Softversko inženjerstvo

Projektovanje softvera

dr Miloš Stojanović*

Visoka tehnička škola strukovnih studija Niš

2017.

*milos.stojanovic@vtsnis.edu.rs

Projektovanje softvera

- Projektovanje SW-a je faza u životnom ciklusu razvoja softvera koja sledi fazu specifikacije zahteva.
- Daje odgovor na pitanje **KAKO** realizovati delove sistema.
- Projektovanje SW-a definiše kako SW treba da radi i sadrži dve podfaze:

➢ Arhitekturno projektovanje i

➢ Detaljno projektovanje.

Arhiteturno projektovanje SW-a

- Arhitekturno projektovanje podrazumeva definisanje arhitekture softvera sastavljene od međusobno zavisnih softverskih modula i komponenti, **njihovih vidljivih osobina** i specifikacija njihovih međusobnih veza.
- Vidljive osobine su:

➢ Skup funkcionalnosti.

➢ Performanse.

➢ Upravljanje izuzecima.

➢ Korišćenje deljivih resursa.

Detaljno projektovanje SW-a

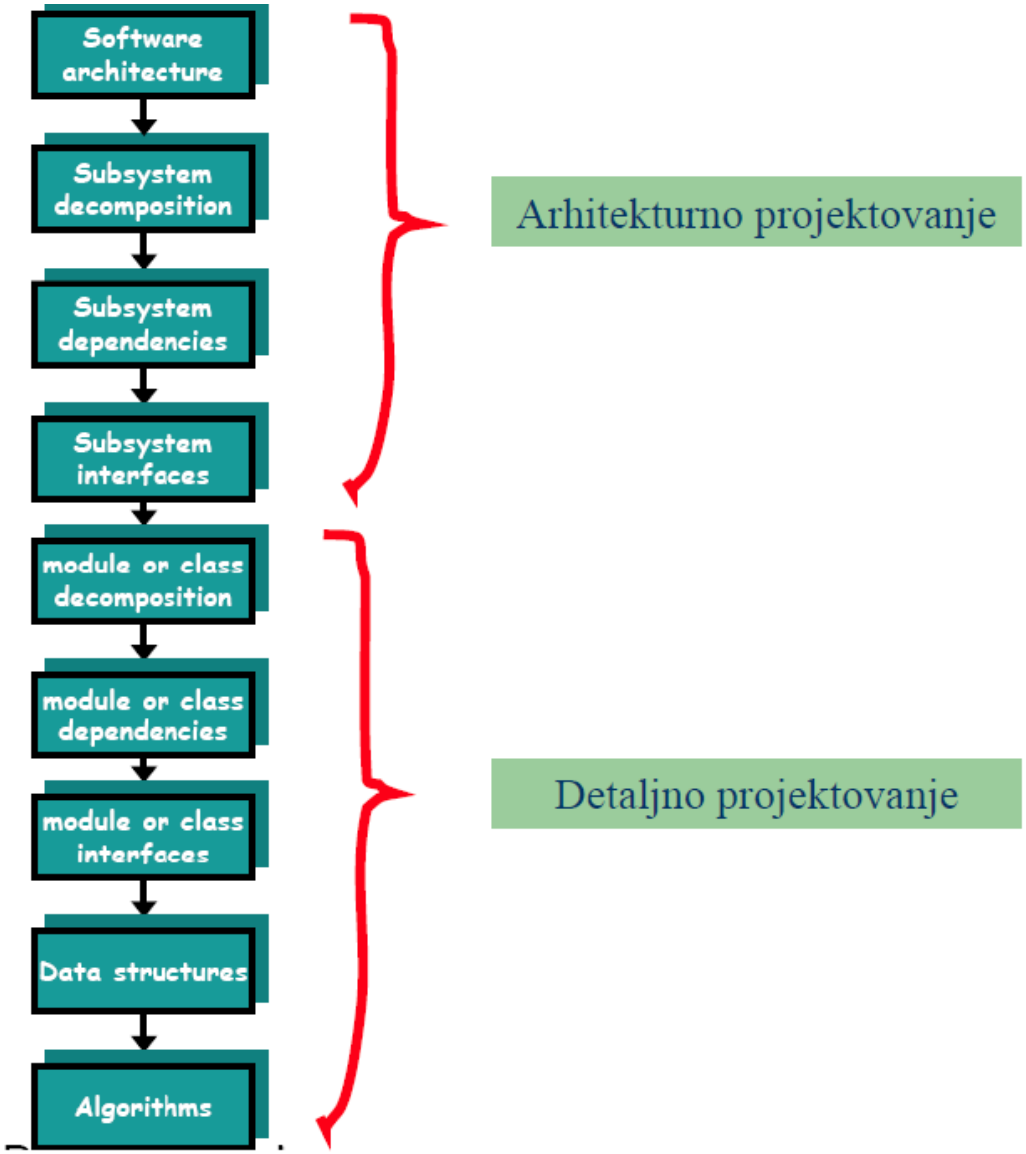
- Detaljno projektovanje podrazumeva:

➢ Projektovanje struktura podataka i baza podataka,

➢ Projektovanje korisničkog interfejsa za interakciju korisnika sa softverskim sistemom i

➢ Projektovanje algoritama za funkcije i module.

Projektovanje SW-a



Dobro projektovan SW

- Vrlo je teško definisati šta je dobar projekat.
- To često zavisi od prioriteta firme koja razvija SW i od prioriteta korisnika (visoka pouzdanost, najkraće vreme, lakoća implementacije...)
- *"I know a good design when I see it"*

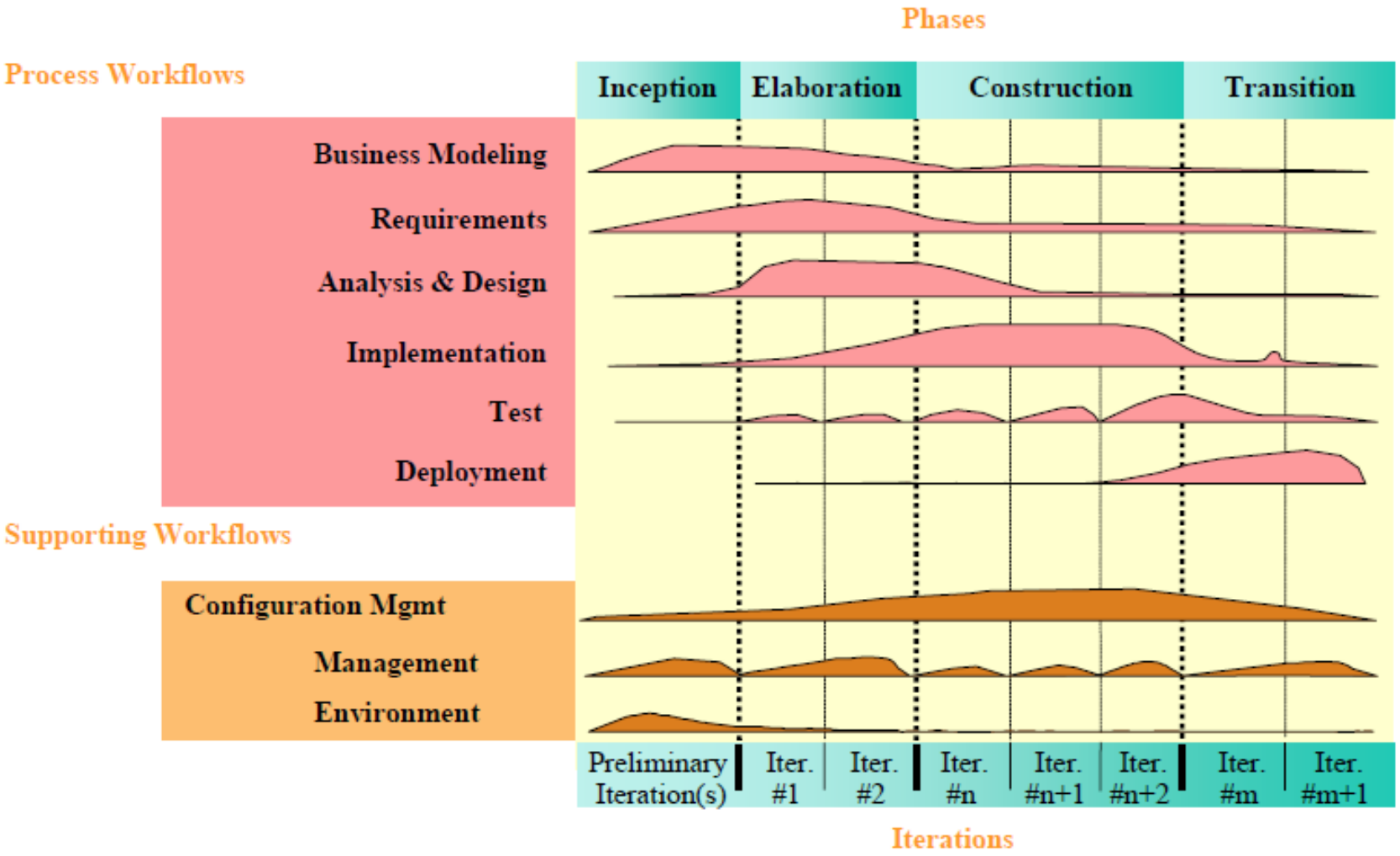
Osobine dobrog projekta

- Hijerarhija – dobar projekat bi trebalo da bude organizovan u dobro projektovanu hijerarhiju komponenata.
- Modularnost – treba izvršiti dekompoziciju sistema u posebne celine – module (na primer, razdvojiti podatke i obradu) sa jasno definisanim interfejsom.

Osobine dobrog projekta

- Nezavisnost – treba grupisati slične stvari u nezavisne module. Ako se kasnije menjaju neke bitne odluke u projektu, posledice će biti lokalizovane.
- Jednostavan interfejs – treba izbeći komplikovan korisnički interfejs, interfejse sa velikim brojem mogućnosti upotrebe kao I interfejse čija bi izmena mogla izazvati neželjene efekte.

Projektovanje SW-a i RUP



Projektovanje SW-a i RUP

- Arhitekturno projektovanje se sprovodi u fazi **elaboracije** i podrazumeva:

➢ Definisanje arhitekture sistema.

➢ Definisanje najbitnijih klasa.

➢ Realizaciju arhitekturno najznačajnijih slučajeva korišćenja.

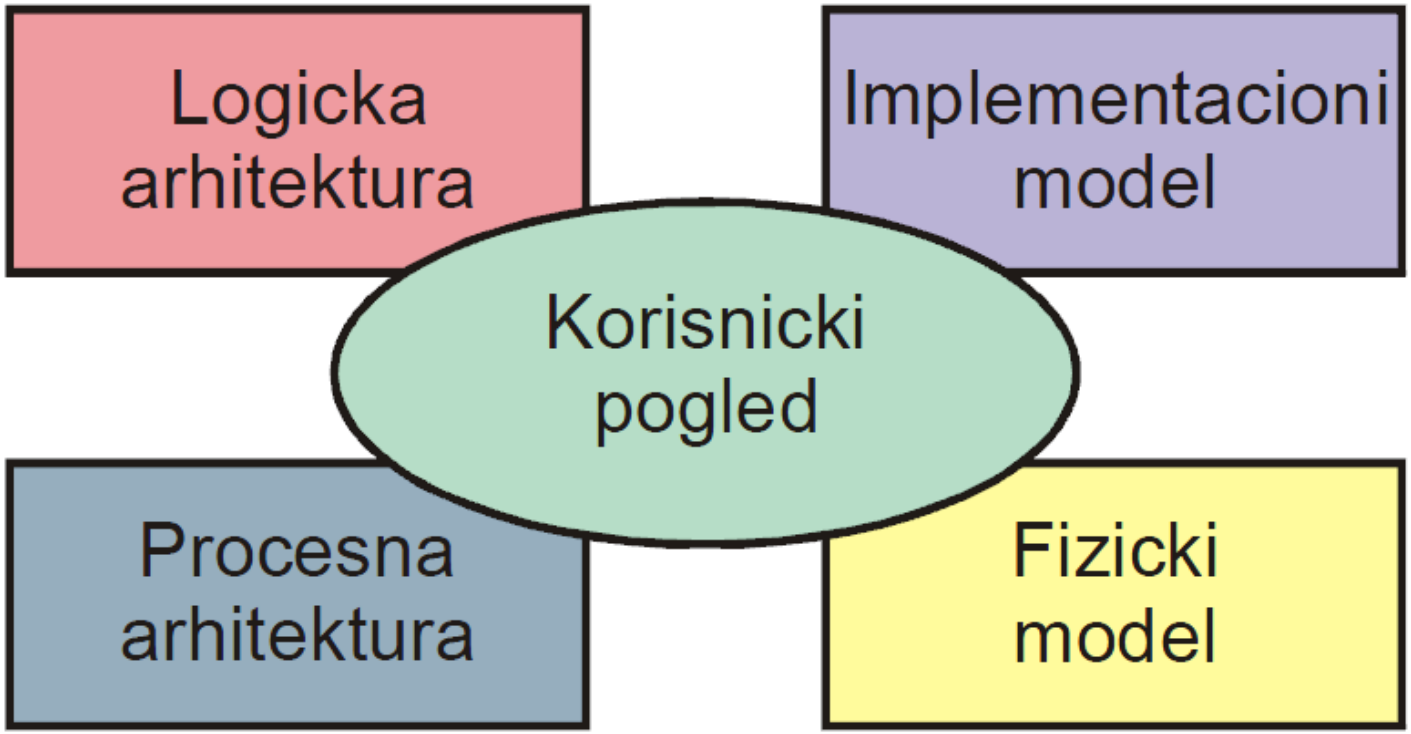
➢ Koriste se UML dijagrami klasa.

- Dokument koji je izlaz iz ove faze je arhitekturni projekat sistema.

Projektovanje SW-a i RUP

- Detaljno projektovanje se sprovodi u fazi **izrade** i podrazumeva:
- Arhitekturni projekat razvijen u detalje.
- Detaljne dijagrame klasa.
- “4+1” model sistema.
- Dokument koji je izlaz iz ove faze je detaljni projekat sistema.

“4+1” Model sistema



Metode projektovanja SW-a

- Postoje tri osnovne grupe metoda za projektovanje SW-a:

➢ *Strukturne metode.*

➢ *OO metode.*

➢ *Kombinovane metode.*

Strukturne metode projektovanja

- One su funkcionalno orijentisane jer se softverski sistem projektuje sa funkcionalnog stanovišta u vidu osnovnih funkcija koje obavlja nad podacima koji čine stanje sistema.
- Podržane su od strane mnogih CASE alata I grafičkih notacija.

OO metode projektovanja

- Softverski sistem se posmatra kao kolekcija objekata u međusobnoj interakciji, pri čemu svaki objekat obuhvata određene podatke i funkcije za manipulisanje tim podacima.
- Uglavnom se koristi UML grafička notacija.

Kombinovane metode projektovanja

- Kombinovane (hibridne) metode uključuju dobre osobine i strukturnih i OO metoda kako bi se došlo do dobro projektovanog softverskog sistema.

Tehnike projektovanja SW-a

- Dva su osnovna pristupa (tehnike) projektovanja SW-a:

➢ Top-down projektovanje.

➢ Bottom-up projektovanje.

Top-down projektovanje SW-a

- Polazi se od vrha sistema, odnosno od najviših slojeva i onda se polako dolazi do podsistema koji su na nižim nivoima.
- Loša strana ove tehnike je ta što forsira razvoj poedinih grana sistema dok neke druge nisu ni započete.
- Takođe, ova tehnika ne sagledava na pravi način već postojeće komponente koje se mogu iskoristiti.

Bottom-up projektovanje SW-a

- Polazi se sa dna sistema, odnosno od najnižih slojeva i onda se polako dolazi do podsistema koji su na višim nivoima.
- Obično se kreće od gotovih komponenata koje se povezuju kako bi se realizovali neki delovi sistema.
- Loša strana ove tehnike je ta što se najviši slojevi sistema (a koje korisnik direktno vidi) dobijaju u kasnim fazama implementacije.

Arhitekturni modeli (stilovi)

- Arhitekturni modeli (stilovi) predstavljaju projektne obrasce za arhitekturu SW-a.
- Oni definišu komponente i konektore koji čine arhitekturu sistema.
- Na ovaj način, arhitektura sistema se može predstaviti kao graf čiji su čvorovi sledeće komponente:

➢ Procedure;
➢ Moduli;
➢ Procesi;
➢ Alati;
➢ Baze podataka;

Arhitekturni modeli (stilovi)

- Grane (potege) grafa čine konektori koji mogu biti:

➢ Pozivi procedura;

➢ Prenosi događaja (evenata);

➢ Upiti baze podataka;

➢ Protočne obrade.

Osnovni arhitekturni modeli

- Repository (Skladište)
- Pipe and Filter (Protočna obrada)
- OO model
- Client/Server model
- Slojeviti model (Layered)
- Event-driven model (Implicitno pozivanje)
- Control model

Repository model

- Ovaj model se koristi kod sistema kod kojih je neophodno deljenje velikih količina podataka.
- Tada se organizuje centralno skladište podataka kome pristupaju podsistemi.
- **Komponente:**

➢ Centralna baza podataka.

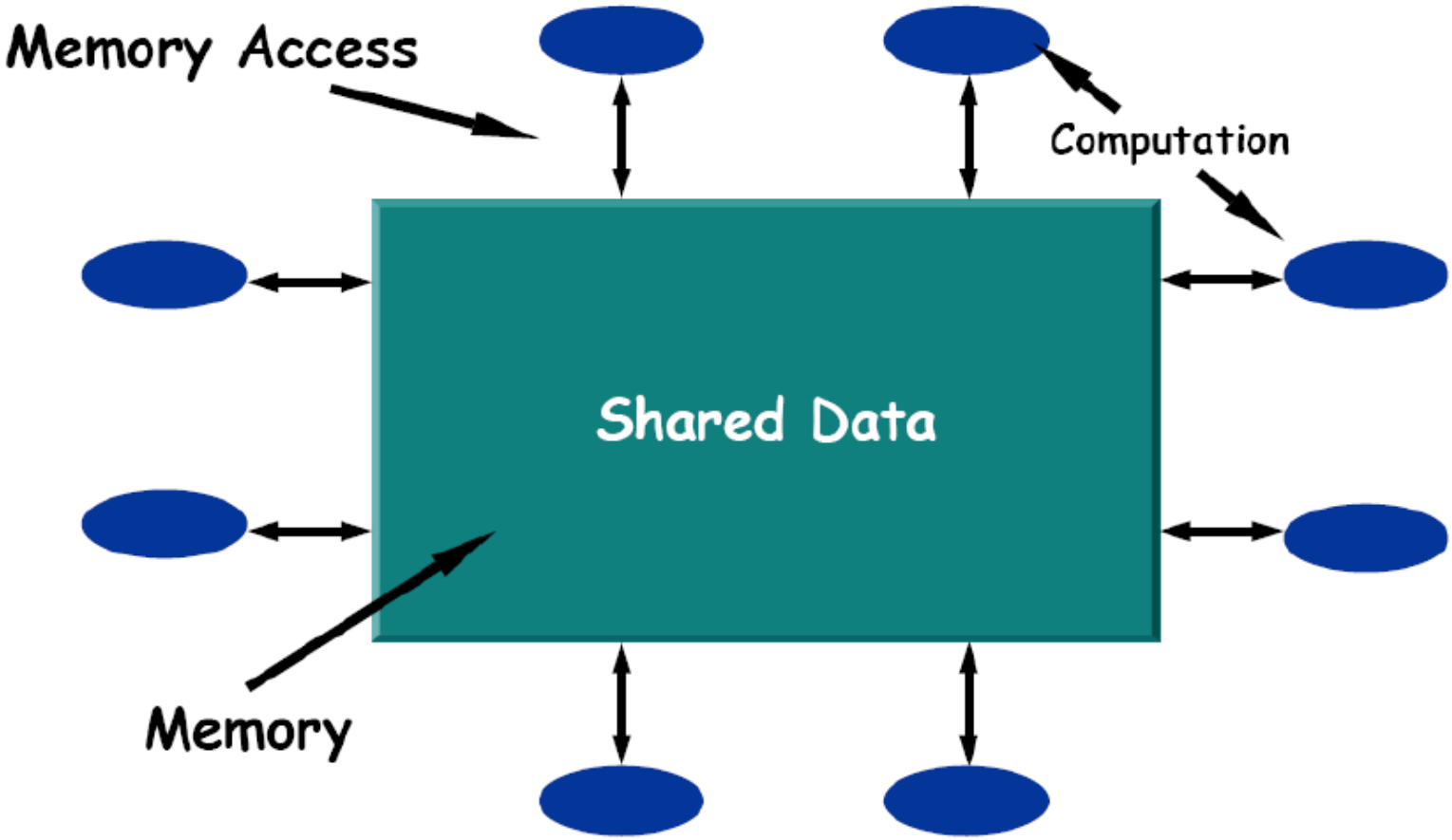
➢ Skup SW komponenti koje pristupaju centralnoj bazi.

- **Konektori:**

➢ Pozivi procedura.

➢ Direktan pristup bazi podataka.

Repository model



Repository model - karakteristike

- Prednosti

➢ Efikasan način za deljenje velike količine podataka;

➢ Podsistemi ne moraju da brinu o nekim aspektima upravljanja podacima kao što su backup, sigurnost,...

➢ Model deljenja podataka je prikazan u obliku šeme skladišta podataka.

- Nedostaci

➢ Podsistemi moraju da se slože oko skladišta podataka. Kompromis je neizbežan;

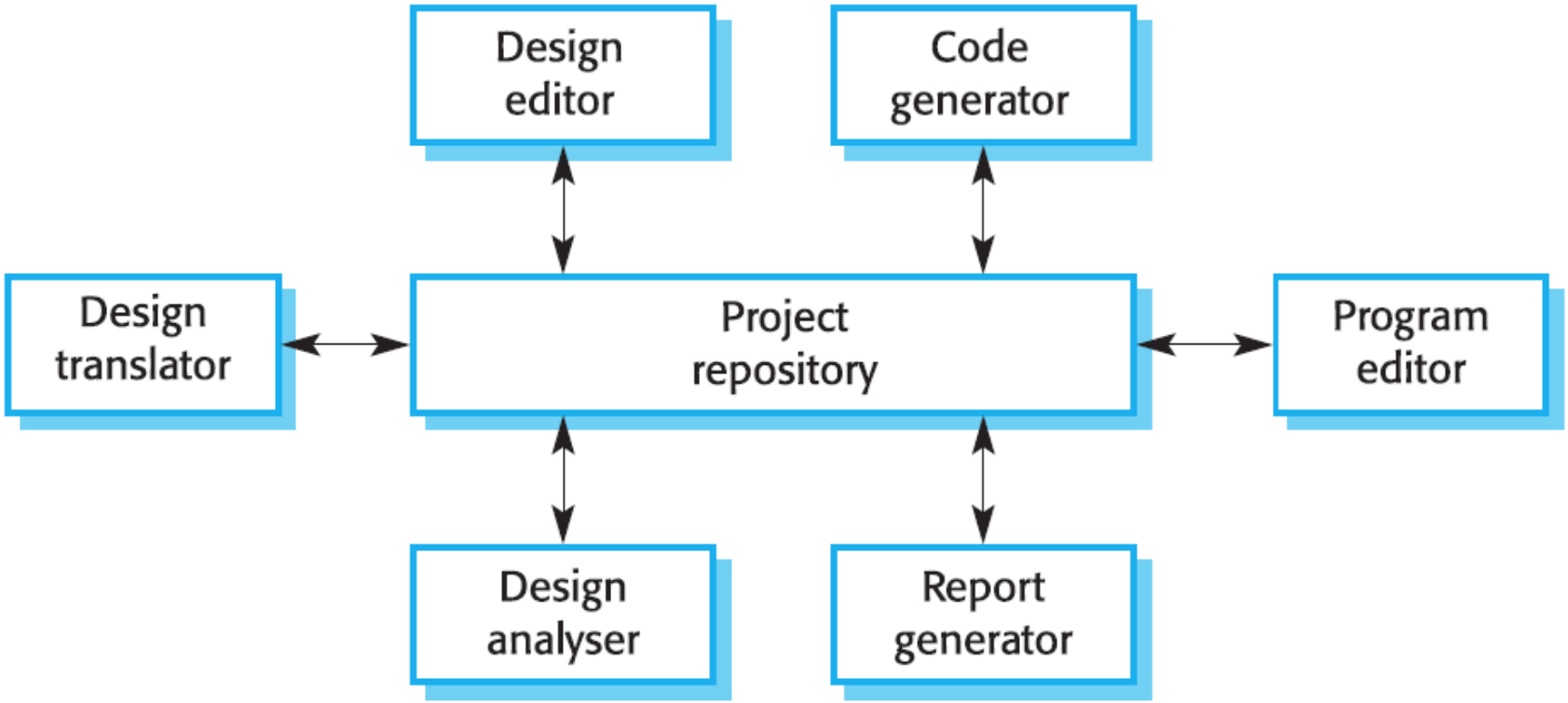
➢ Evolucija podataka je skupa;

➢ Otežana je distribucija podataka.

Repository model - primeri

- Informacioni sistemi
- Okruženja za razvoj SW-a
- Grafički editori
- Baze znanja u sistemima veštačke inteligencije
- Sistemi za reverse engineering

Repository model - primer



http://www.cs.ccsu.edu/~stan/classes/CS530/Notes14/06-ArchitecturalDesign.html

Pipe and Filter model (protočna obrada)

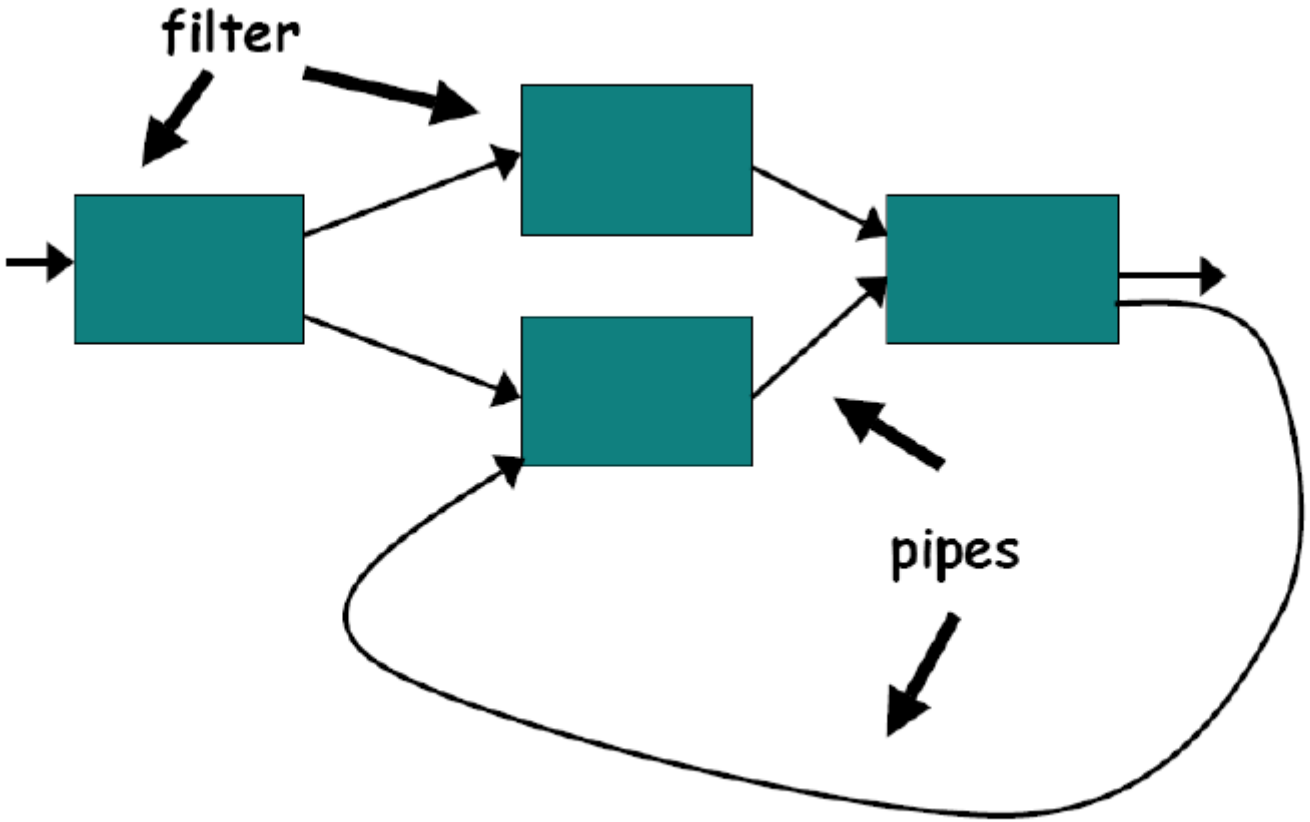
- Ovaj model se koristi kod sistema kod kojih je neophodno izvršiti unapred definisane serije nezavisnih obrada nad podacima.
- Komponente prihvataju tokove podataka na ulazu i generišu tokove podataka na izlazu.
- Komponente:

➢ Komponente se često zovu i filtri koji vrše određene transformacije ulaznih podataka.

- Konektori:

➢ Konektori se često zovu pipe (cevovod) jer povezuju tokove i filtre (izlaz jednog filtra vode na ulaz drugog filtra).

Pipe and Filter model (protočna obrada)



Pipe and Filter model - karakteristike

- Prednosti

➢ Lako razumevanje ulazno-izlaznog ponašanja celokupnog sistema kao skupa individualnih ponašanja filtera u sistemu;

➢ Lako je ponovno korišćenje filtera (reuse), jer je između dva filtra već definisan format razmene podataka.

➢ Laka je izmena ili proširenje sistema (zamenom postojećih ili dodavanjem novih filtera).

➢ Prirodno je podržano konkurentno izvršenje.

- Nedostaci

➢ Nije najbolji izbor kod interaktivnih sistema zbog velikog broja transformacija;

➢ Povećava kompleksnost sistema i smanjuje efikasnost;

Pipe and Filter model - primeri

- Tradicionalni kompajleri (prevodioci)

lexical analysis + parsing + semantic analysis + code generation

- Unix shell skriptovi

https://en.wikipedia.org/wiki/Pipeline_(Unix)

https://en.wikipedia.org/wiki/Pipeline_(computing)

OO model

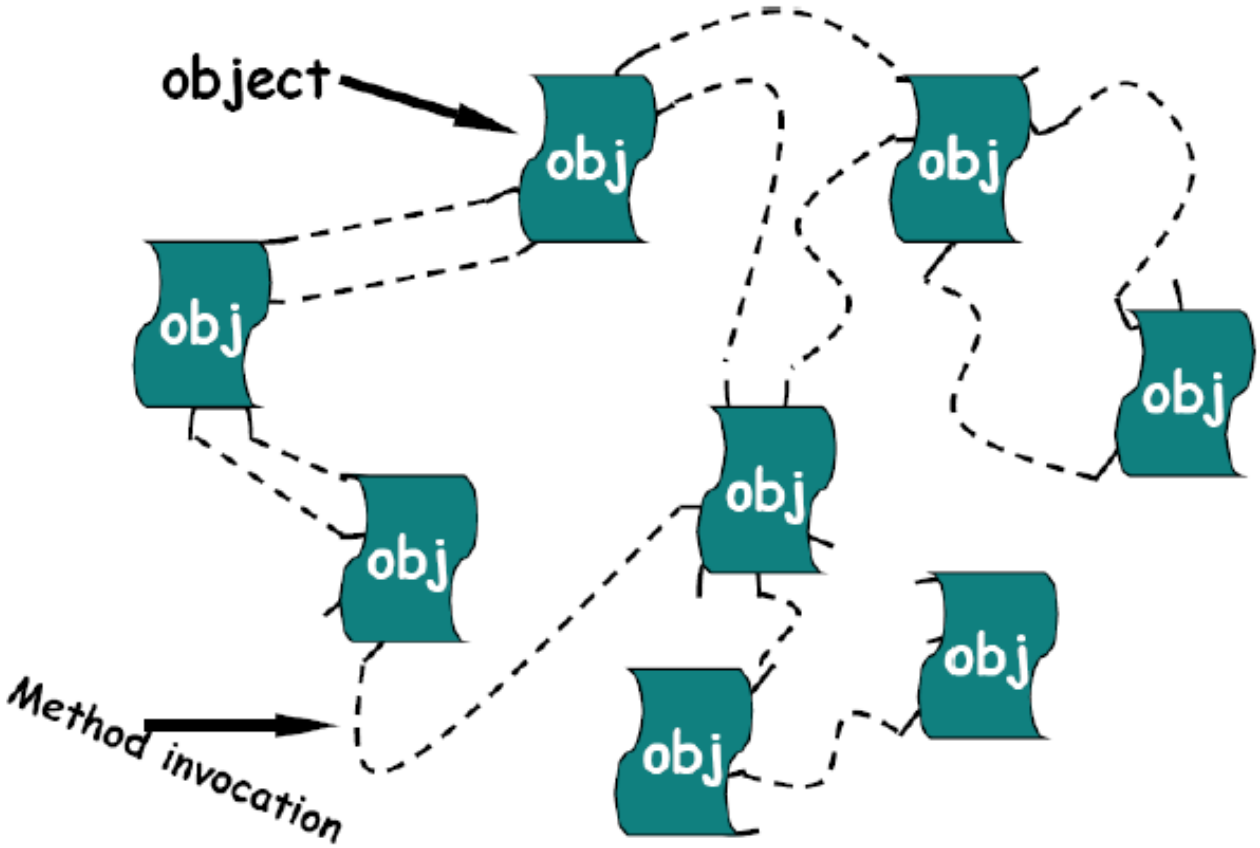
- Ovaj model se koristi kod sistema kod kojih je neophodno izvršiti zaštitu i enkapsulaciju podataka.
- Podaci i pridružene operacije su enkapsulirane u apstraktne tipove podataka (klase i objekte).
- Komponente:

➢ Objekti

- Konektori:

➢ Metodi (servisi) objekata.

OO model



OO model - karakteristike

- Prednosti

➢ Zbog skrivanja informacija od klijenata, moguće je promeniti implementaciju objekata bez uticaja na klijente;

➢ Moguće je projektovanje sistema kao skupa autonomnih agenata.

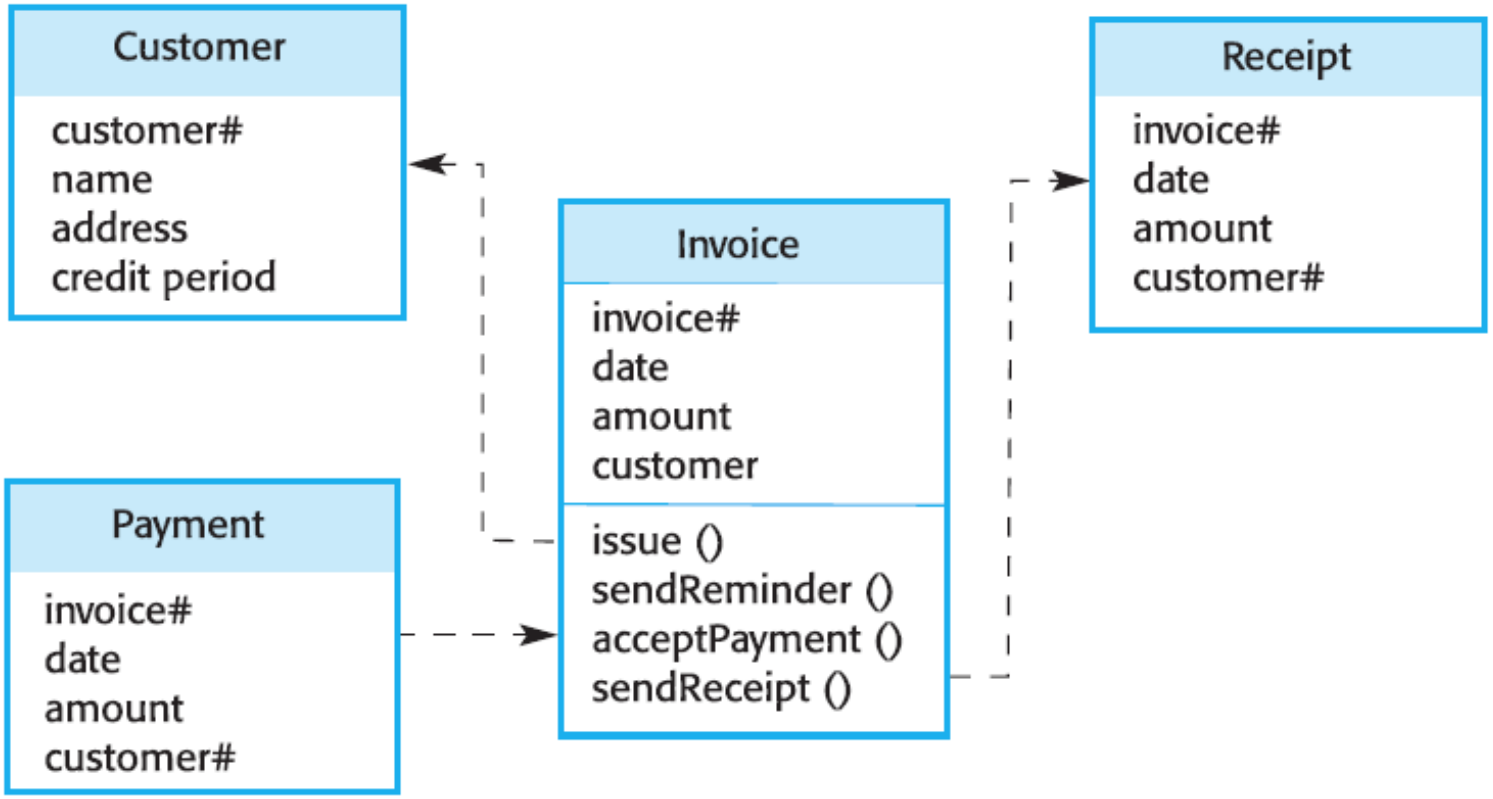
➢ Moguće je direktno mapiranje entiteta iz realnog sveta u objekte.

- Nedostaci

➢ Neophodno je poznavanje indentiteta objekata. Ukoliko se izmeni identitet nekog objekta, moraju se izvršiti izmene i kod svih objekata koji ga pozivaju;

➢ Mogućnost pojave “bočnih efekata”;

OO model - primer



Client-Server Model

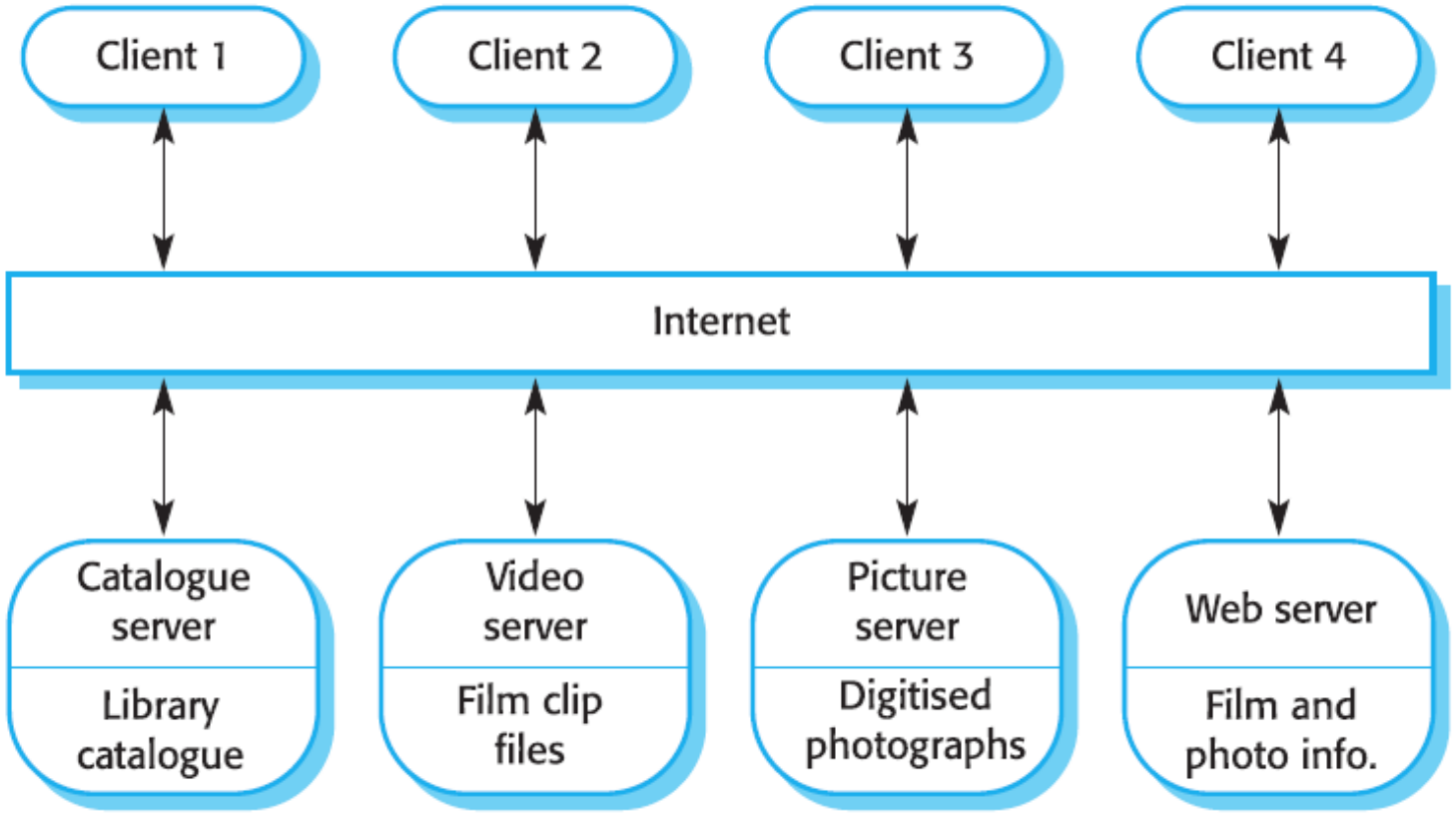
- Ovaj model se koristi kod distribuiranih sistema.
- Sastoji se od skupa stand-alone servera koji obezbeđuju specifične servise (štampa, Web, baza podataka,...), skupa klijenata koji pozivaju te servise i mreže koja omogućuje udaljeni pristup.
- Komponente:

➢ Serveri, klijenti.

- Konektori:

➢ Mreža, servisi servera.

Client-Server Model - primer



Client-Server model - karakteristike

- Prednosti
  - Efikasno korišćenje mrežnih sistema;
  - Omogućava korišćenje slabijeg HW-a za klijente, obzirom da server odrađuje većinu posla.
  - Lako dodavanje novih servera i upgrade postojećih.
- Nedostaci
  - Neefikasna razmena podataka između klijenata (moraju da idu preko servera);
  - Redundantnost podataka.
  - Ne postoji centralni registar imena servera i servisa; Nije lako otkriti koji serveri i servisi su na raspolaganju.

Slojeviti (Layered) Model

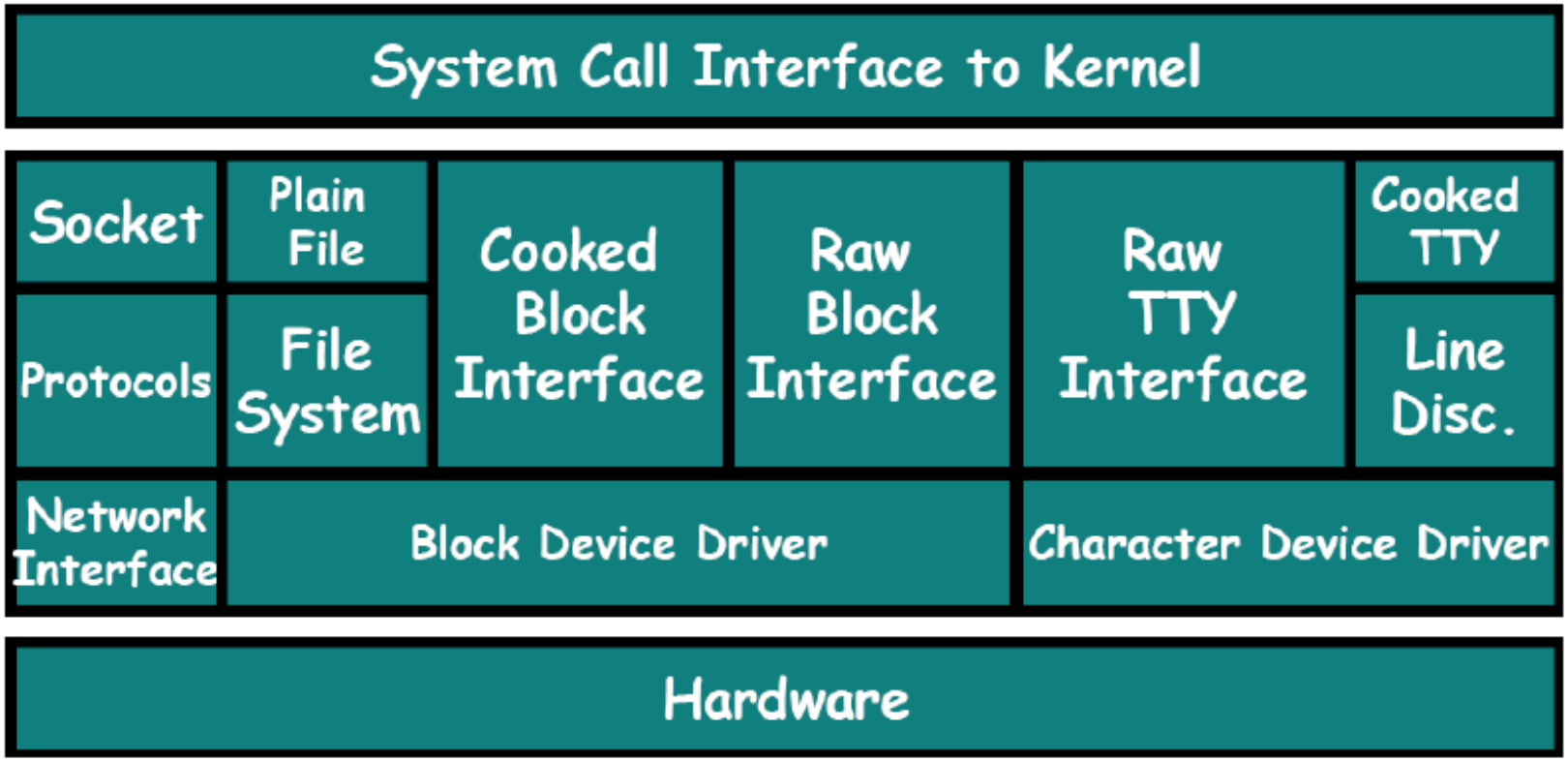
- Ovaj model se koristi kod modeliranja interfejsa među podsistemima.
- Sistem se organizuje u skup slojeva (layera) od kojih svaki obezbeđuje jedan skup funkcionalnosti sloju iznad i služi kao klijent sloju ispod.
- Omogućava inkrementalni razvoj podkomponenti u različitim slojevima.

➢ Komponente:

- Slojevi.
- Konektori:

➢ Interfejsi.

Layered model – primer OS Unix



Layered model - karakteristike

- Prednosti

➢ Promena interfejsa jednog sloja može da utiče na maksimalno još dva sloja;

➢ Laka zamena jednog sloja drugim ukoliko su im interfejsi identični.

➢ Baziran je na visokom nivou apstrakcije.

- Nedostaci

➢ Ne mogu svi sistemi da se lako organizuju po ovom modelu;

Event driven model (Implicitno pozivanje)

- Ovaj model se koristi kod sistema koji su upravljani eksterno generisanim događajima (events).
- Postoje dve osnovne grupe ovih modela:

➢ Broadcast modeli

➢ Interrupt-driven modeli

- Komponente:

➢ Komponente i podsistemi koji generišu ili obrađuju evente.

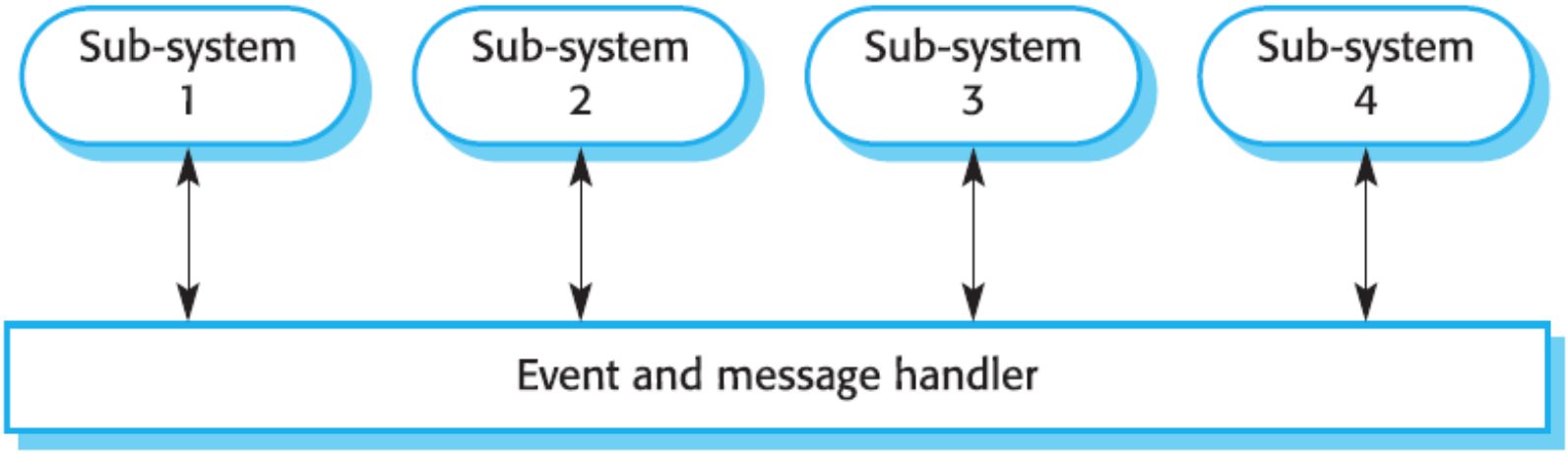
- Konektori:

➢ Broadcast sistem i event procedure.

Broadcast modeli

- Kod ove grupe modela, generisani događaj (event) se prosleđuje svim komponentama i podsistemima u sistemu. Svaka komponenta koja upravlja generisanim događajem može da obradi događaj.
- Efikasni su kod integracije podsistema koji se nalaze na različitim računarima u mreži.
- Podsistemi neznaju da li će i kada eventi biti obrađeni.
- Podsistemi se registruju za određene događaje i kada se oni generišu, upravljanje se prenosi na podsistem koji upravlja tim događajem.

Broadcast model – primer



Broadcast model – primer razvojnog okruženja

- Ovaj model se često koristi kod razvojnih okruženja za integraciju alata:

➢ Debager se zaustavi na prekidnoj tački i generiše događaj da je to uradio.

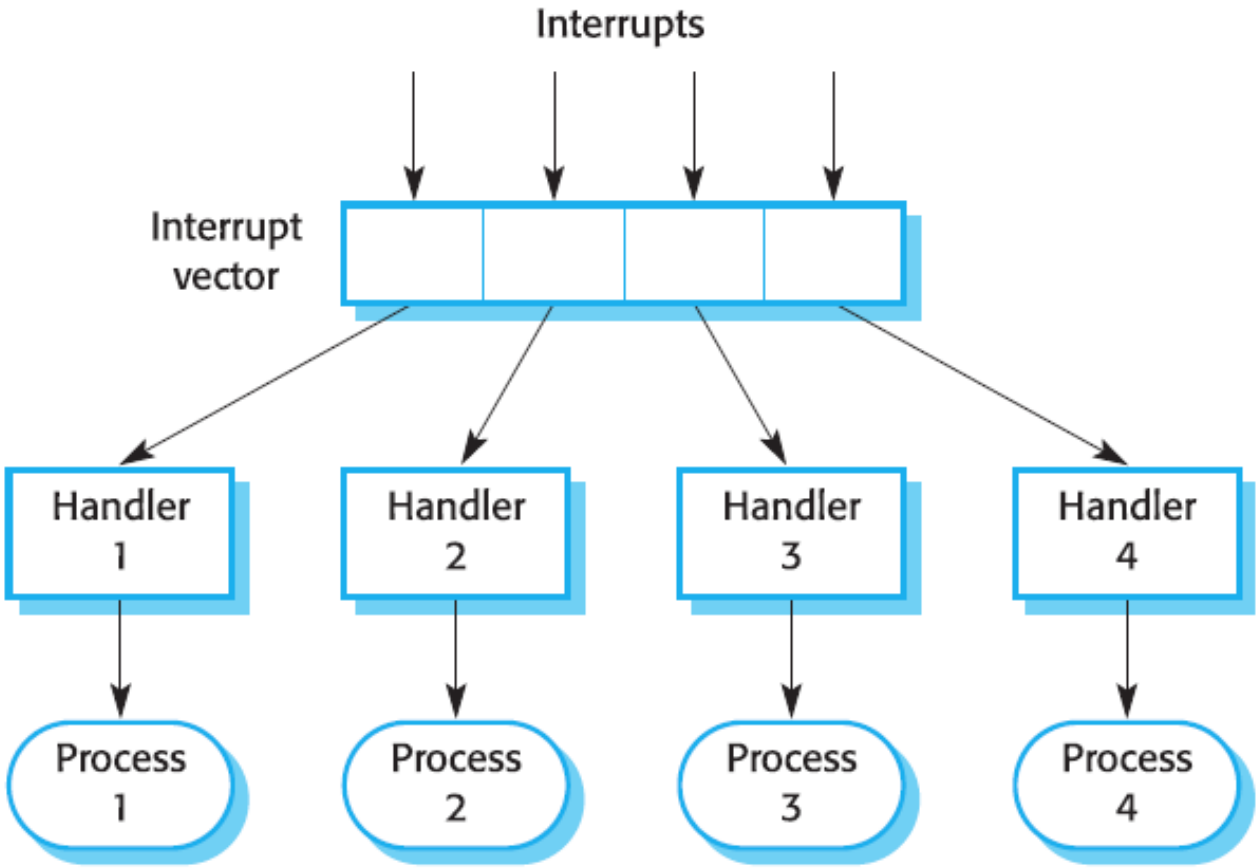
➢ Editor odgovara na taj događaj tako što skroluje sadržaj koda na liniju gde je postavljena prekidna tačka.

➢ ...

Interrupt-driven modeli (modeli upravljani prekidima)

- Koriste se kod sistema za rad u realnom vremenu gde je osnovna stvar brzi odgovor sistema na neki događaj.
- Obezbeđuju brzu reakciju sistema na događaje, ali su komplikovani za realizaciju i pogotovo za testiranje i validaciju sistema.

Interrupt-driven model – primer



Event driven model - karakteristike

- Prednosti
  - Podrška višestrukom korišćenju SW-a (reuse);
  - Laka evolucija sistema.
  - Lako uvođenje nove komponente u sistem (jednostavno se registruje za neki event).
- Nedostaci
  - Kada komponenta generiše događaj ona ne može da zna da li će neka komponenta da odgovori na njega i kada će obrada događaja biti završena;

Control model

- Koristi se kod sistema gde je potrebna centralizovana kontrola.
- Kontrolni podsistem upravlja tokom informacija između ostalih podsistema.
- Postoje četiri osnovne grupe ovih modela:

➢ Call-return modeli
➢ Manager modeli
➢ Feed-back modeli
➢ Open-loop modeli

- Komponente:

➢ Kontrolni algoritam i podsistemi.

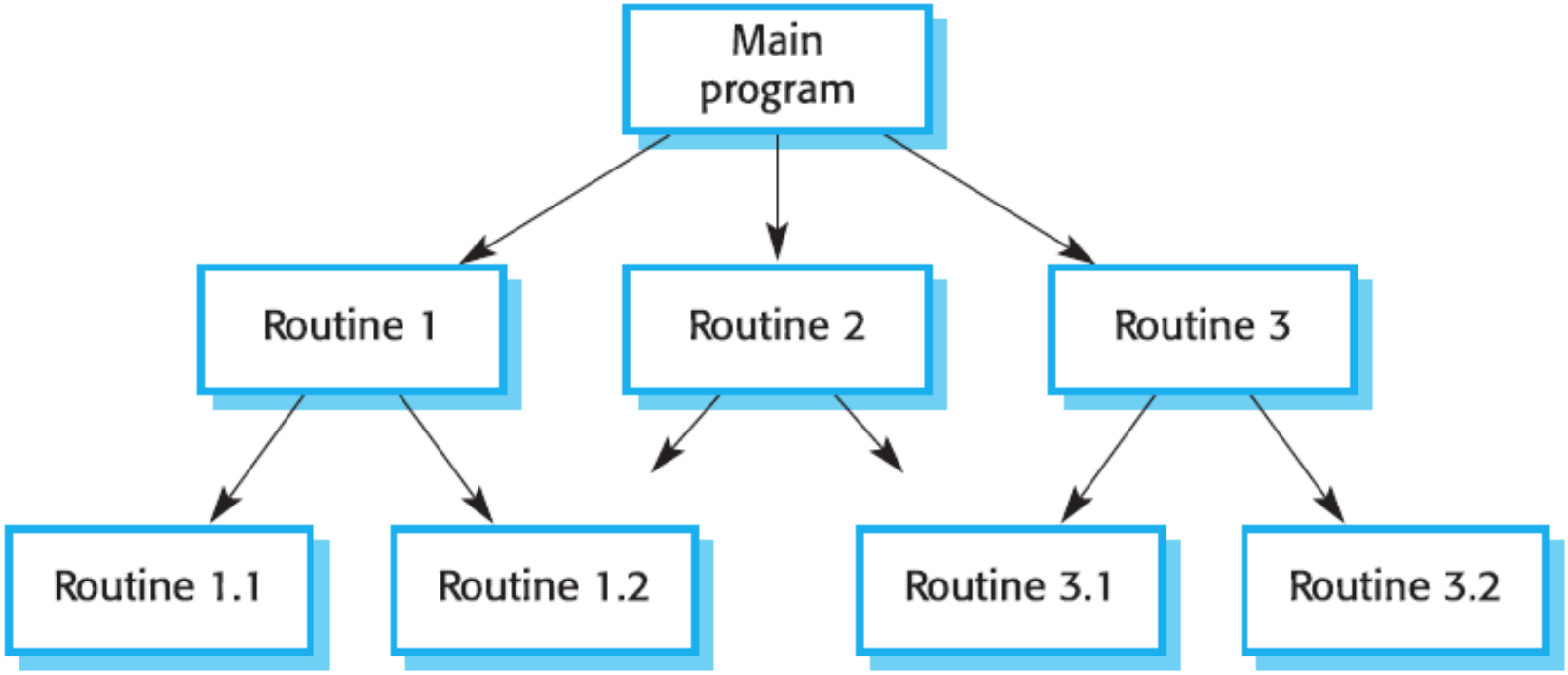
- Konektori:

➢ Relacije između tokova podataka.

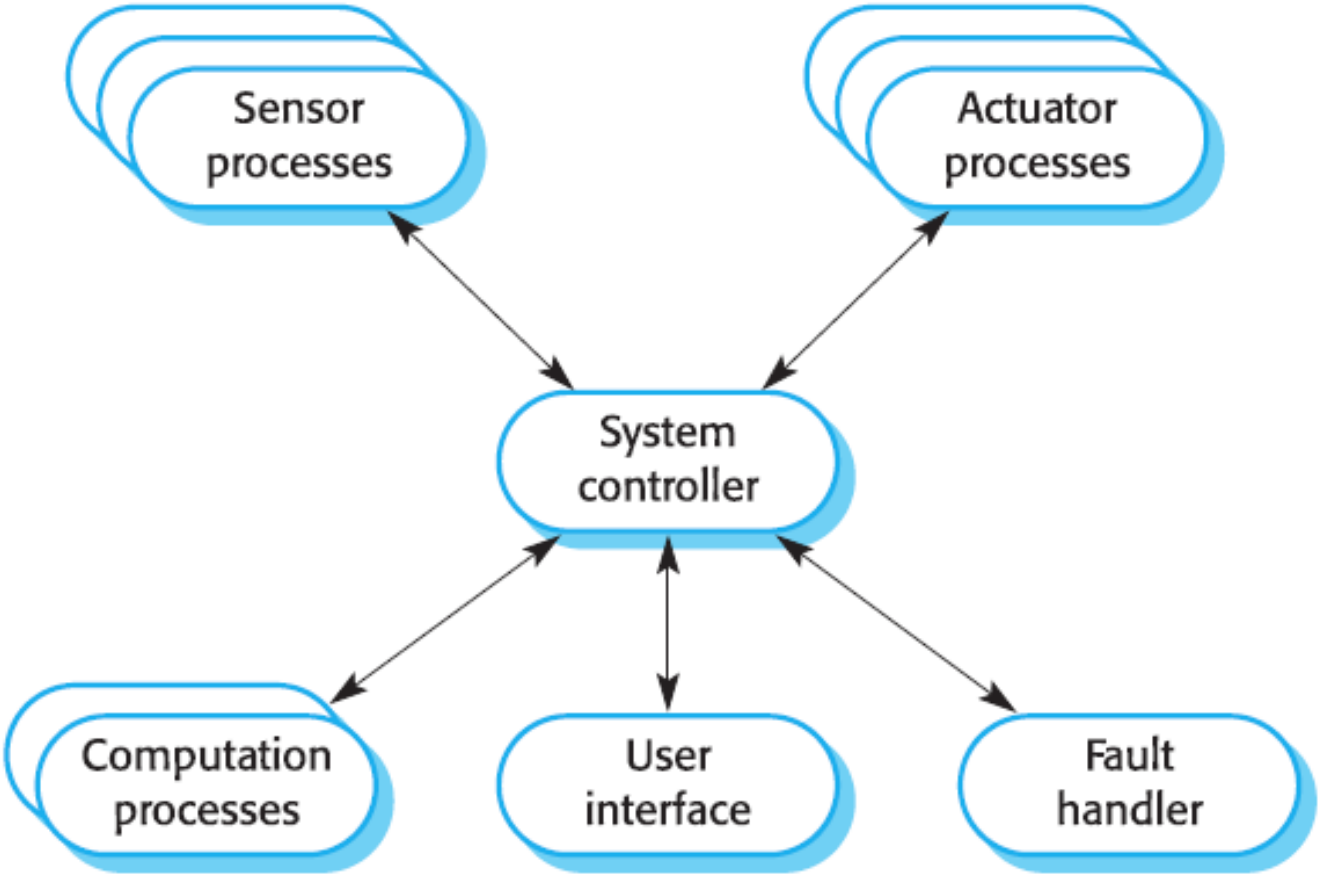
Call-return modeli

- Kod ove grupe modela, kontrola kreće od vršnih podsistema i proteže se naniže (top-down pristup).
- Pogodni su za sekvencijalne sisteme.

Call-return model – primer



Call-return model – primer sistema za rad u realnom vremenu

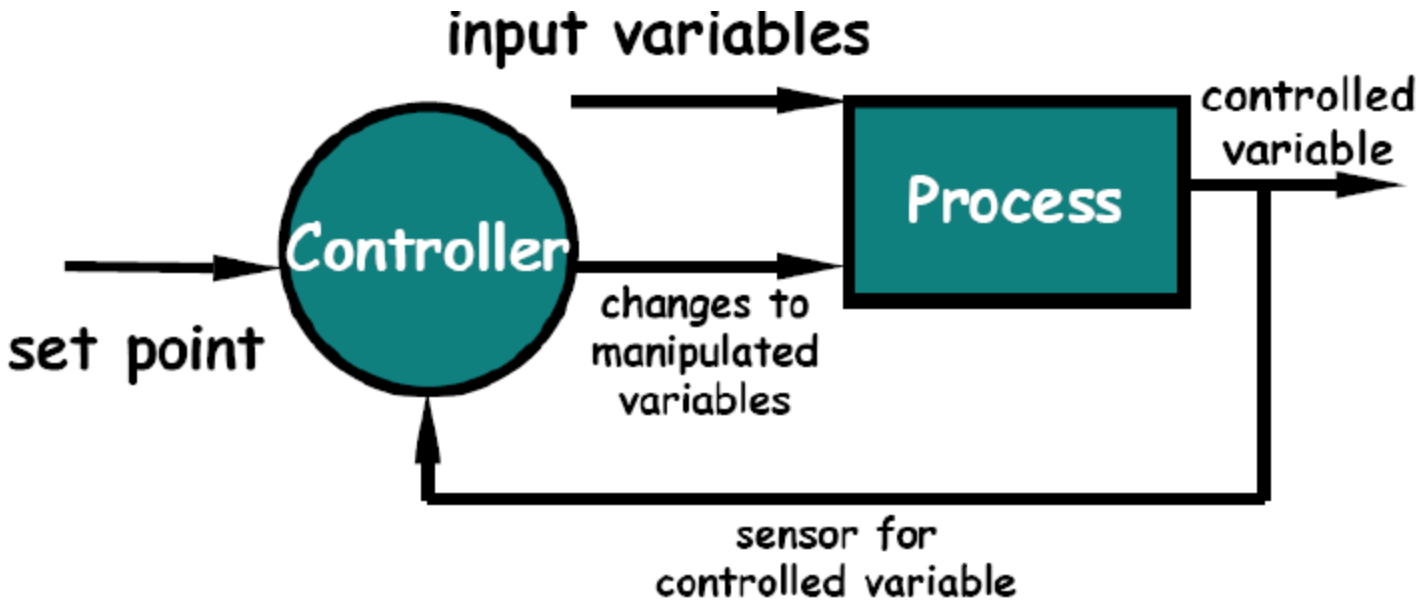


Manager modeli

- Ova grupa modela se primenjuje kod konkurentnih sistema.
- Jedna sistemska komponenta određuje početak, zaustavljanje i koordinaciju rada svih procesa u sistemu.

Feed-back modeli

- Kod ove grupe modela, neke promenljive sistema se kontrolišu i njihove vrednosti se koriste za podešavanje sistema.



Feed-back modeli

- Kod ove grupe modela, neke promenljive sistema se kontrolišu ali se njihove vrednosti ne koriste za podešavanje sistema.