# Projektiranje informacijskih sustava

SDLC faza dizajna – arhitektura softverskog sustava

Ak. god. 2011/2012

# Arhitektura softverskog sustava

- Što je to arhitektura softverskog sustava?
- The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them.
- Software Architecture in Practice, Second Edition, Len Bass, Paul Clements, Rick Kazman (2003)

# Arhitektura softverskog sustava

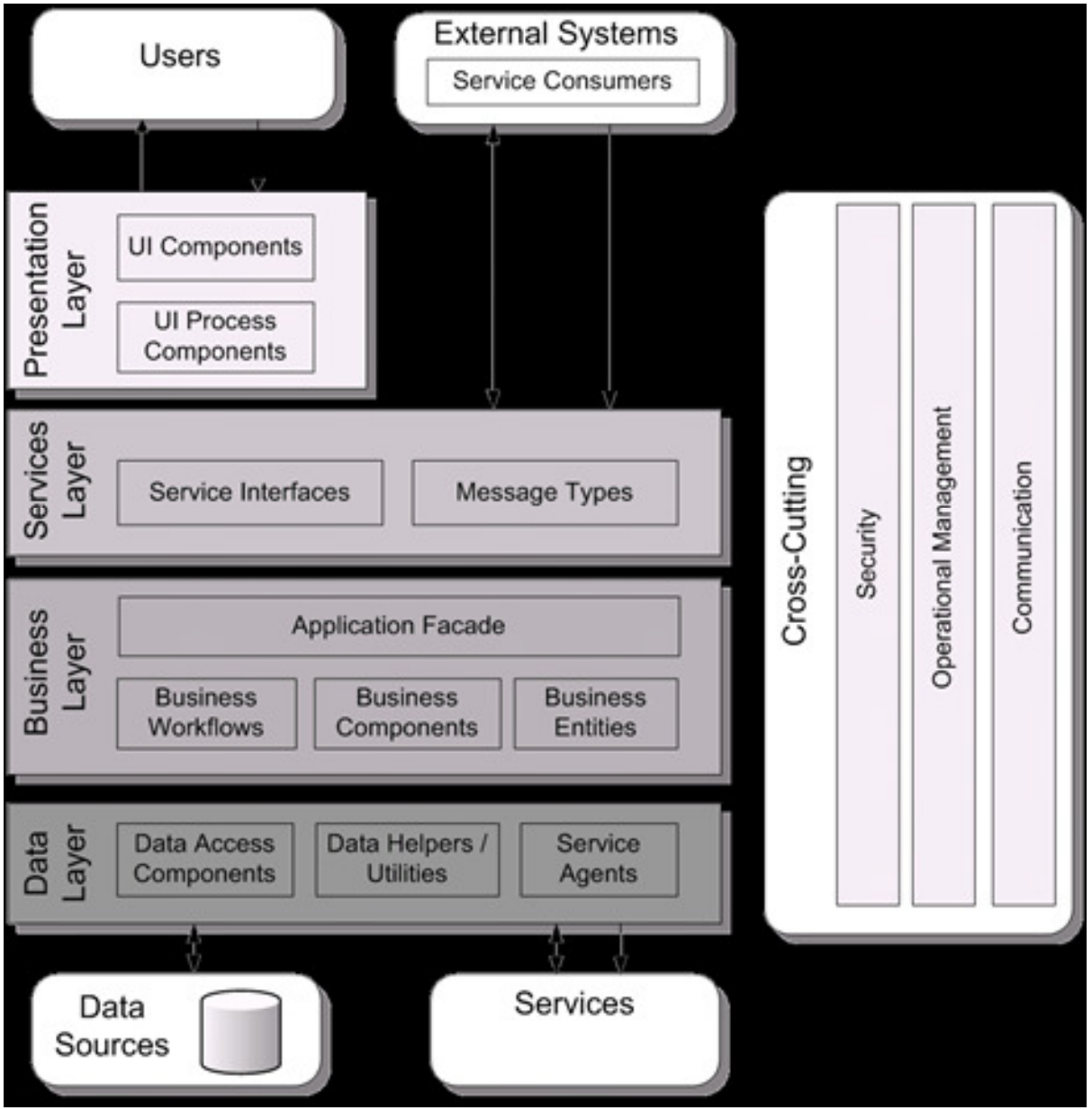
- Arhitektura softverskog sustava je struktura ili strukture sustava, koje sadrže softverske elemente, vanjska vidljiva svojstva tih elemenata i njihove međusobne relacije.
- Softverski element može biti izvršni modul, datoteka koda, objekt …..

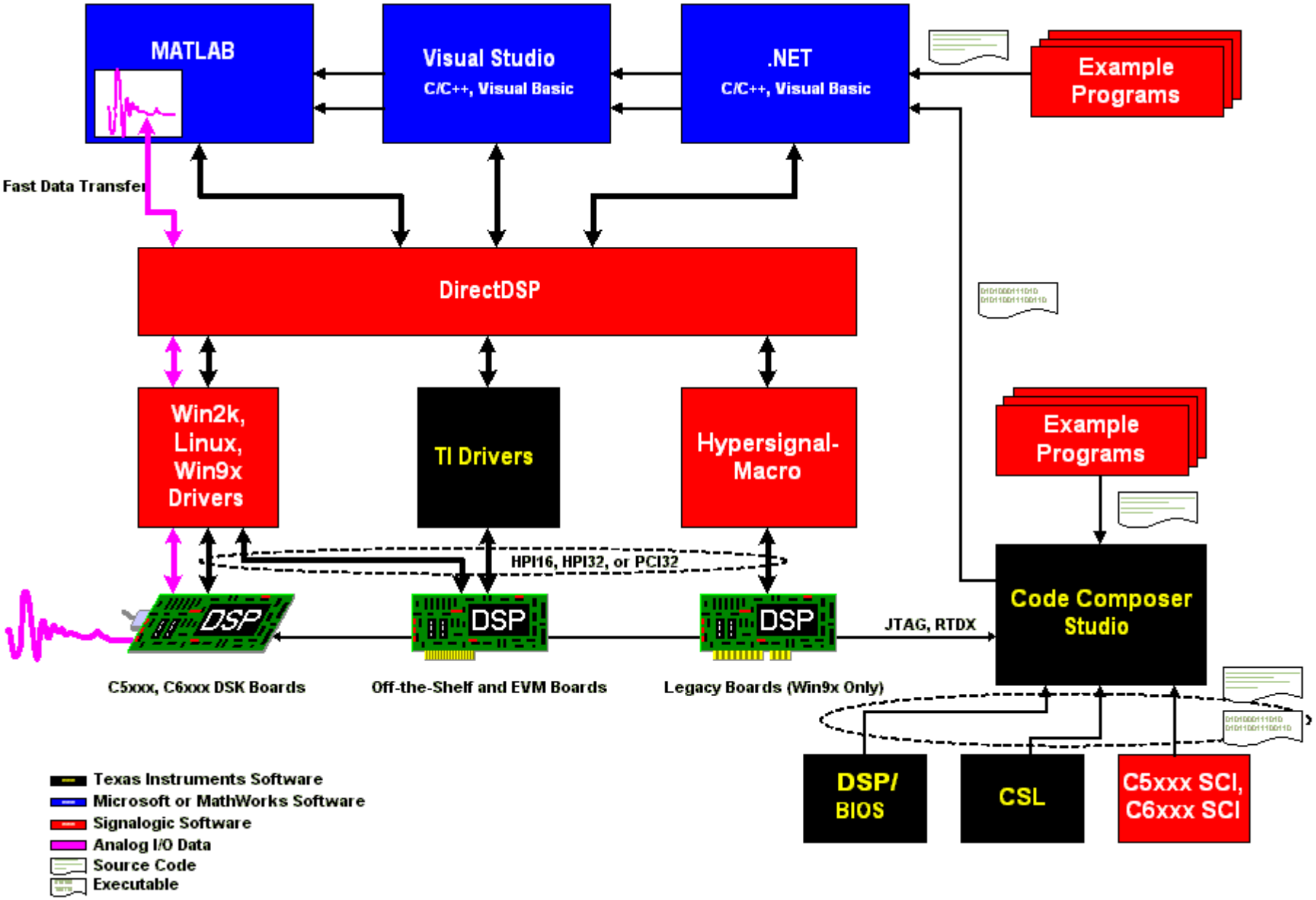
# Arhitektura softverskog sustava

Izvor: www.wikipedia.org

- The software architecture of a system is the set of structures needed to reason about the system, which comprise software elements, relations among them, and properties of both.
- The term also refers to documentation of a system's software architecture.
- Documenting software architecture facilitates communication between stakeholders, documents early decisions about high-level design, and allows reuse of design components and patterns between projects.

Izvor: http://www.techbubbles.com/wp-content/uploads/2009/01/arch.png






Izvor: http://www.signalogic.com/images/DirectDSP_Architecture.gif

# Arhitektura softverskog sustava

- Obično jedna struktura (ili jedan pogled na sustav) nije dovoljan za predstavljanje arhitekture sustava.
- Stoga se arhitektura često predstavlja višeslojno, s obzirom na razinu apstrakcije na kojoj se promatra sustav.
- Na najvišoj razini predstavljanja arhitekture opisuje se cjelokupni sustav obično kroz arhitekturalne predloške (*architecture patterns*) (npr. klijent/server aplikacija, višeslojna aplikacija,....).

# Arhitektura softverskog sustava

- Na nižim razinama se arhitektura definira s obzirom na specifičnu svrhu same aplikacije (npr. bankarska aplikacija sa sigurnosnim mehanizmima na strani klijenta,...)
- Na još nižoj razini je arhitektura pojedinih softverskih komponenti i njihova interakcija. Na ovoj razini se koriste dizajnerskih predlošci, klase, komponente, paketi, ..... (npr. multi-threading aplikacija na strani server koja za svaki klijentski zahtjev otvara novi thread.....)

# Arhitektura softverskog sustava

- Često se koristi pristup da se složena aplikacija dijeli na implementacijske jedinice koje predstavljaju softverske komponente (npr. dll-ov, com-ove, php skripte, itd.).
- Svaka od jedinica ima posebnu funkcionalnost.
- Često se prema pojedinim jedinicama sustava raspodjeljuje posao u programerskom timu. Npr. netko će napraviti PHP skriptu koja sadrži klasu sa upitima prema bazi, netko će napraviti ActiveX kontrolu koja će u web pregledniku korisnika prikazivati neke podatke,...

# Arhitektura softverskog sustava

- Implementacijske jedinice sadrže i programe i podatke koje druge implementacijske jedinice mogu pozvati jer se inače pojedini elementi ne mogu integrirati.
- Ovo je statički prikaz arhitekture fokusiran na to kako je funkcionalnost sustava podijeljena u manje implementacijske jedinice.
- Druge strukture mogu biti fokusirane više na interakciju pojedinih elemenata. Npr. recimo da imamo paralelne procese, tu je jako bitno identificirati sinkronizaciju među pojedinim elementima.

# Arhitektura softverskog sustava

- Vanjska vidljiva svojstva elemenata podrazumijevaju “servise” koje elementi nude drugim elementima (element dohvaća podatke iz baze, element izračunava srednju vrijednost), performanse elemenata, dijeljenje resursa sa drugim elementima i sl.
- Npr. imamo element koji dohvaća podatke iz baze – to on nudi kao “servis” – kao vanjsko vidljivo svojstvo. Performansa bi bila npr. da mu za te podatke treba 10 milisekundi. Ako ubacimo element koji traži taj servis od promatranog elementa, ali ga treba dobiti za manje od 10 ms očigledno nam ovakva arhitektura ne odgovara.

# Arhitektura softverskog sustava

- Arhitektura sustava obavezno uključuje relacije između softverskih elemenata. Npr. element grafičkog sučelja koji prikazuje podatke iz baze se oslanja na “servis” elementa koji kupi podatke iz baze.
- Informacije o elementima koje se ne odnose na njihovu interakciju nisu bitne za arhitekturu sustava. Npr. element koji kupi podatke iz baze ima više threadova izvršavanja.

# Arhitektura softverskog sustava

- Arhitektura sustava je model sustava tj. apstrakcija sustava koja izostavlja detalje o svojstvima koji ne utječu na ponašanje elemenata prema dugim elementima u sustavu i njihovu međusobnu interakciju.
- Često je interakcija među elementima definirana sučeljem elementa koje odvaja informacije bitne za interakciju sa elementom (javne) od informacija koje nisu vezane uz interakciju elemenata (privatne).

# Arhitektura softverskog sustava

- Pojam arhitekture softverskog sustava prvi su definirala Edgser Dijkstra 1968. i David Parnas početkom 70-tih. Oni su identificirali važnost strukture softverskog sustava.
- Naime u početku razvoja softvera kreće sa programskim jezici niske razine. Razvoj aplikacije u programskim jezicim niske razine je vrlo eksplicitan, nema skrivenih razine i funkcionalnosti koje se javljaju u programskim jezicima više razine koji vode do sve složenijih aplikacija.

# Arhitektura softverskog sustava

- Početkom 90-tih istraživanja se usmjeravaju na arhitekturalne predloške (ili stilove), jezike za opis arhitekture (*ADL - architecture description languages*), načine dokumentiranja arhitekture i sl.
- U razvoj područja značajnu ulogu imaju instituti i sveučilišta (Carnegie Mellon, Irvin Institut na Sveučilištu Kalifornija, …)
- Izvor: www.wikipedia.org

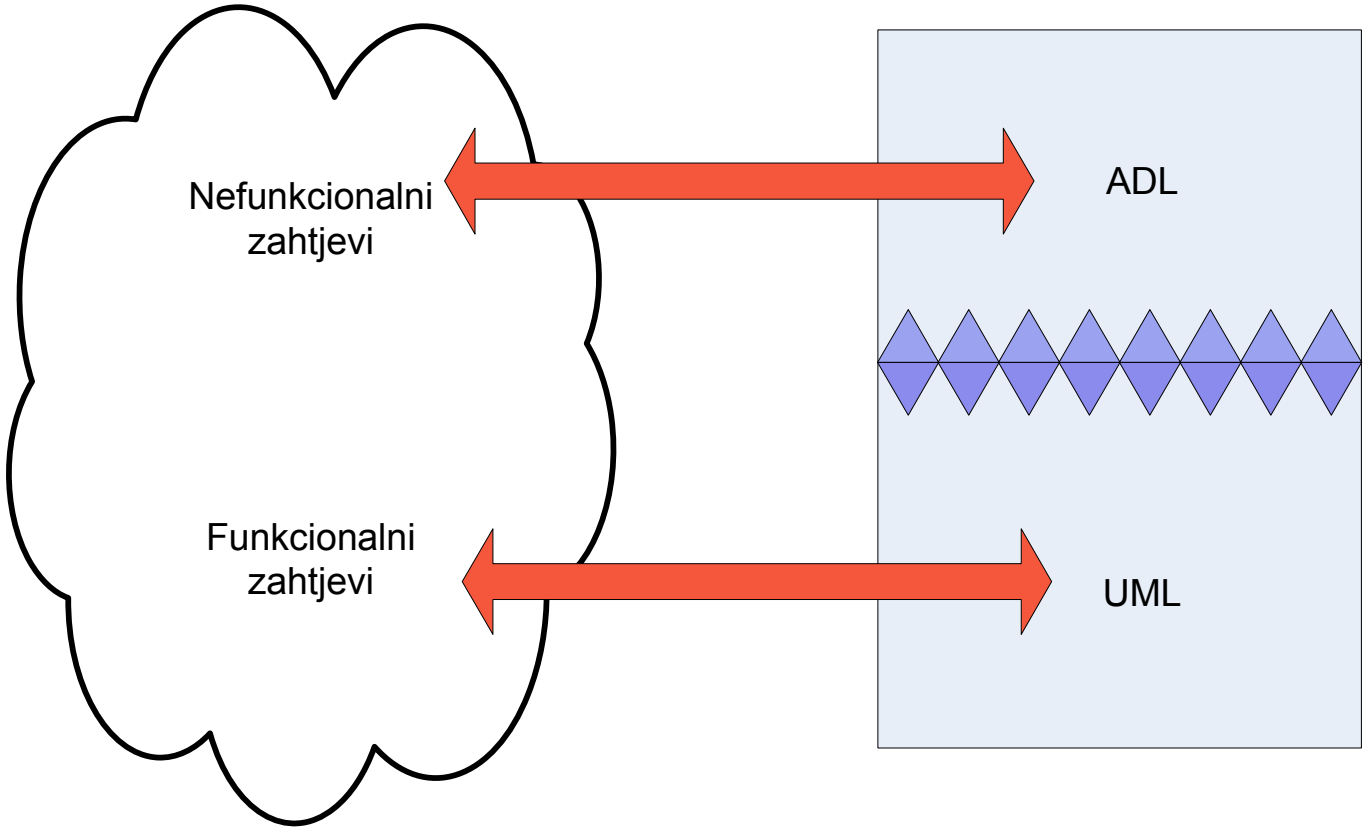
# Jezici za opis arhitekture

- ADL jezici pružaju formalni način za predstavljanje arhitekture.
- Mogu podržavati automatsko generiranje koda.
- Omogućavaju analizu arhitekture kroz analizu performansi, uporabljivosti (*usability*) sa aspekta korisnika, sigurnosti i sl.
- Namijenjeni su interpretaciji i od čovjeka i od računala.
- Nažalost ne postoji suglasnost što ADL jezici trebaju predstaviti u sustavu, posebno što se tiče ponašanja.
- ADL jezici - ACME (CMU/USC), Rapide (Stanford), Wright (CMU), Unicon (CMU),…

# ADL ↔ UML

- ADL jezici mogu biti i grafički.
- U čemu je razlika između ADL jezika i UML?
- UML je primarno namijenjen opisu funkcionalnosti aplikacije, dok je arhitektura sustava usmjerena na to kako ćemo tu funkcionalnost realizirati.
- Ali ove granice nisu tako oštre. ADL i UML jedan drugome ulaze u područje.

# ADL ↔ UML



# UML

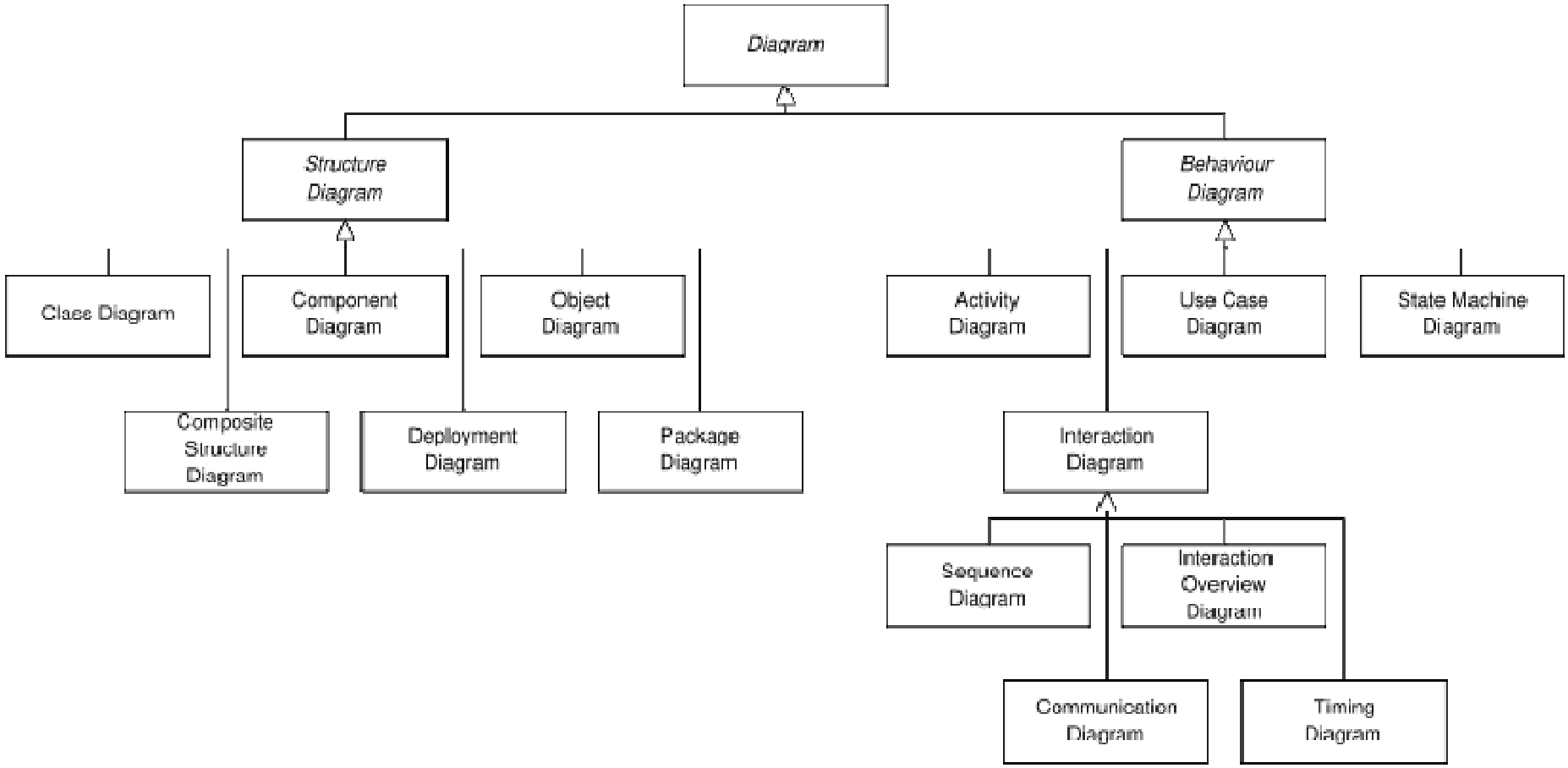
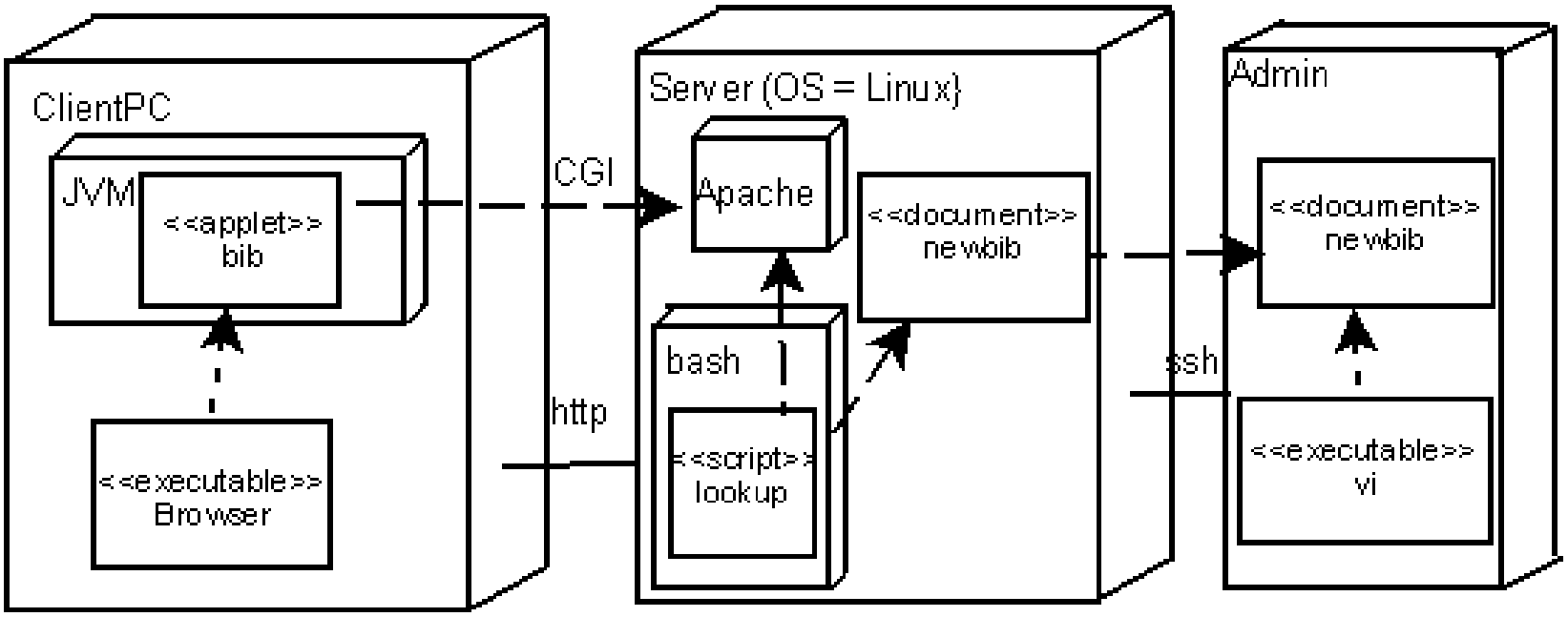
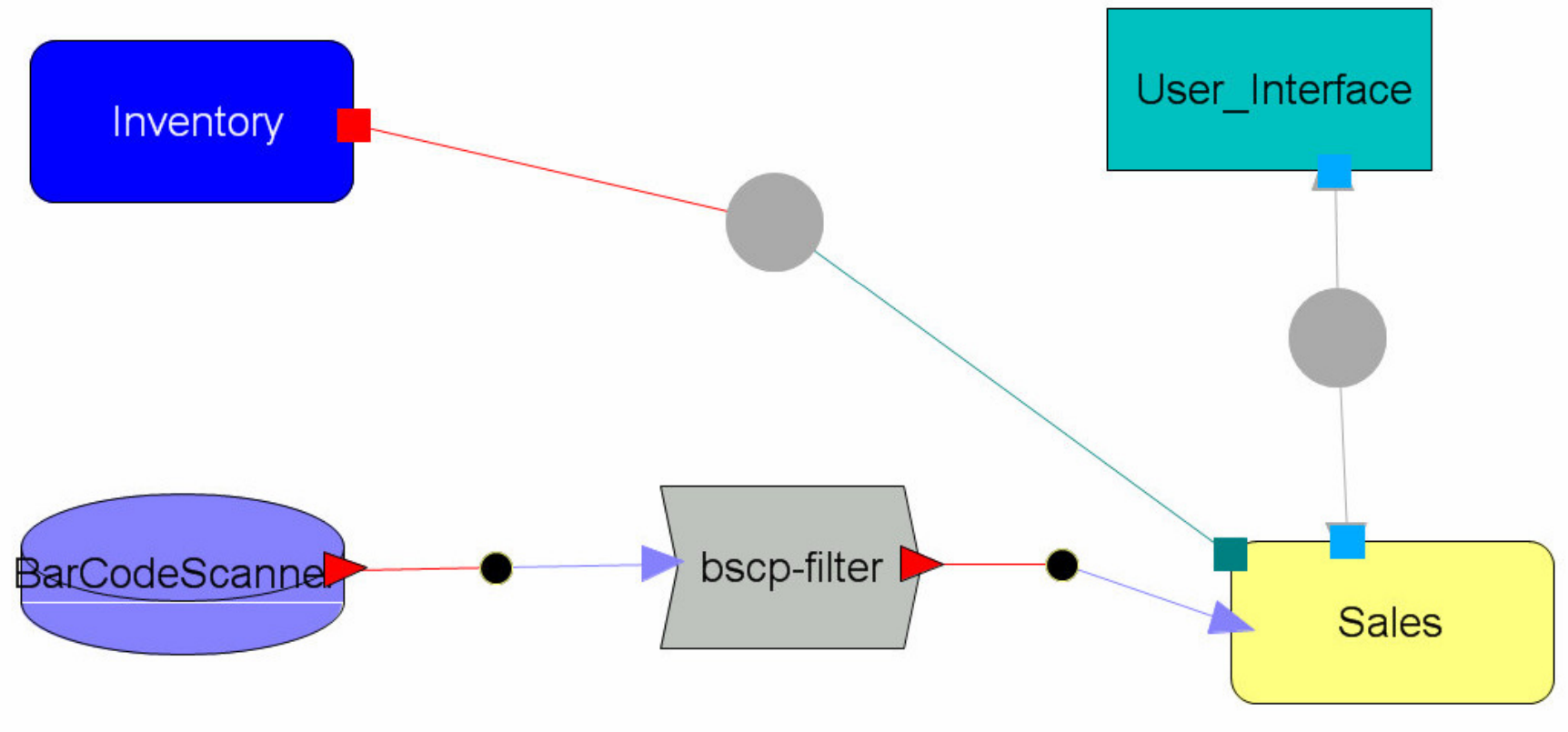


Diagram
Structure Diagram
Behaviour Diagram
Class Diagram
Component Diagram
Object Diagram
Activity Diagram
Use Case Diagram
State Machine Diagram
Composite Structure Diagram
Deployment Diagram
Package Diagram
Interaction Diagram
Sequence Diagram
Interaction Overview Diagram
Communication Diagram
Timing Diagram

# Dijagram isporuke



# Jezici za opis arhitekture – primjer ACME



# Jezici za opis arhitekture - primjer ACME

```
import $AS_GLOBAL_PATH\families\ClientAndServerFam.acme;
import $AS_GLOBAL_PATH\families\PipesAndFiltersFam.acme;
System pos : ClientAndServerFam, PipesAndFiltersFam =  new ClientAndServerFam, PipesAndFiltersFam extended with {
    Component bscp-filter : Filter =  new Filter extended with {
        Property flowPaths : Set{flowpathRecT} = {[
            fromPt : string = "input";
            toPt : string = "output" ]};
    };

    Connector raw : Pipe =  new Pipe extended with {
        Property flowPaths : Set{flowpathRecT} = {[
            fromPt : string = "source";
            toPt : string = "sink" ]};
        Property bufferSize : int = 0;
    };

    Component BarCodeScanner : DataSource =  new DataSource;

    Attachment BarCodeScanner.output to raw.sink;
    Attachment bscp-filter.input to raw.source;
    Component Sales : ClientT, DataSink =  new ClientT, DataSink extended with {
        Port p0;

    };

    Connector bscp : Pipe =  new Pipe extended with {
        Property flowPaths : Set{flowpathRecT} = {[
            fromPt : string = "source";
            toPt : string = "sink" ]};
        Property bufferSize : int = 0;
    };

    Attachment bscp-filter.output to bscp.sink;
    Component User_Interface = {
        Port p0;
```

# IEEE 1471

- IEEE 1471 je skraćeno ime za standard sa oznakom ANSI/IEEE 1471-2000, *Recommended Practice for Architecture Description of Software-Intensive Systems.*
- To je IEEE standard za predstavljanje arhitekture softverskog sustava.
- IEEE je postavio normative za konceptualni okvir za opis arhitekture softverskog sustava.

# IEEE 1471

- Opis arhitekture sustava se koristi za:
  - Predstavljanje sustava i njegovu procjenu
  - Komunikaciju među zainteresiranima za sustav
  - Procjenu i usporedbu arhitektura
  - Planiranje i upravljanje razvojem sustava
  - Predstavljanje trajnih karakteristika i principa sustava za olakšavanje mogućih promjena sustava (npr. ako je to web aplikacija ne možemo unijeti element koji se odnosi na desktop aplikaciju)
  - Verifikaciju implementacije sustava sa postavljenom arhitekturom (npr. ako je identificiran element koji nudi određeni servis drugim elementima, a on nije realiziran kao poseban element)

# IEEE 1471

- Korištena terminologija:
  - *architect* - osoba, tim ili organizacija zadužena za dizajniranje arhitekture sustava.
  - *architectural description* (AD) - kolekcija produkata koji dokumentiraju arhitekturu.
  - *architecture* - temeljna organizacija sustava koja sjedinjuje pojedine komponente, njihove međusobne relacije i relacije prema okolinu sustava, te principe dizajna i razvoja sustava.
  - *designing* - aktivnosti definiranja, dokumentiranja, održavanja, unapređivanja i provjere određene implementacije definirane arhitekture.
  - *system* - kolekcija komponenti organizirana sa određenom funkcionalnom svrhom.
  - *system stakeholder* - osoba, tim ili organizacija zainteresirane za sustav.
  - *view* - predstavljanje sustava iz određene perspektive.
  - *viewpoint* - konvencije za kreiranje određenog prikaza (*view*).

# IEEE 1471

- Neki pojmovi i postavke konceptualnog okvira IEEE 1471 standarda su:
  - Postavljeni meta-modeli za opis arhitekture
  - Arhitektura je definirana za određenu instancu softverskog sustava (nema generičkih arhitektura)
  - Opis arhitekture se obavezno prezentirani kroz više pogleda, tj. samo jedan pogleda na sustav ne može opisati arhitekturu sustava u potpunosti
  - Određeni pogled na sustav identificira određena svojstva arhitekture i definira se različitim tehnikama modeliranja i reprezentacije koje mogu prikazati upravo ona svojstva na koje se taj pogled na sustav odnosi.

# IEEE 1471

- Arhitektura se prema IEEE 1471 organizira kroz različite prikaze, analogne nacrtima u građevinarstvu (vodovodne instalacije, statika zgrade, …). Definirani prikazi su:
  - Functional/logic view
  - Code/module view
  - Development/structural view
  - Concurrency/process/thread view
  - Physical/deployment view
  - User action/feedback view
  - Data view

# Arhitektura softverskog sustava

- Arhitektura razvijanog sustava se može oslanjati na neki od definiranih predložaka, kombinaciju predložaka ili koristiti vlastiti pristup. Često korišteni predlošci su:
  - Klijent-server arhitektura
  - Višeslojna arhitektura (npr. troslojna arhitektura - prezentacijska logika, poslovna logika, podatkovna logika)
  - Objektno-orijentirana arhitektura
  - Servisno orijentirana arhitektura
  - Front-end/back-end arhitektura (Joomla)
  - Monolitna aplikacija…..