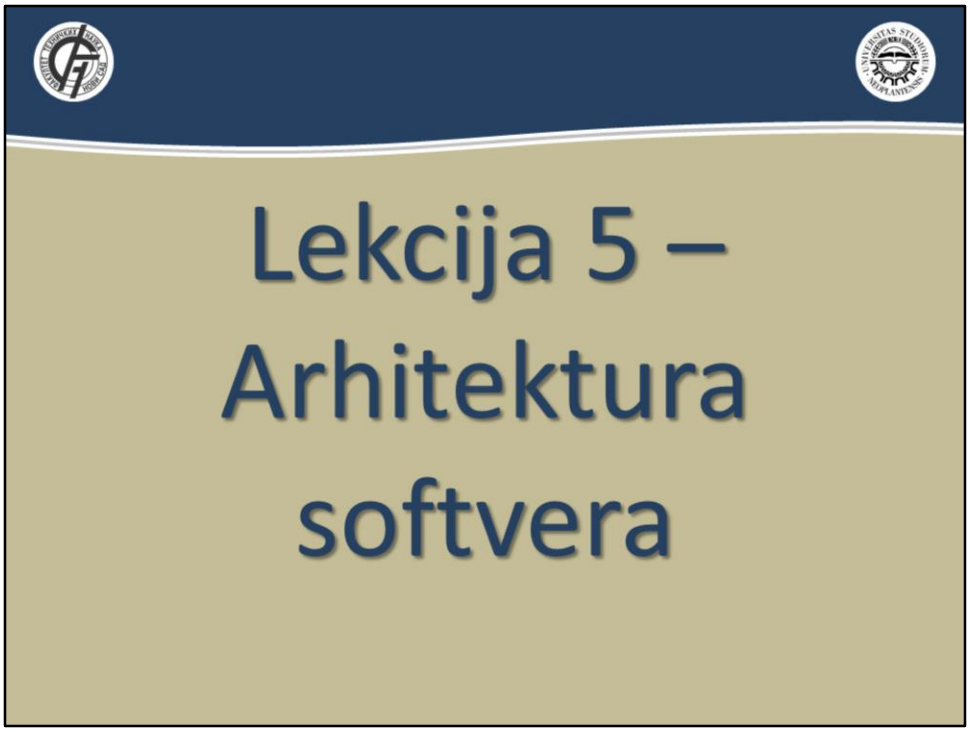
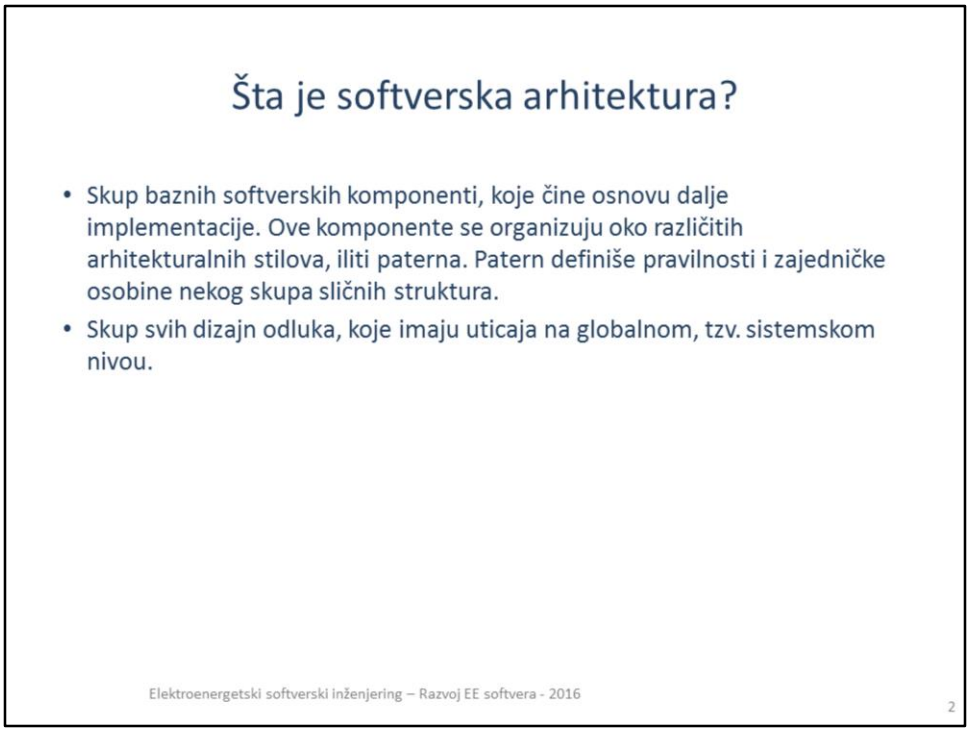

Lekcija 5 –
Arhitektura
softvera



Svaki softverski proizvod ima arhitekturu (može biti bazirana na jednom stilu, ili na kombinaciji više stilova), jedina je razlika u tome da li je ona eksplicitno naglašena, tj da li je nastala planski, ili ad-hoc.

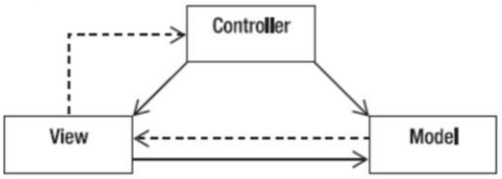
Arhitektura se najčešće predstavlja vizuelno putem UML-a. Čvorovi predstavljaju procesne tačke, dok veze između njih komunikacione puteve (protok podataka, pozivi metoda, itd.).

Osnovni arhitekturalni stilovi – deo 1

- **Pipe-and-filter** – filteri su nezavisni procesni čvorovi

Primer kreiranja svih anagrama na osnovu skupa zadatih reči na ulazu u Unix-u: `sign <recnik.txt | sort | squash >anagrami.txt`

- **Objektno-orijentisani: Model-View-Controller**



Primer MVC-a iz knjige (http://nifty.stanford.edu/2004/RabbitHunt/)

Pipe-and-filter se takođe koristi i u Web aplikacijama, gde niz filtara prvo obradi zahtev (pre procesiranja), zatim se procesira zahtev, da bi na kraju niz filtara obradila izlaz pre slanja ka klijentu. Dalje, MapReduce paradigma za obradu velike količine podataka u distribuiranom režimu prati sličnu logiku. Primer traženja frekvencije pojavljivanja reči u tekstu je tipičan primer za pipe-and-filter, kako putem spajanja Unix komandi, tako i za realizaciju pomoću Hadoop sistema.

Pipe-and-filter nije pogodan za reaktivne sisteme, niti za interaktvne.

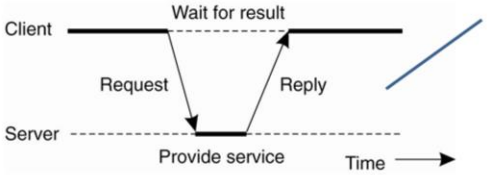
Kod MVC-a je zapravo napravljeno mapiranje: ulaz (Controller) -> procesiranje (Model) -> izlaz (View) Ovaj patern se može proizvoljno skalirati čak i na cele distribuirane sisteme. Tipičan primer su 3-Tier rešenja, gde je View zapravo Web server, Model je baza podataka, dok je Controller aplikativni server u sredini. Naravno, nekada je teško baš povući granice između kontrolera i modela, ili između kontrolera i view-a (prikaza), ali je suština razdvajanje komponenti sistema na dobro enkapsulirane celine (separation of concerns, information hiding, encapsulation). Na ovaj način se mogu ponovno koristiti nezavisno jedno od drugih u različitim kontekstima.

Na primeru Rabbit Hunt-a, vidimo da kontroler kreira grafički interfejs za ceo

program, ali unutar nje odvaja samo prostor za View instancu. Ona dobija parametar od kontrolera gde da se prikazuje. Na taj način, kontroler ne zna detalje prikaza igre, ali kontroliše gde se taj prikaz pojavljuje, kada se osvežava, itd.

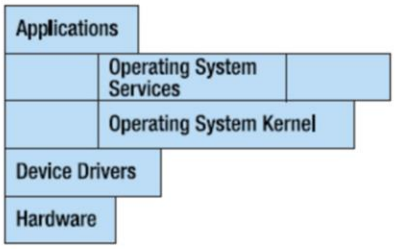
Osnovni arhitekturalni stilovi – deo 2

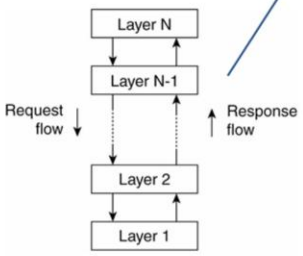
- **Klijent-server** – ovo je više apstrakcija za lakše razumevanje uloga komponenti u nekom sistemu (uglavnom distribuiranim, ali ne mora da bude)



Generalna interakcija između klijenta i servera, odnosno slojevita arh. (slike prezuete iz knjige *Distributed Systems: Principles and Paradigms, 2e* autora Tanenbaum & van Steen

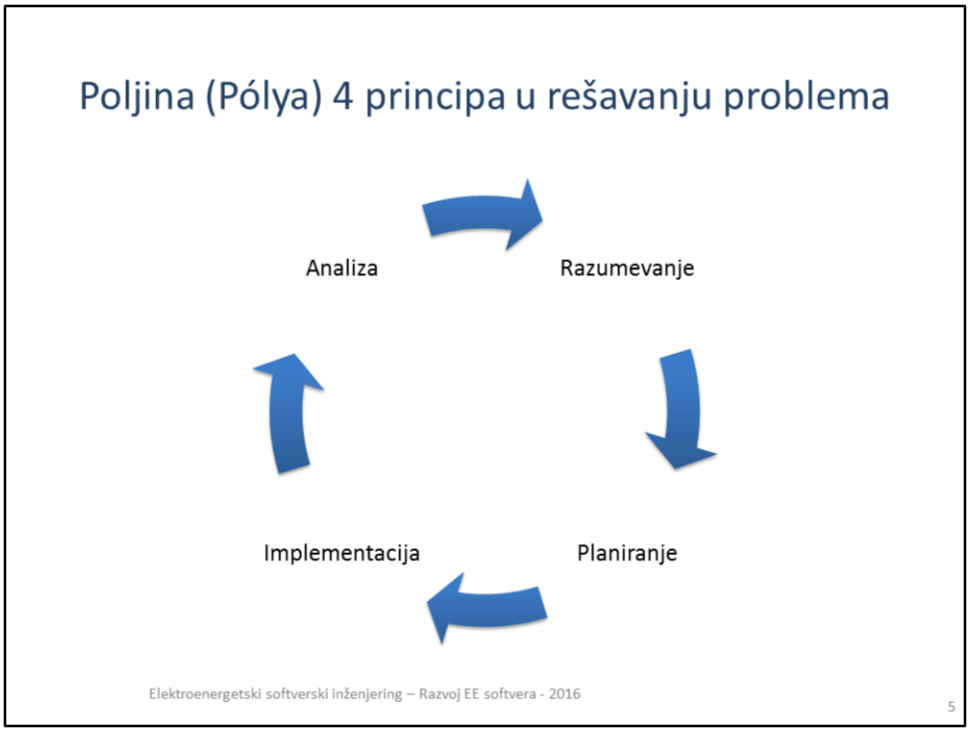
- **Slojevita** – tipični primeri: operativni sistemi i mrežni protokoli





Dobar primer lokalnog klijent-server stila je odnos između aplikacija i sistema za štampanje dokumenata (deo operativnog sistem i često se naziva *print spooler*).

Kod operativnih sistema slojevita izgradnja obezbeđuje da se novi uređaji i njihovi drajveri mogu dodavati bez promene ostalih komponenti sistema. Takođe, aplikacije ne mogu direktno pristupati hardveru bez posredovanja operativnog sistema.



Doduše nije arhitekturalni stil u užem smislu, vredi pomenuti opšti pristup rešavanju problema, onako kako ga je formulisao Pólya György (https://en.wikipedia.org/wiki/How_to_Solve_It). Na ovoj ideji se temelji i N. Wirth-ova tehnika top-down programskog dizajna. Poljin ciklus može biti vezan za incijalni problem ili za neki potproblem.

Kasnije su nastali i razni slični procesi za druge oblasti, na primer, Demingov ciklus (https://en.wikipedia.org/wiki/PDCA).