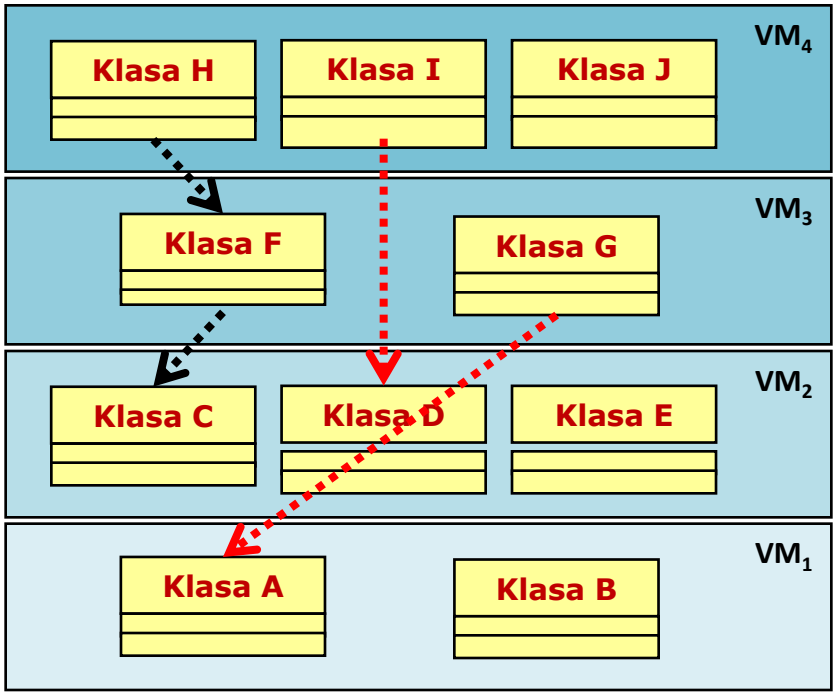


Projektni ciljevi:

– fleksibilnost
– lako održavanje
– prenosivost

Otvorena arhitektura (*transparent layering*)

– Svaka VM može da poziva operacije iz bilo koje VM ispod.



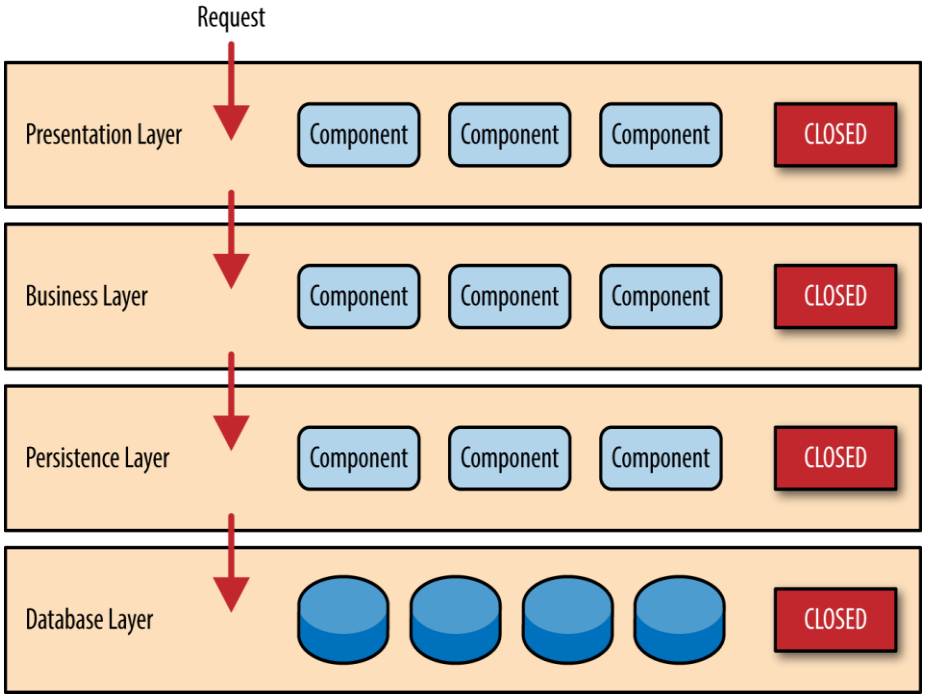
Projektni ciljevi:

– *run-time* efikasnost

2. Dekompozicija sistema (nastavak)

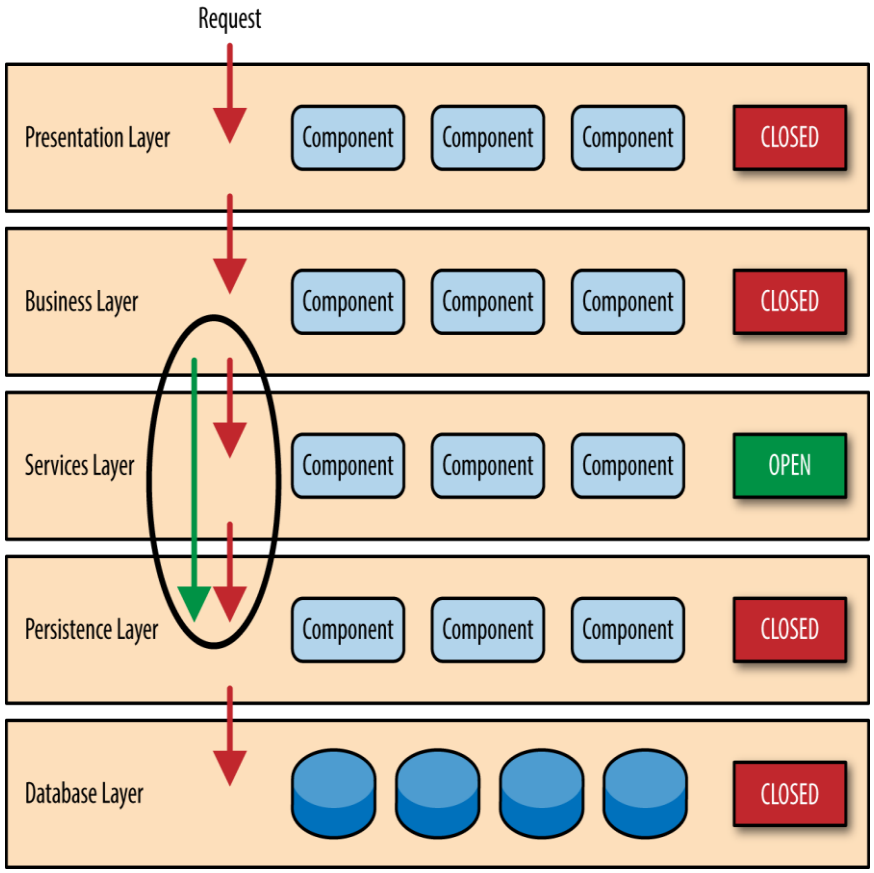
Zatvorena arhitektura (*opaque layering*)

– Svaka VM može da poziva samo operacije iz VM neposredno ispod.



Otvorena arhitektura (*transparent layering*)

– Svaka VM može da poziva operacije iz bilo koje VM ispod.



2. Dekompozicija sistema (nastavak)

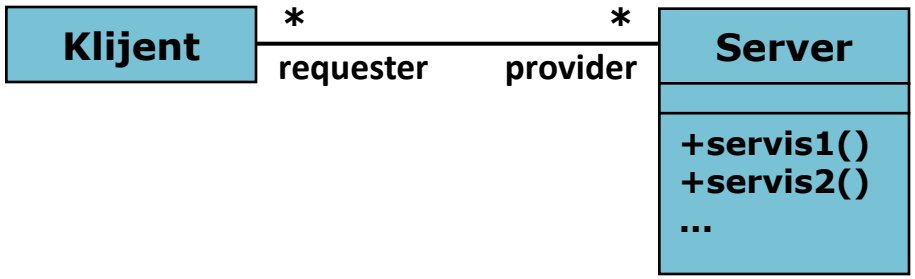
Tipični arhitekturni stilovi (šabloni za dekompoziciju)

Klijent/Server arhitekturni stil

- Jedan ili više **servera** obezbjeđuje servise **klijentima**
- **Klijent poziva server**, **server izvršava odgovarajući servis** i **vraća rezultat**
  - **Klijenti znaju interfejs servera** (bez toga nema ni poziva servisa)
  - **Server ne mora da zna interfejs klijenta**
- U opštem slučaju, odziv je neposredan
- Krajnji korisnici imaju interakciju samo sa klijentom

Klijent/Server arhitekture

- Česta upotreba u sistemima sa bazama podataka
  - Front-end: korisnička aplikacija (klijent)
  - Back-end: manipulacija bazom (server)
- Funkcije klijenta:
  - unos podataka (prilagođeni UI interfejs)
  - *front-end* obrada podataka
- Funkcije servera:
  - centralizovano upravljanje podacima
  - integritet i konzistentnost podataka
  - sigurnost
  - ...



2. Dekompozicija sistema (nastavak)

Tipični arhitekturni stilovi

Projektni ciljevi u klijent/server arhitekturi

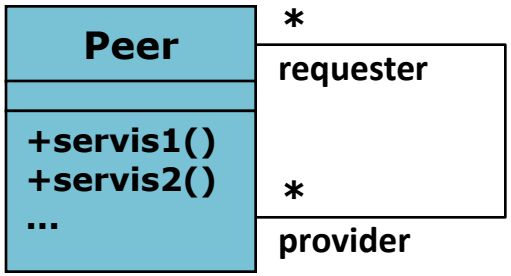
Prenosivost (portabilnost) –	Server ima mogućnost izvršavanja na različitim operativnim sistemima i u različitim mrežnim okruženjima
Slojevitost i transparentnost –	Server može da ima distribuiranu arhitekturu, ali ga klijenti vide kao jedinstven “logički” servis
Visoke performanse –	Klijenti optimizovani za intenzivnu interakciju sa korisnikom Server optimizovan za CPU i *storage* procesiranje
Skalabilnost –	Server ima mogućnost posluživanja velikog broja klijenata
Fleksibilnost –	Mogućnost implementacije klijenta na različitim terminalnim uređajima (desktop, laptop, PDA, mobile, ...)
Pouzdanost –	Server ima ugrađene mehanizme koji obezbjeđuju konzistentnost, integritet, transakcije, ...

2. Dekompozicija sistema (nastavak)

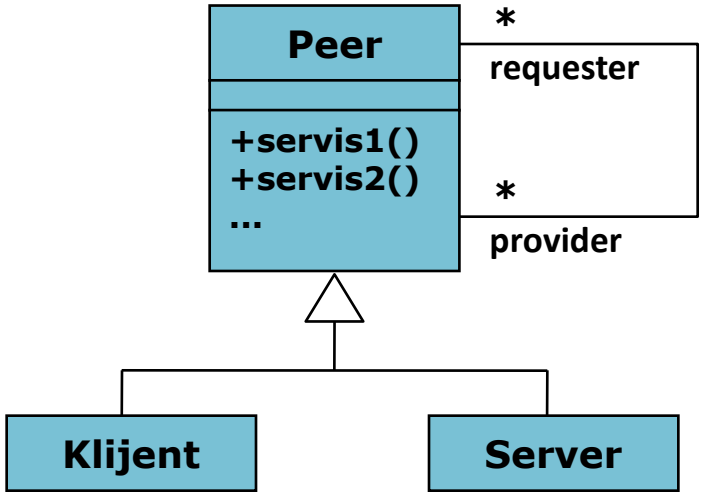
Tipični arhitekturni stilovi

Peer-to-Peer arhitekturni stil

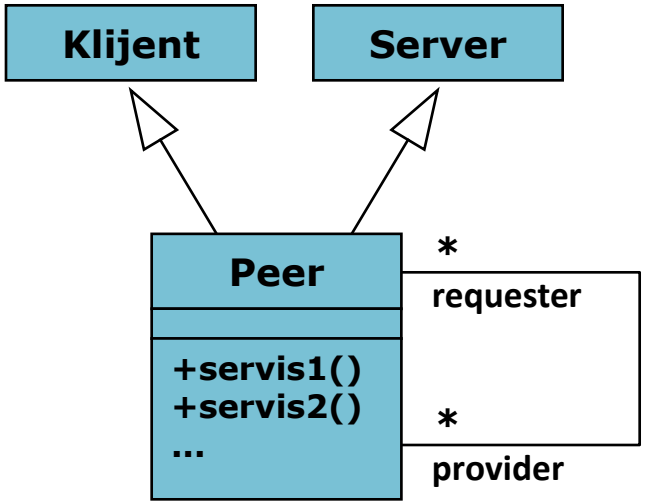
- ***peer*** = vršnjak, parnjak
- generalizacija klijent/server arhitekturnog stila
- **Klijenti mogu da budu serveri, a serveri mogu da budu klijenti**



Projektne peer-to-peer alternative?



✘ **Peer može da bude klijent ili server**



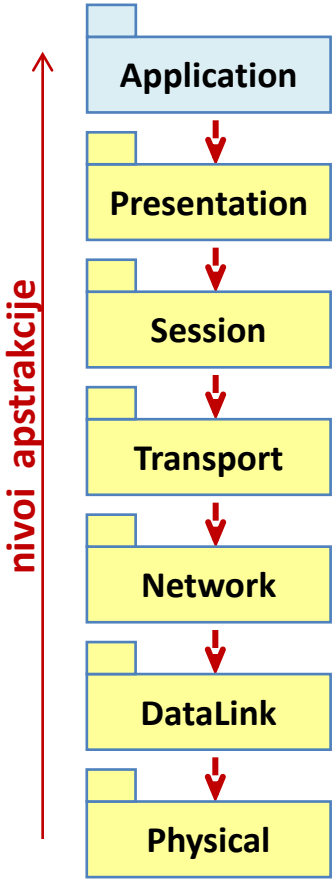
✓ **Peer može da bude i klijent i server**

2. Dekompozicija sistema (nastavak)

Tipični arhitekturni stilovi

Primjer peer-to-peer arhitekture

- **OSI** (*Open System Interconnection*)
  - 7-slojni referentni komunikacioni model



Sloj na kojem projektujemo softver (osim ako nije riječ o steku protokola)

Konverzije, enkripcija/dekripcija, kompresija

Uspostavljanje, održavanje i terminiranje sesije i autorizacija

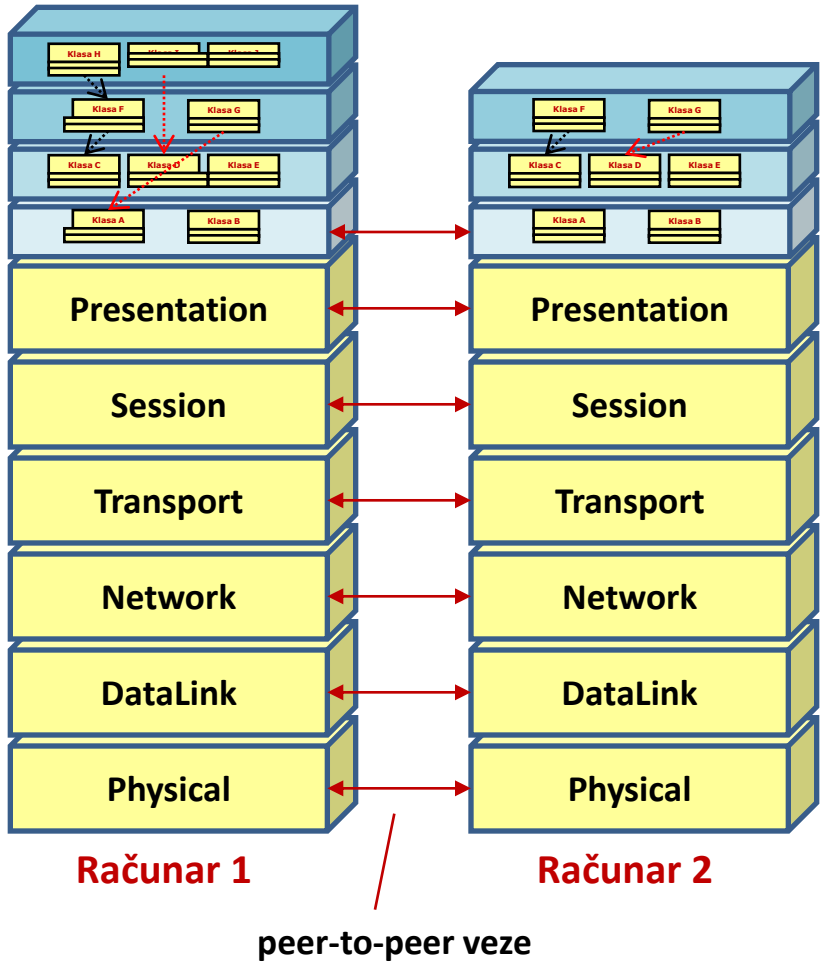
Pouzdan paketski prenos

Rutiranje i mrežni prenos podataka

Formiranje i prenos *frame*-ova bez grešaka

Reprezentuje hardverski interfejs prema mreži

OSI je primjer zatvorene arhitekture

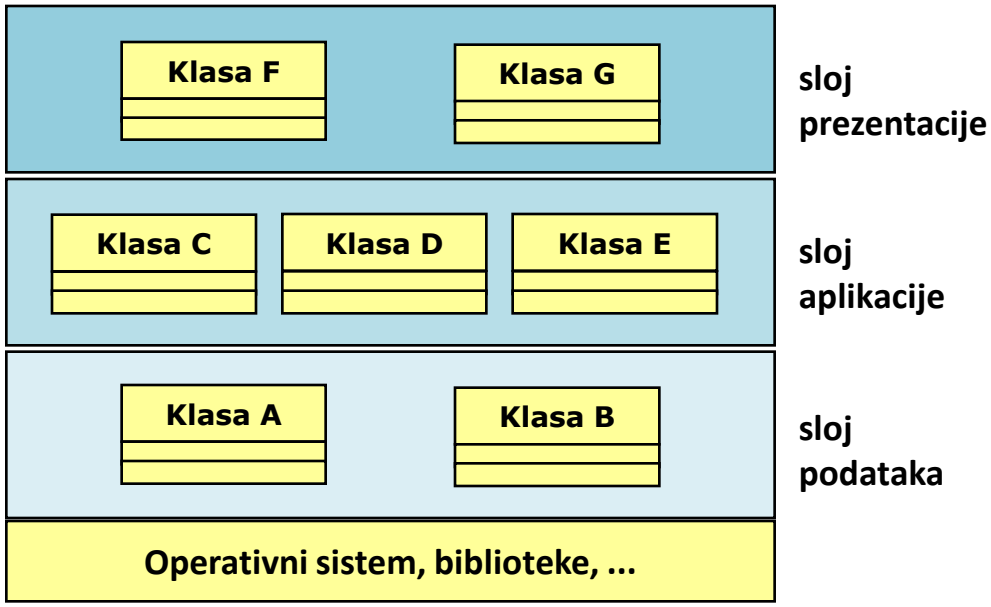


2. Dekompozicija sistema (nastavak)

Tipični arhitekturni stilovi

3-slojni (3-Layer) arhitekturni stil

- Arhitekturni stil u kojem je sistem strukturisan na tri hijerarhijska podsistema (sloja)
- Arhitekturni stil kojeg čine tri virtuelne mašine (Dijkstra, 1965):
  - **sloj prezentacije**: korisnički interfejs / klijent
  - **sloj aplikacije**: *middleware* / *business logic*
  - **sloj podataka**: baza podataka



3-slojna (3-Tier) arhitektura

- Softverska arhitektura koja podrazumijeva distribuciju tri sloja na tri odvojena hardverska čvora

Termini Layer i Tier često se koriste bez razlike

Layer = tip (npr. klasa, podsistem)

Tier = instanca (npr. objekat, čvor)

3-slojni arhitekturni stil tipično se koristi u web programiranju:

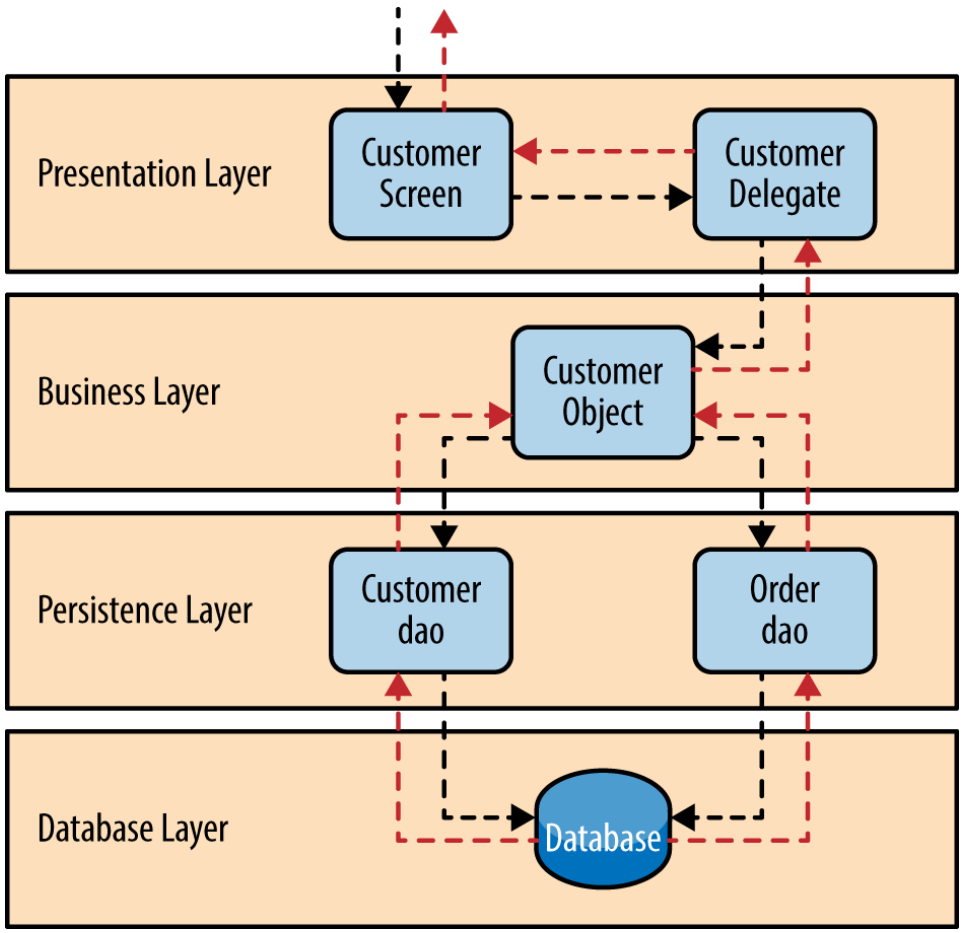
1. Web Browser implementira korisnički interfejs
2. Web Server servisira zahtjeve web browsera
3. DBMS upravlja podacima

2. Dekompozicija sistema (nastavak)

Tipični arhitekturni stilovi

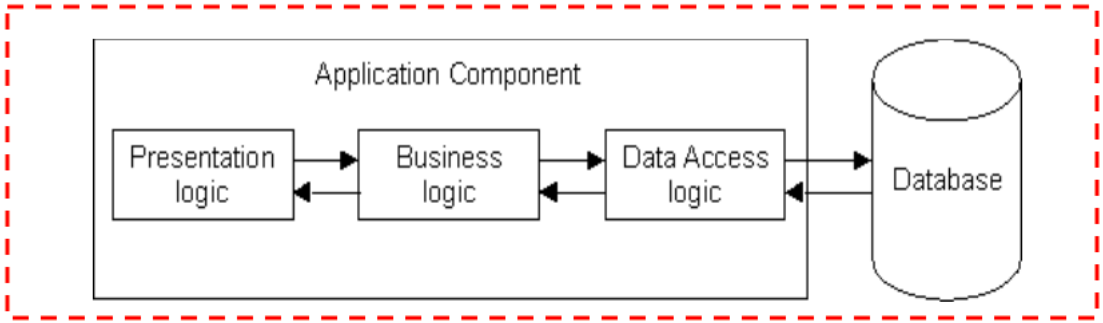
4-slojni (4-Layer) arhitekturni stil

- Arhitekturni stil u kojem je sistem strukturisan na četiri hijerarhijska podsistema (sloja)
- Npr:
  - **Prezentacioni sloj**
    korisnički interfejs
    (JSF + manged beans)
  - **Sloj aplikativne logike**:
    kontroleri za poslovnu logiku
    (local Spring bean / remote EJB3 bean)
  - **Sloj pristupa perzistentnim objektima**
    upravljanje perzistentnim objektima
    (ORM mapping)
  - **Sloj baze podataka**
    DBMS + baza podataka



2. Dekompozicija sistema (nastavak)

Realizacija slojeva – različite arhitekture i različiti fizički razmještaji



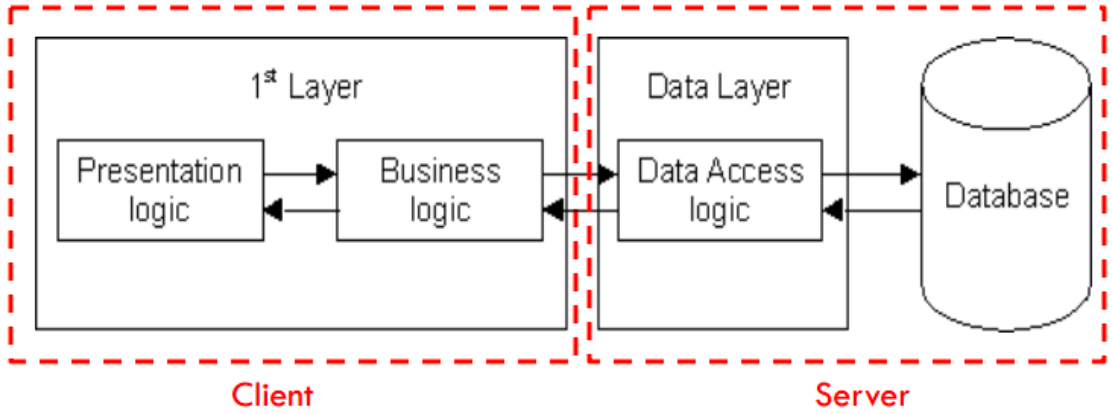
1-slojna (1-Tier) arhitektura

Svi slojevi na istom čvoru i jako spregnuti

Otežana skalabilnost – sve na jednom procesoru

Otežana portabilnost – nova implementacija?

Otežano održavanje – slojevi su integrisani

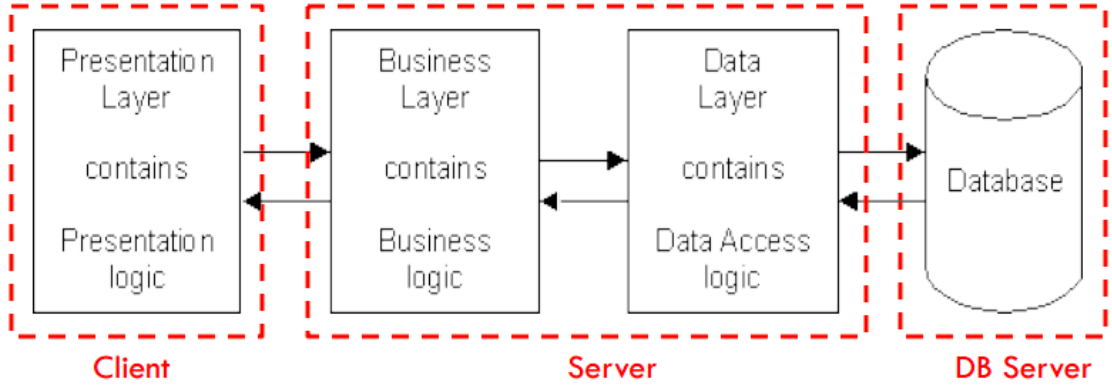


2-slojna (2-Tier) arhitektura

Baza podataka i upravljanje podacima na serveru

Olakšan prelazak na drugi DBMS/bazu podataka

Prezentacioni i aplikativni sloj jako spregnuti



3-slojna (3-Tier) arhitektura

Svi slojevi su slabo spregnuti

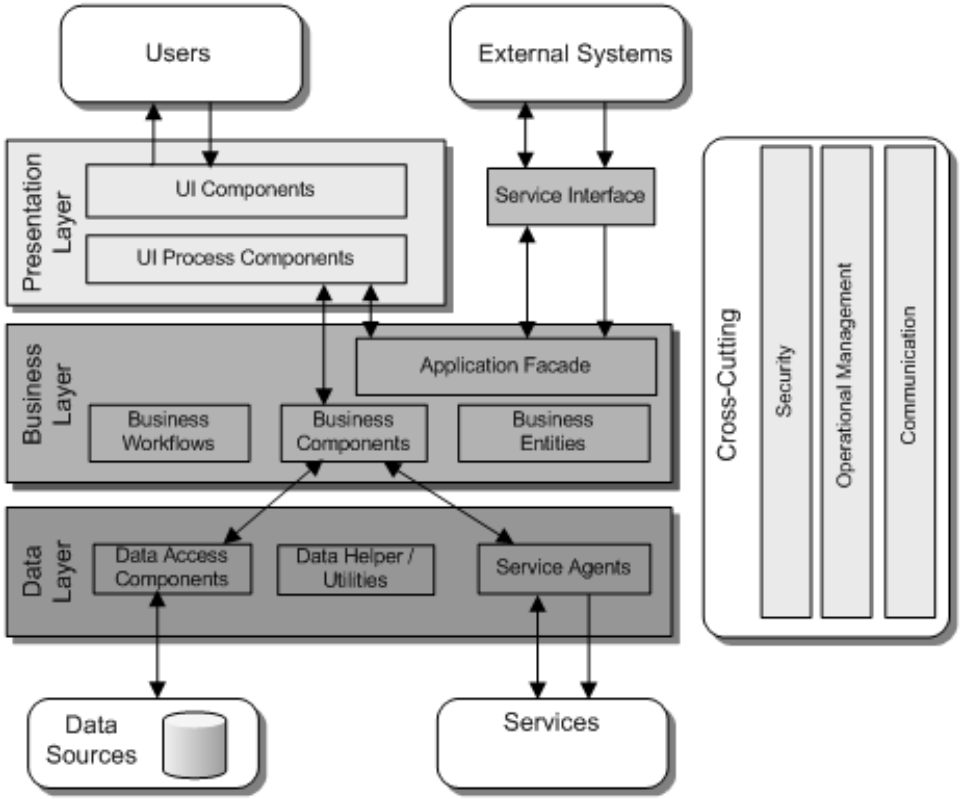
Svaki sloj može biti na posebnom čvoru

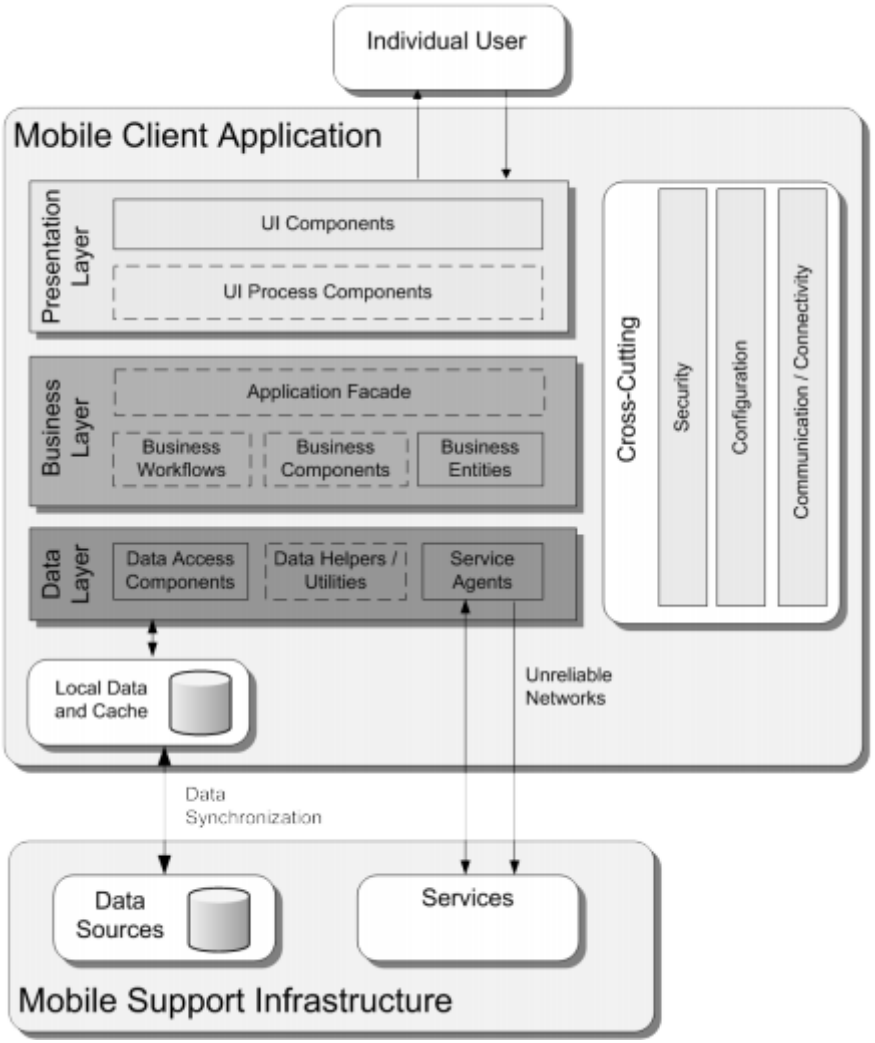
Olakšan razvoj, testiranje, održavanje, reuse

2. Dekompozicija sistema (nastavak)

Tipični arhitekturni stilovi

Primjeri višeslojnih arhitekturnih stilova





2. Dekompozicija sistema (nastavak)

Prednosti višeslojnog stila (visoka kohezija – niska sprega)

Jednostavnost razvoja – Veoma popularan arhitekturni stil u industriji
Razvoj po slojevima – specijalizacija znanja (front-end, back-end, perzistentni sloj)

Jednostavnost testiranja – Svaka komponenta pripada specifičnom sloju pa je komponente u drugim slojevima lako zamijeniti korespondentnim stabovima (zamjenskim testnim komponentama) i slojeve testirati nezavisno

Jednostavnost održavanja – Slojevi su nezavisni i slabo spregnuti pa je lakše raditi izmjene na pojedinim komponentama i nezavisno ih mijenjati alternativnim

Višestruka upotreba (reuse) – Niži slojevi su nezavisni od viših i mogu biti višestruko korišteni

Mane višeslojnog stila

Zahtjevniji razmještaj – Veći broj komponenata na različitim čvorovima

Performanse – Veći broj slojeva smanjuje efikasnost

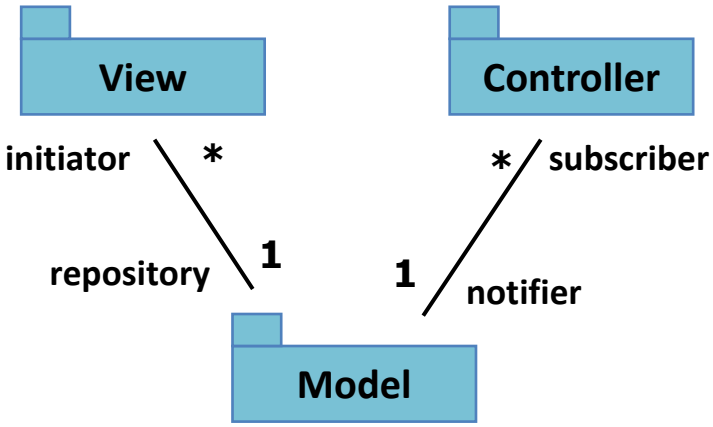
Skalabilnost – Svaki sloj za sebe je monolitan i nije ga lako skalirati
Horizontalno – sloj treba višestruko replicirati na više čvorova
Vertikalno – slojevi na različitim fizičkim čvorovima

2. Dekompozicija sistema (nastavak)

Tipični arhitekturni stilovi

MVC (Model-View-Controller) arhitekturni stil

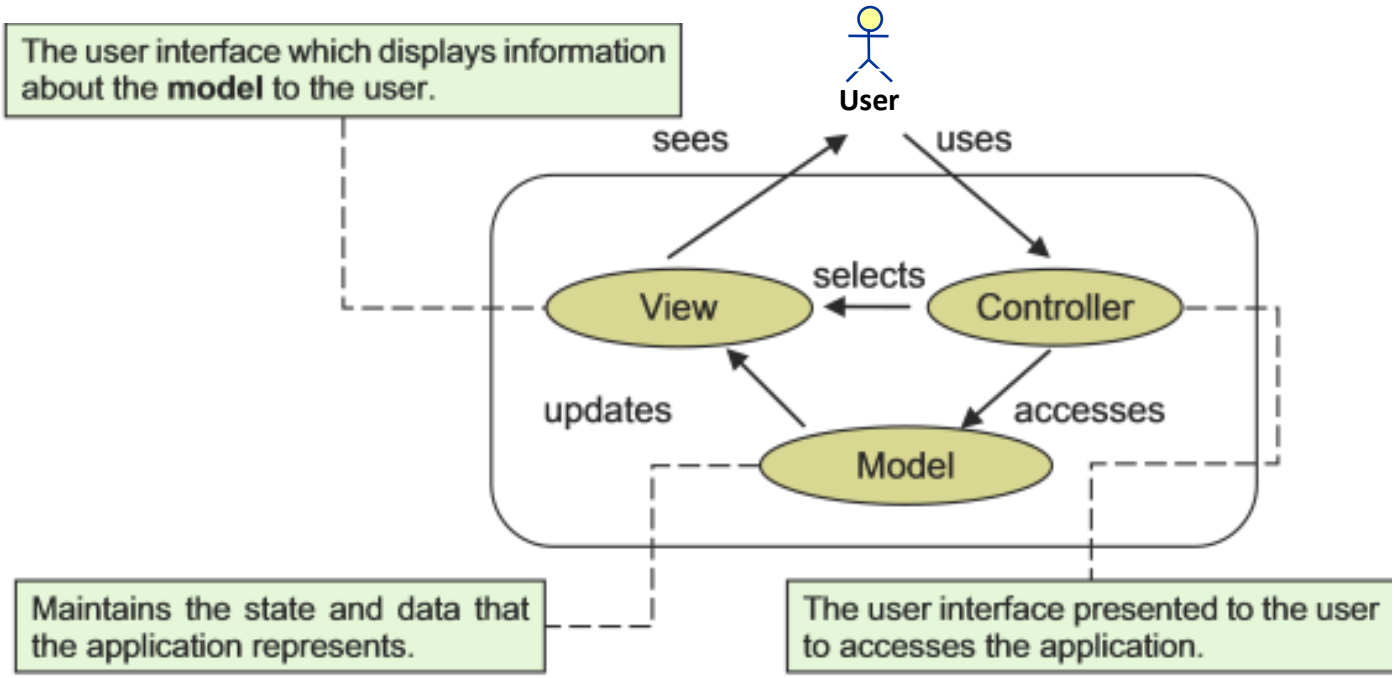
- MVC arhitekturni stil klasifikuje podsisteme u tri kategorije:
  - **model : reprezentacija i pristup podacima (domenskim objektima)**
  - **view: prezentacija podataka korisniku**
  - **controller: aplikativna logika / odgovoran za sekvencu interakcija sistema i korisnika i prosljeđivanje promjena modela prema pogledu**
- Motivacija za MVC:
  - Interfejs sistema mijenja se mnogo češće nego aplikativna logika
  - Aplikativna logika mijenja se mnogo češće nego domenski objekti



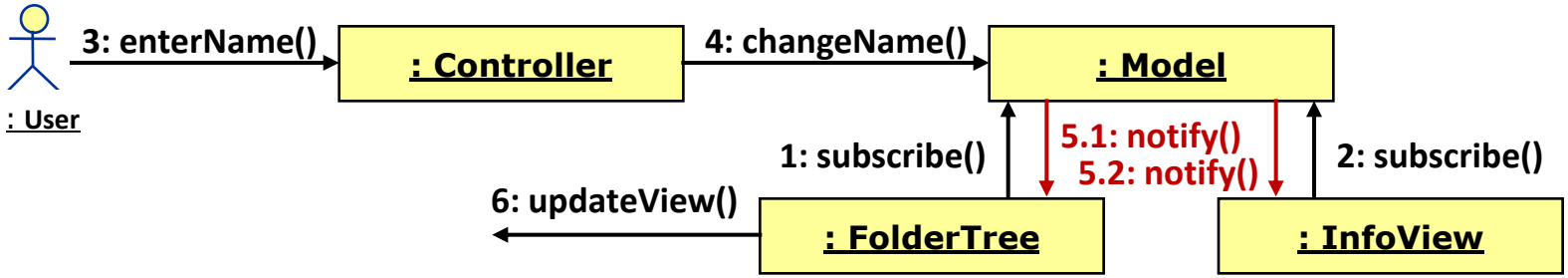
2. Dekompozicija sistema (nastavak)

Tipični arhitekturni stilovi

MVC – interakcija objekata



Primjer interakcije:



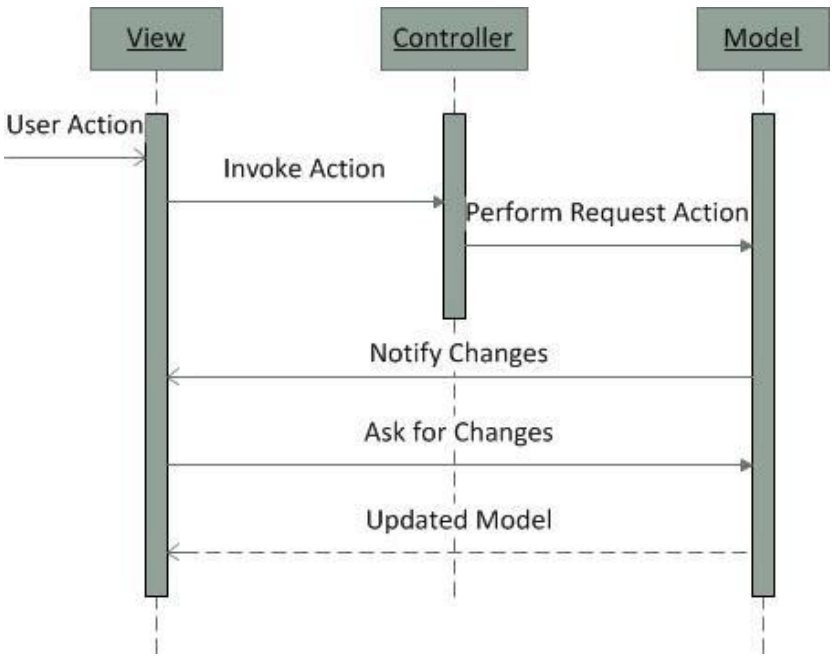
2. Dekompozicija sistema (nastavak)

Tipični arhitekturni stilovi

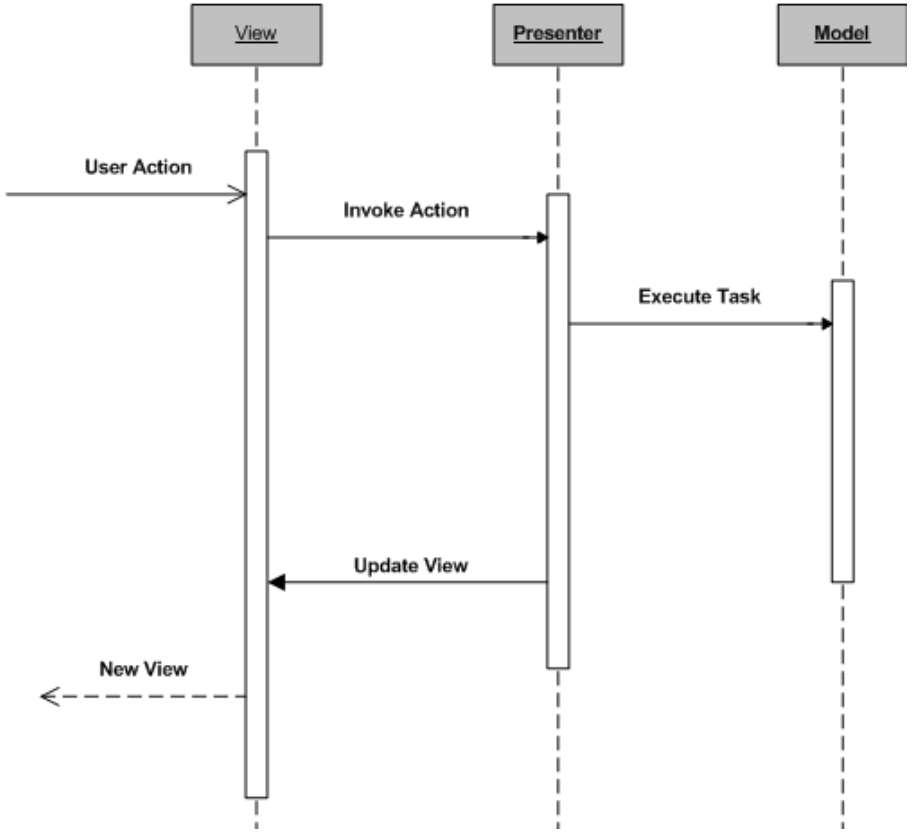
MVC varijante – sprega modela i pogleda nije direktna, nego ide preko kontrolera

- Model-View-Adapter (MVA) ili Mediated MVC ili Model-Mediator-View (MMV)
- Model-View-Presenter (MVP)
- Model-View-ViewModel (MVVM)

MVC

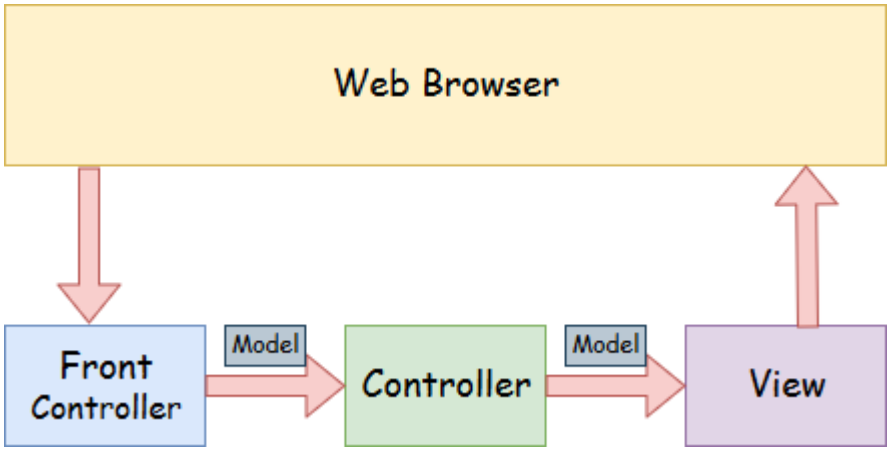


MVP-MVVM



2. Dekompozicija sistema (nastavak)

MVC – primjene (Spring)

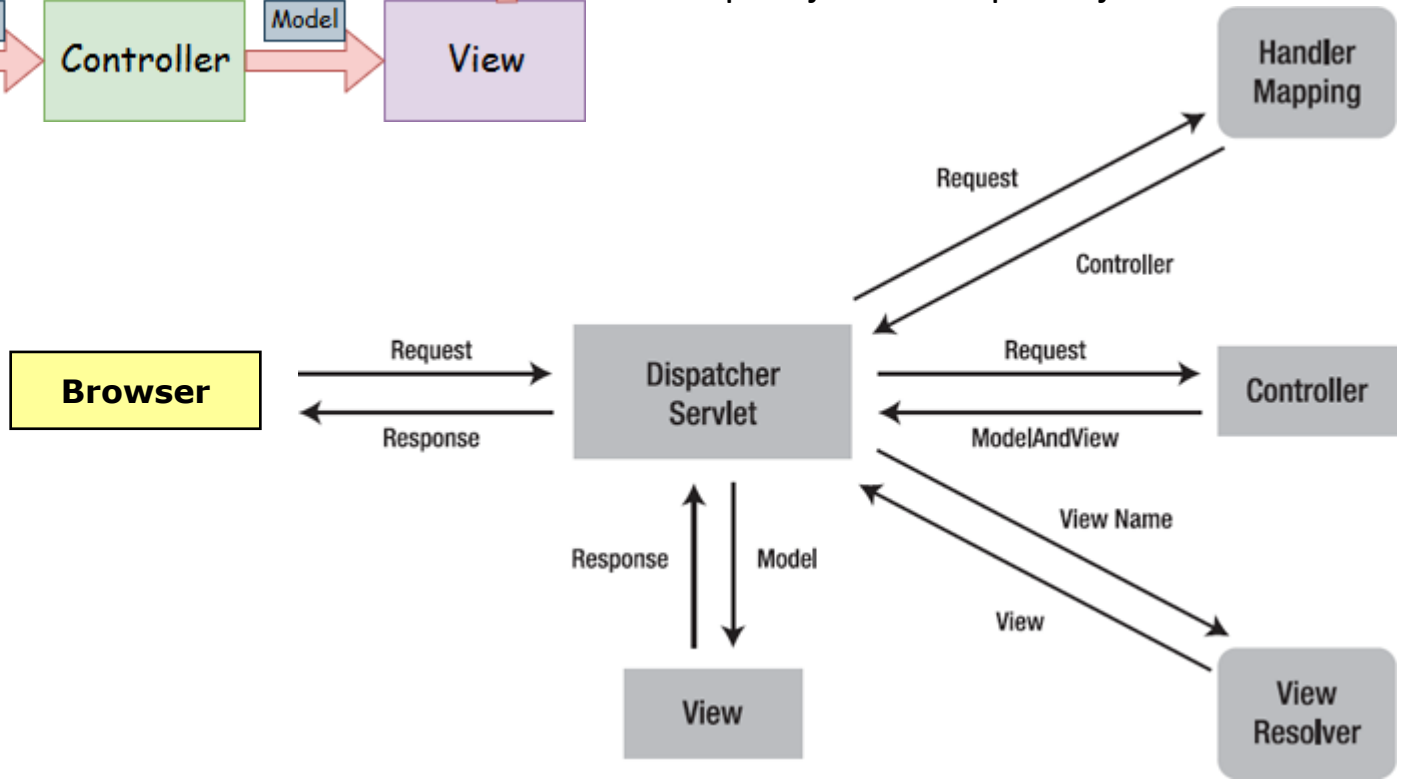


Model – sadrži sve podatke (pojedinačni objekti ili kolekcije)

Controller – implementira poslovnu logiku

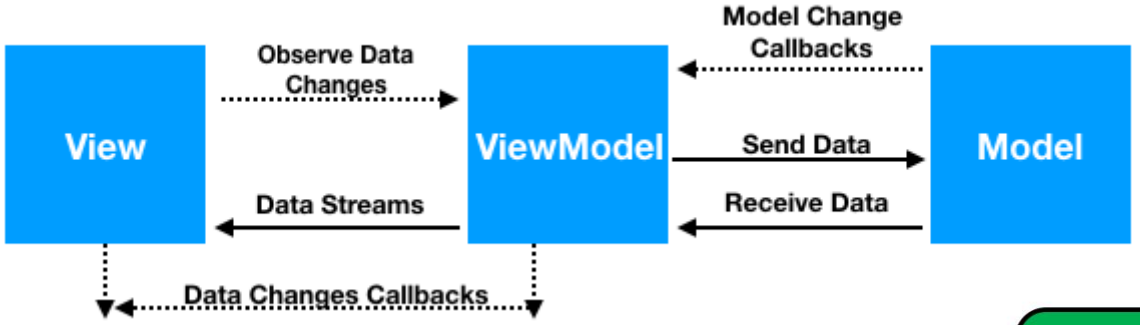
View – reprezentuje korisnički interfejs

Front Controller - DispatcherServlet klasa upravlja tokom aplikacije



2. Dekompozicija sistema (nastavak)

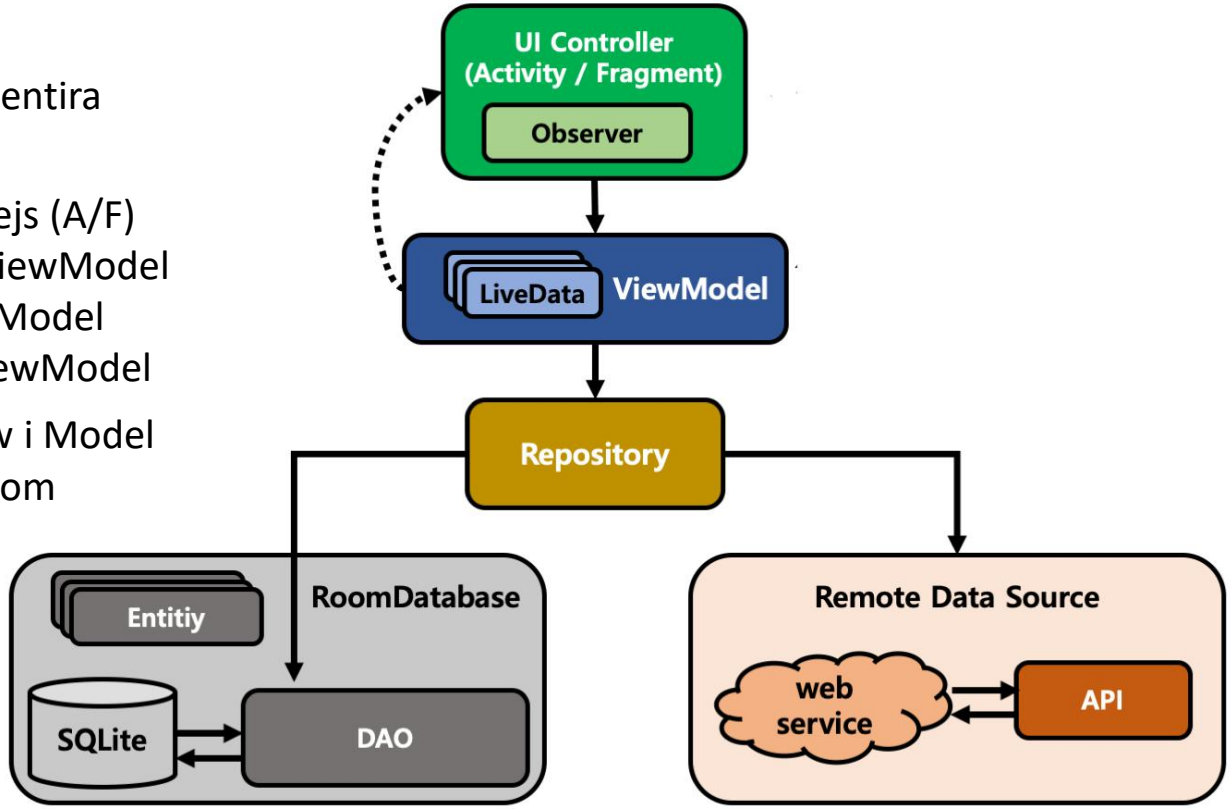
MVVM – primjene (Andriod app)



Model – sadrži sve podatke i implementira poslovnu logiku Android aplikacije

View – reprezentuje korisnički interfejs (A/F) prosljedjuje akcije korisnika prema ViewModel
Odgovor ne dobija direktno od ViewModel
Ima opserver koji se prijavljuje na ViewModel

ViewModel – medjusloj između View i Model ima sinhronu komunikaciju sa Modelom



2. Dekompozicija sistema (nastavak)

Tipični arhitekturni stilovi

MVC arhitekturni stil – prednosti i nedostaci

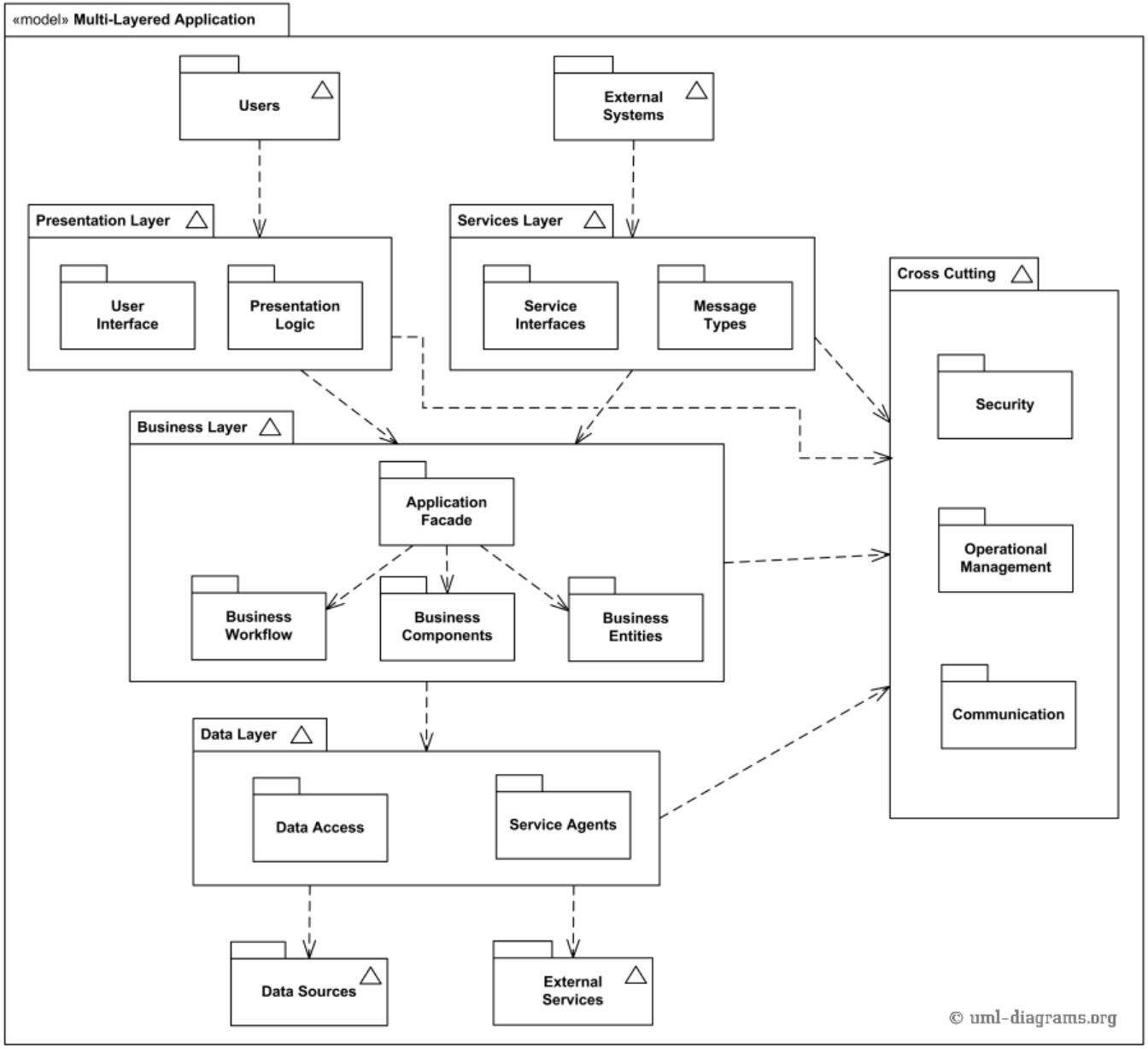
Prednosti

- Skalabilnost – jednostavno proširivanje (dodavanje novih kontrolera i pogleda)
- Jednostavnost dodavanja novih tipova korisnika (dodatni pogledi i kontroleri)
- Moguć istovremeni razvoj različitih komponenata
- Omogućava primjenu principa “Separation of Concerns”

Nedostaci

- Nije pogodan kod složenijih realizacija pogleda – nove web tehnologije
- Otežano održavanje – dosta koda u kontrolerima
- Otežano uvođenje novih nivoa apstrakcije

2. Dekompozicija sistema (nastavak)



Primjer dekompozicije

Multi-layered application UML model diagram example

2. Dekompozicija sistema (nastavak)

Primjer dekompozicije

UML package diagram example of a multi-layered web architecture