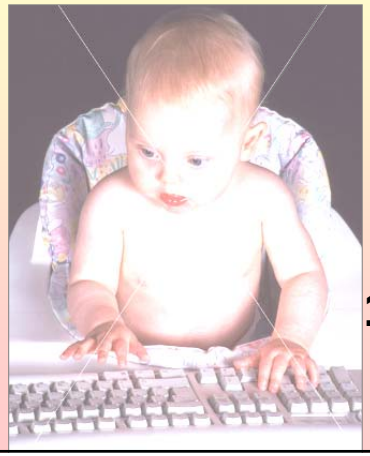
Dr Milunka Damnjanović, red.prof,
**OBJEKTNO ORIJENTISANE TEHNIKE PROJEKTOVANJA SISTEMA**



# 10 Pristupanje projektovanju

## Razlika između analize i projektovanja:

**Reč analiza potiče od grčke reči koja znači razlaganje na sastavne delove.**

**Aktivnost analize karakterisana je pitanjem ŠTA je potrebno novom sistemu**

**Projektovanje rezultuje rešenjem koje zadovoljava zahteve koji su analizirani.**

**Aktivnost projektovanja vezana je za razmatranja KAKO će novi sistem zadovoljiti zahteve.**

## Projektovanje:

**Projektovanje se može shvatiti ilči kao stepen u razvojnom životnom ciklusu sistema ili kao aktivnost koja se dešava u toku razvoja sistema.**

**Projekat se sastoji od glavnih faza (počinjanje, razrada, konstrukcija i izmena) koje iteriraju i zahtevaju sve više vremena kad se napreduje kroz projekat.**

**Analiza, projektovanje i konstrukcija su i faze i aktivnosti i velikim delom se prepliću.**

## Projektovanje u tradicionalnom životnom ciklusu:

**Prednosti:**

- **Upravljanje projektom.**
- **Veština i iskustvo zaposlenih mogu se razvijati koristiti za specijalizovane zadatke.**
- **Odluke klijenta – kraj analize obično je odlučujuća tačka u projektu.**
- **Izbor razvojnog okruženja odlaže se dok se analizom ne sagledaju potrebe sistema.**

## Projektovanje u iterativnom životnom ciklusu:

**Prednosti:**

- **Ublažavanje rizika.**
- **Promena menadžmenta**
- **Učenje tima.**
- **Poboljšan kvalitet.**

## Logičko i fizičko projektovanje:

**Dve faze projektovanja:**

**projektovanje *zavisno od implementacije* ili *logičko* projektovanje, i**

**projektovanje *nezavisno od implementacije* ili *fizičko* projektovanje.**

**Logičko projektovanje razmatra aspekte sistema koji nisu vezani za implementacionu platformu, a fizičko projektovanje razmatra aspekte koji zavise od implementacione platforme.**

## Sistemsko projektovanje i detaljno projektovanje:

**Projektovanje sistema odvija se na dva nivoa: sistemsko projektovanje i detaljno projektovanje.**

**Sistemsko projektovanje razmatra kompletnu arhitekturu sistema i postavlja standarde.**

**Detaljno projektovanje razmatra projektovanje individualnih komponenata koje se uklapaju u arhitekturu i zadoljavaju standarde.**

**Kod OO projektovanja, detaljno projektovanje odnosi se na projektovanje objekata.**

## OO detaljno projektovanje #1:

**Tradicionalno, detaljno projektovanje odnosi se na projektovanje ulaza, izlaza, procesa i struktura baza podataka.**

**Isti aspekti postoje i kod OO projektovanja i organizuju se pomoću klasa.**

***U toku faze analize projekta, koncepti u poslu treba da budu identifikovani i predstavljeni pomoću klasa, i korisnički slučajevi identifikovani i opisani. Uvedene su klase za upravljanje interfejsom sa korisnikom, interfejsom sa drugim sistemima, podacima i ukupnom koordinacijom drugih klasa u programima.***

## OO detaljno projektovanje #2:

**Coad i Yourdon (1991) dodaju tri komponente projektu klasa:**

- **komponentu interfejsa sa čovekom,**
- **komponentu upravljanja podacima, i**
- **komponentu upravljanja zadacima.**

## OO detaljno projektovanje #3:

**Coad et al. (1997) predlaže drugačiji skup komponenata:**

- **komponentu interfejsa sa čovekom,**
- **komponentu upravljanja podacima, i**
- **komponentu za interakciju sistema.**

## OO detaljno projektovanje #4:

**Larman (1998) predlaže arhitekturu u tri sloja:**

- **sloj prezentacije,**
- **logički sloj aplikacije, i**
- **memorijski sloj.**

## Bitni aspekti u OO detaljnom projektovanju:

**Ponovna upotreba projekta dešava se na dva nivoa: prvi, kroz upotrebu okvira projekta, a drugi kroz prepoznavanje klasa koje su već projektovane u organizaciji ili kupljene gotove kao komponente.**

**Dodela odgovornosti klasama je zadatak koji se odnosi na ponovnu upotrebu.**

**Pored ponovne upotrebljivosti, ima još mnogo drugih kriterijuma za kvalitetan projekat.**

## Kvalitet i ciljevi analize i projektovanja:

**Kvalitet projekta potpuno zavisi od kvaliteta analize koja mora da zadovolji četiri kriterijuma:**

***Ispravan delokrug rada*** **(šta sistem uključuje, a šta isključuje),**

***Potpunost*** **(sve što se zna o sistemu iz prikupljenih zahteva mora biti dokumentovano i uključeno u odgovarajuće dijagrame)**

***Ispravan sadržaj*** **(dokumentacija analize mora biti ispravna i precizna, u obliku teksta, dijagrama ili kvantitativnih prikaza).**

***Konzistentnost*** **-** ***usklađenost*** **(svi modeli, kao korisnički slučajevi, klase, atributi i operacije, koji se odnose na istu stvar treba da imaju isto ime).**

## Ciljevi i ograničenja #1:

**Dobar projekat je**

***efikasan,***
***fleksibilan,***
***upravljiv,***
***zadovoljavajući, i***
***produktivan;***

**a takođe i**

***funkcionalan,***
***prenosiv,***
***siguran i***
***ekonomičan.***

## Ciljevi i ograničenja projekta #2:

***Fukcionalan*****. Očekuje se korektno i kompletno izvršavanje funkcija koje su proklamovane.**

**Efikasan. Funkcionalnost treba da se ostvaruje efikasno i u pogledu vremena i u pogledu resursa.**

***Ekonomičan*****. Odnosi se ina cenu projektovanja i na cenu hardvera i softvera koji će biti neophodan za funkcionisanje sistema.**

***Pouzdan*****. Sistem mora biti pouzdan na dva načina: prvo, ne sme da bude podložan ni hardverskim ni softverskim greškama, i drugo, mora pozdano da održava integritet podataka u sistemu.**

## Ciljevi i ograničenja projekta #3:

***Siguran*****. Sistem mora biti siguran pod nasrtajima zlonamernika i neautorizovanih insajdera.**

***Fleksibilan*****. Mogućnost adaptiranja sistema promenama poslovnih zahteva tokom vremena naziva se i izmenljivost.**

***Opšti*****. Podrazumeva *proširivanje* da bi sistem postao sistem za opštu namenu, i *prenosivost* sistema.**

***Izgradljiv*****. Projekat mora biti jasan i bez nepotrebne kompleksnosti.**

## Ciljevi i ograničenja projekta #4:

*Upravljiv*. Dobar projekat mora da omogući menadžeru da proceni količinu rada potrebnu za implementiranje različitih podsistema.

*Održljiv*. Aktivnosti održavanja uključuju otkrivanje bagova, modifikovanje izveštaja i izgleda ekrana, poboljšanja programa da bi radio sa novim poslovnim zahtevima, premeštanje sistema na novi hardver i otkrivanje novih bagova uvedenih pri svim pomenutim akcijama.

*Upotrebljiv.* Sistem treba da bude zadovoljavajući i produktivan.

*Ponovna-upotrebljivost*. Postiže: ekonomski efekat, pomoć kao projektni okvir projektnih elemenata i ponovnu upotrebu klasa.

## Merivi ciljevi projekta:

- Bolji odgovor mušterijama.
- Povećanje udela u tržištu.
- Smanjivanje grešaka u računima.
- Povećanje broja obrađenih narudžbi u vršnim periodima.

## Sistemsko projektovanje #1:

Sistemsko projektovanje fokusira se na odluke visokog nivoa koje se tiču strukture celog sistema.

- U toku projektovanja sistema, izvode se sledeće aktivnosti:
- Identifikovanje podsistema i velikih komponenata.
- Identifikovanje prisutne konkurencije.
- Dodeljivanje podsistema procesorima.
- Odabiranje strategije upravljanja podacima.
- Odabiranje strategije za interakciju čovek-računar.

## Sistemsko projektovanje #2:

- Specificiranje standarda za razvijanje koda.
- Planiranje kontrolnih aspekata aplikacije.
- Pravljenje planova testiranja.
- Postavljanje prioriteta za projektne kompromise.
- Identifikovanje zahteva implementacije (npr. konverzije podataka).

## Arhitektura softvera:

*Arhitektura softvera predstavlja opis podsistema i komponenata softverskog sistema i odnosa između njih.*

U kontekstu OO razvoja, konceptualna arhitektura razmatra se sa strukturom statičkog klasnog modela i vezama između komponenata modela.

Logička arhitektura može da obuhvati klasni model, dok fizička arhitektura razmatra preslikavanje softvera na hardverske komponente. U svim slučajevima, ova dva različita aspekta kombinuju se da bi se definisala softverska struktura za sistem.

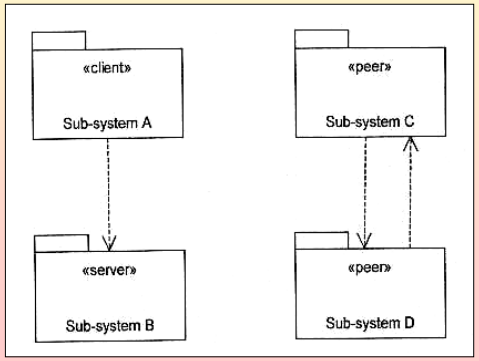
## Podsistemi:

Deljenje informacionog sistema na podsisteme ima sledeće prednosti:

- Formira manje jedinice u razvoju.
- Pomaže da se maksimizuje ponovna upotreba na nivou komponenata.
- Pomaže projektantima da se izbore sa kompleksnošću.
- Popravlja održljivost.
- Dodaje prenosivost.

## Komunikacija među podsistemima #1:

Dva tipa komunikacije:
*klijent-server i među-jednakim (peer-to-peer).*




## Komunikacija među podsistemima #2:

Komunikacija *klijent-server* zahteva da podsistem klijent zna interfejs podsistema servera, a *komunikacija je samo u jednom smeru: podsistem klijent zahteva usluge od podsistema server, a obrnuto nije moguće*.

Komunikacija među-jednakim (peer-to-peer) zahteva da svaki podsisten zna interfejs drugog, dakle vezuje ih čvršće. *Komunikacija je dvosmerna i svaki podsistem može da zahteva uslugu drugog.*

## Raslojavanje i deljenje:

**Postoje dva načina deljenja sistema na podsisteme:**

- **raslojavanje**
- **deljenje.**

## Slojeviti podsistemi #1:

**Arhitektura može biti *otvorena* i *zatvorena*.**

**Kod *zatvorene slojevite arhitekture*, neki sloj (npr. N) može da koristi usluge samo sloja ispod njega (sloja N-1).**

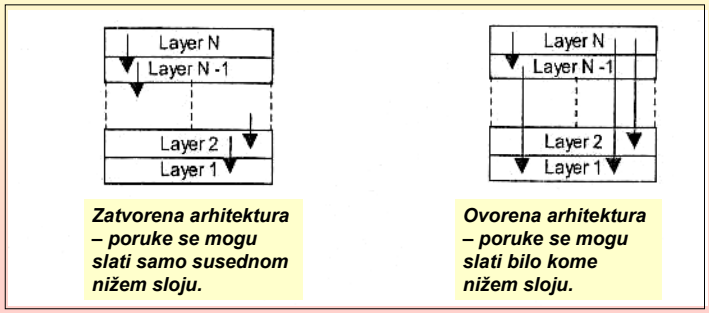
***Prednost:* Minimizovane zavisnosti između slojeva i redukovan uticaj promena interfejsa bilo kog sloja.**

**Kod *otvorene slojevite arhitekture*, neki sloj N može direktno da koristi usluge svih slojeva ispod njega.**

***Prednost*: Kod je mnogo kompaktniji jer uslugama svih slojeva može se direktno pristupati sa svih viših slojeva bez potrebe za ekstra programskim kodom za prenošenje poruka kroz sve međuslojeve.**

***Nedostatak*: Razbijanje inkapsulacije slojeva, povećana zavisnost između slojeva i povećane poteškoće kad treba menjkati sloj.**

## Slojeviti podsistemi #2:




***Zatvorena arhitektura – poruke se mogu slati samo susednom nižem sloju.***

***Ovorena arhitektura – poruke se mogu slati bilo kome nižem sloju.***

## Primena raslojavanja arhitekture:

**Potrebno je:**

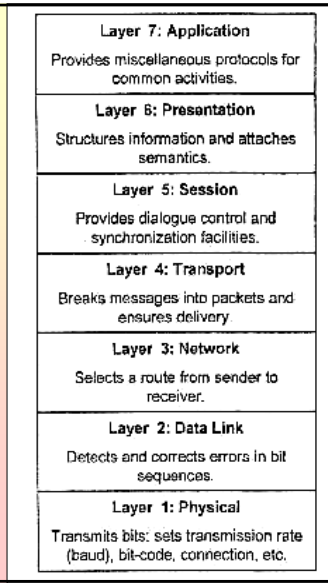
- **održavati stabilnost interfejsa svakog sloja,**
- **konstruisati ostale sisteme koji koriste neki od nižih slojeva,**
- **razmotriti varijacije nivoa granularnosti za podsistem,**
- **dalju podelu kompleksnih slojeva,**
- **redukciju osobina zbog arhitekture bliskih slojeva.**

## Raslojavanje podsistema:

**Svaki sloj odgovara jednom ili više podsistema koji se razlikuju međusobno po nivoima apstrakcije ili različitim fokusom funkcionalnosti.**

**Funkcionisanje: Najviši sloj koristi usluge (servise) koje obezbeđuje slij neposredno ispod njega. Ovaj može zahtevati usluge sloja koji je neposredno ispod njega, i tako dalje.**

## Mrežni protokol:




## Koraci u raslojavanju:

1. **Definisati kriterijum po kome će aplikacija biti grupisana po slojevima.**
2. **Odrediti broj slojeva.**
3. **Imenovati slojeve i dodeliti im funkcionalnost.**
4. **Specificirati usluge svakog servisa.**
5. **Poboljšati uslojavanje iteriranjem koraka 1 do 4.**
6. **Specificirati interfejs svakog sloja.**
7. **Specificirati strukturu svakog sloja.**
8. **Specificirati komunikaciju između svaka dva sloja.**
9. **Specificirati spregu između susednih slojeva.**

## Podeljeni podsistemi:

**Sistem može biti razbijen na podsisteme u toku analize zbog veličine ili kompleksnosti sistema.**

**Podsistemi analize mogu se razmatrati i u toku projektovanja zbog koherentnosti i kompatibilnosti sa kompletnom arhitekturom sistema.**

**Podsistemi nastali deljenjem moraju imati jasno definisane granice i dobro specificirane interfejse, tj. dobru inkapsulaciju tako da se ne menjaju ako se menjaju drugi podsistemi u okolini.**

## Arhitektura model-izgled-kontroler Model-View-Controller (MVC):

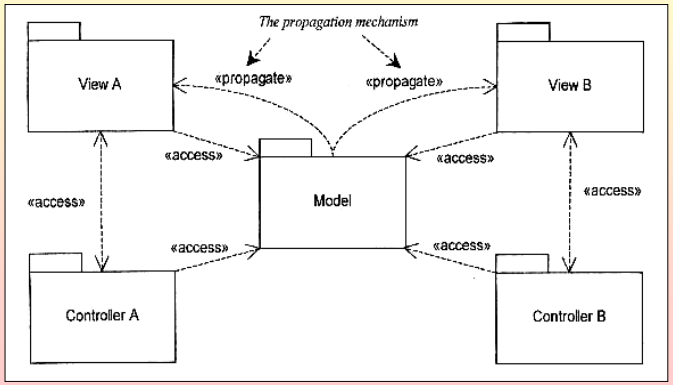
***Model***. **Model obezbeđuje centralnu funkcionalnost aplikacije i pazi na komponente koje pripadaju View-u i Controller-u.**

***View***. **Svaki View odgovara posebnom stilu i formatu prezentacije podataka korisniku. View uzima podatke od Model-a i obnavlja njihovu prezentaciju kada su podaci promenjeni u nekom drugom View-u. View kreira svoj sopstveni Controller.**

***Controller***. **Controller prima korisnikov ulaz u formi događaja koji okida izvršenje operacija unutar Model-a. Ovo može da izazove promene informacije i okine obnavljanje u svim View-ovima obezbeđujući da svi budu obnovljeni.**

***Mehanizam prostiranja***. **On dozvoljava Model-u da informiše svaki View da su promenjeni podaci o modelu i kao rezultat View mora da obnovi sebe. Često se zove mehanizam zavisnosti.**

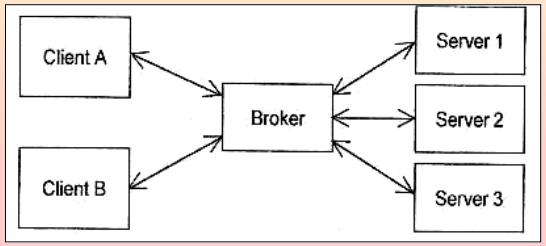
## Generalna struktura model-izgled-kontroler Model-View-Controller (MVC):




## Arhitektura distribuiranih sistema:

**Informacioni sistem može biti distribuiran po računarima na istoj ili na različitim lokacijama.**

**Generalna *broker* arhitektura povećava fleksibilnost komponenata klijent-server.**




## Struktura organizacije za arhitekturu i razvoj:

**Deljenje sistema na podsisteme omogućava bolje upravljanje projektom.**

**Svaki podsistem može se dodeliti jednom razvojnom timu koji može da radi nezavisno od ostalih.**

**Ako isti podsistem razvijaju dva razvojna tima, između njih mora da postoji intenzivna komunikacija. Zato se ti timovi spajaju u jedan tim.**

## Konkurentnost #1:

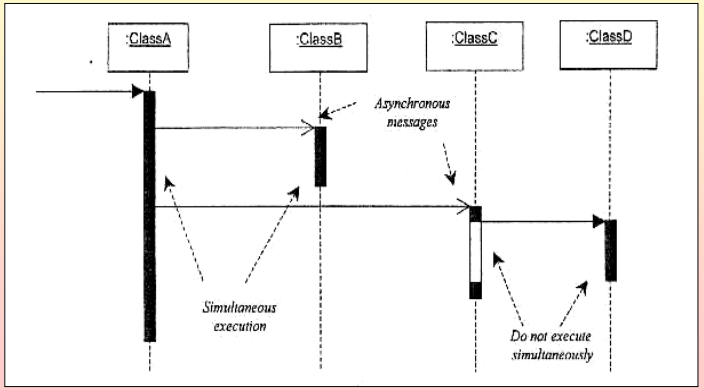
**Postoji više načina identifikovanja okolnosti kada je potrebno uvesti konkurentnu obradu:**

**1. Korisnički slučaj pokazuje zahtev da sistem treba da može da istovremeno odgovori na različite događaje od kojih svaki okida poseban kontrolni put (thread).**

**2. Ako se pokaže da klasa ima kompleksna ugnježdena stanja koja i sama imaju konkurentna podstanja, onda projekat mora da obezbedi rukovanje konkurentnošću.**

**3. Kada postoji zahtev da objekat ima konkurentno ponašanje, ponekad je neophodno podeliti objekat na posebne objekte da bi se izbegla potreba za konkurentna aktivnost u bilo kome objektu.**

## Konkurentna aktivnost u dijagramu interakcije:



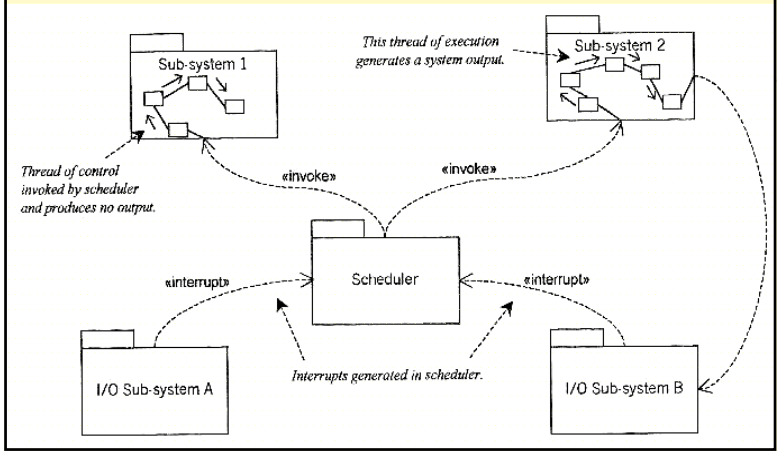

## Konkurentnost #2:

**Razlika između logičke i fizičke konkurentnosti: postoji mnogo načina simuliranja postojanja više procesora koristeći samo jedan fizički procesor.**

**Kada ne postoje tesna vremenska ograničenja, multi-tasking operativni sistem može da postigne zadovoljavajuću realizaciju konkurentnosti.**

**Kada postoje tesna vremenska ograničenja, uvodi se podsistem za vremensko raspoređivanje (scheduler) da bi se obezbedilo da svaki kontrolni put (thread) funkcioniše unutar njegovih ograničenja u pogledu vremena odgovora.**

## Konkurentnost upravljana raspopređivačem:



## Konkurentnost #3:

Konkurentnost se može implementirati multi-thread programskim jezikom (kao Java). Ovo omogućava direktnu implementaciju konkurentnosti unutar jednog procesorskog zadatka.

Multiprocesorska okolina omogućava da svaki konkurentni zadatak bude implementiran na posebnom procesoru.

## Dodeljivanje sistema višestrukim procesorima:

- Aplikaciju podeliti na podsisteme.
- Proceniti zahteve za procesiranje za svaki podsistem.
- Odrediti kriterijum pristupa i zahteve lokacije.
- Identifikovati zahteve konkurentnosti za podsisteme.
- Svaki podsistem dodeliti jednoj radnoj platformi – ili opšte namene (PC ili radna stanica) ili specijalizovanoj (ugrađenom mikrokontroleru ili specijalnom serveru).
- Odrediti zahteve za komunikaciju između podsistema.
- Specificirati infrastrukturu za komunikaciju.

## Upravljanje podacima:

Sistemi za upravljanje podacima pružaju razne mogućnosti:

- različiti korisnici različiro vide podatke,
- kontrola višekorisničkog pristupa,
- distribucija podataka preko različitih platformi,
- sigurnost,
- nametanje ograničenja integriteta,
- pristupanje podacima od različitih aplikacija,
- oporavak podataka,
- prenosivost preko platformi,
- pristup podacima preko jezika upita, i
- optimizacija upita.

## Prioriteti u projektnim kompromisima:

Osnove prioriteta moraju se definisati u dogovoru sa klijentom.

Osnove za projektne kompromise obezbeđuju konzistentnost među odlukama učinjenim na različitim stepenima razvoja sistema.

Takođe, oni obezbeđuju konzistentnost između različitih podsistema.

Ne postoje osnovi kompromisa za sve slučajeve.

## Zaključak:

**Identifikovali smo sledeće aktivnosti kao relevantne za projektovanje sistema:**

- **Identifikovanje podsistema i krupnih komponenata.**
- **Identifikovanje prisutne konkurentnosti.**
- **Dodeljivanje podsistema procesorima.**
- **Odabiranje strategije upravljanja podacima.**
- **Specificirati standarde razvoja.**
- **Postaviti prioritete za projektne kompromise.**
- **Identifikovati zahteve implementacije (t.j. konverzije podataka).**