

# **SOFTWARE REUSE**

VIŠESTRUKA UPOTREBA SOFTVERA

# KOREKTAN PREVOD?

- *Reupotrebljiv* softver? ( ne postoji prefiks RE u srpskom jeziku )
- *Ponovo upotrebljiv* softver? ( totalno bezveze )
- Upotrebljiv više puta?
- Itd.

# PLAN RADA 😊

1. Počnimo sa primerom!
2. Kako i zašto?
3. „Pejzaž“ višestruke upotrebe softvera
4. Frameworks
5. Proizvodne linije softvera
6. COTS sistemi


# 1/6 PRIMER

- Nekada davno bili smo mladi i naivni.
- I hteli smo da nađemo maksimum niza.
- I napisali smo program.



# 1/6 PRIMER

```
static int findMax(int[] numbers){  
  
    int maximum = numbers[0];  
  
    for (int i = 1; i < numbers.Length; i++)  
    {  
        maximum = numbers[i] > maximum ? numbers[i] : maximum;  
    }  
  
    return maximum;  
}
```



# 1/6 PRIMER

ŠTA NIJE DOBRO?  
SVAŠTA.

# 1/6 PRIMER FANTASTIČAN

```
static T findMax<T>(IEnumerable<T> stuff) where T : IComparable {
```

```
    T maximum = stuff.ElementAt(0);
```

```
    for (int i = 1; i < stuff.Count(); i++)
```

```
    {
```

```
        maximum = maximum.CompareTo(stuff.ElementAt(i)) < 0 ? stuff.ElementAt(i) : maximum;
```

```
    }
```

```
    return maximum;
```

```
}
```



## 2/6 KAKO I ZAŠTO?

### Kako?


- Upotreba celih sistema
- Upotreba komponenti
- Upotreba objekata i funkcija





## 2/6 KAKO I ZAŠTO?

### Zašto?

- Zato što morate
  - Isprobani softver je pouzdan softver
  - Znamo koliko će da nas košta
  - Poštovanje standarda
  - Brži razvoj
  - Specijalisti
- 

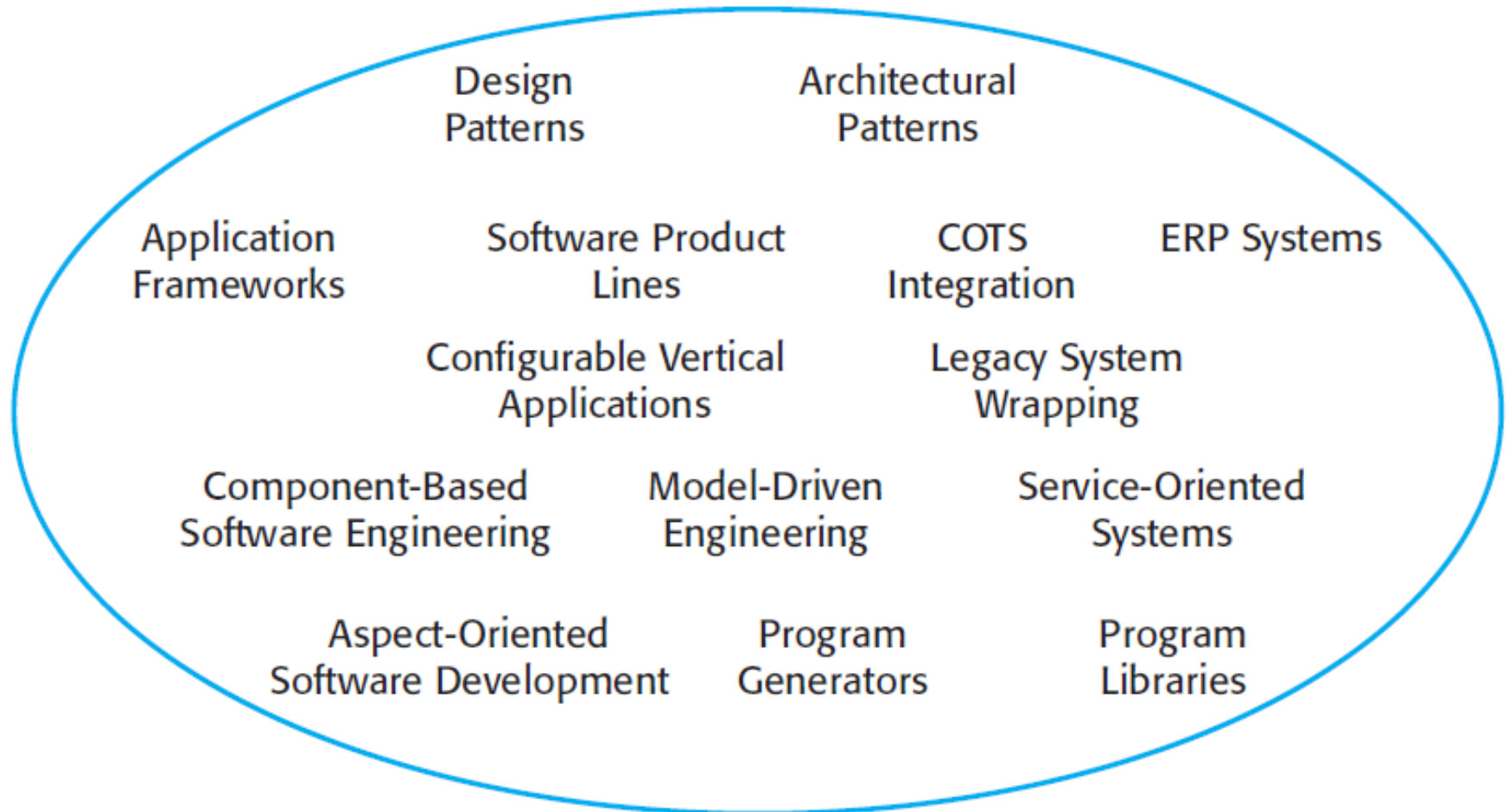
## 2/6 KAKO I ZAŠTO?

### Zašto ne?

- Troškovi korišćenja tuđeg softvera
- Nekompatibilnost alata
- Ja to mogu bolje! sindrom



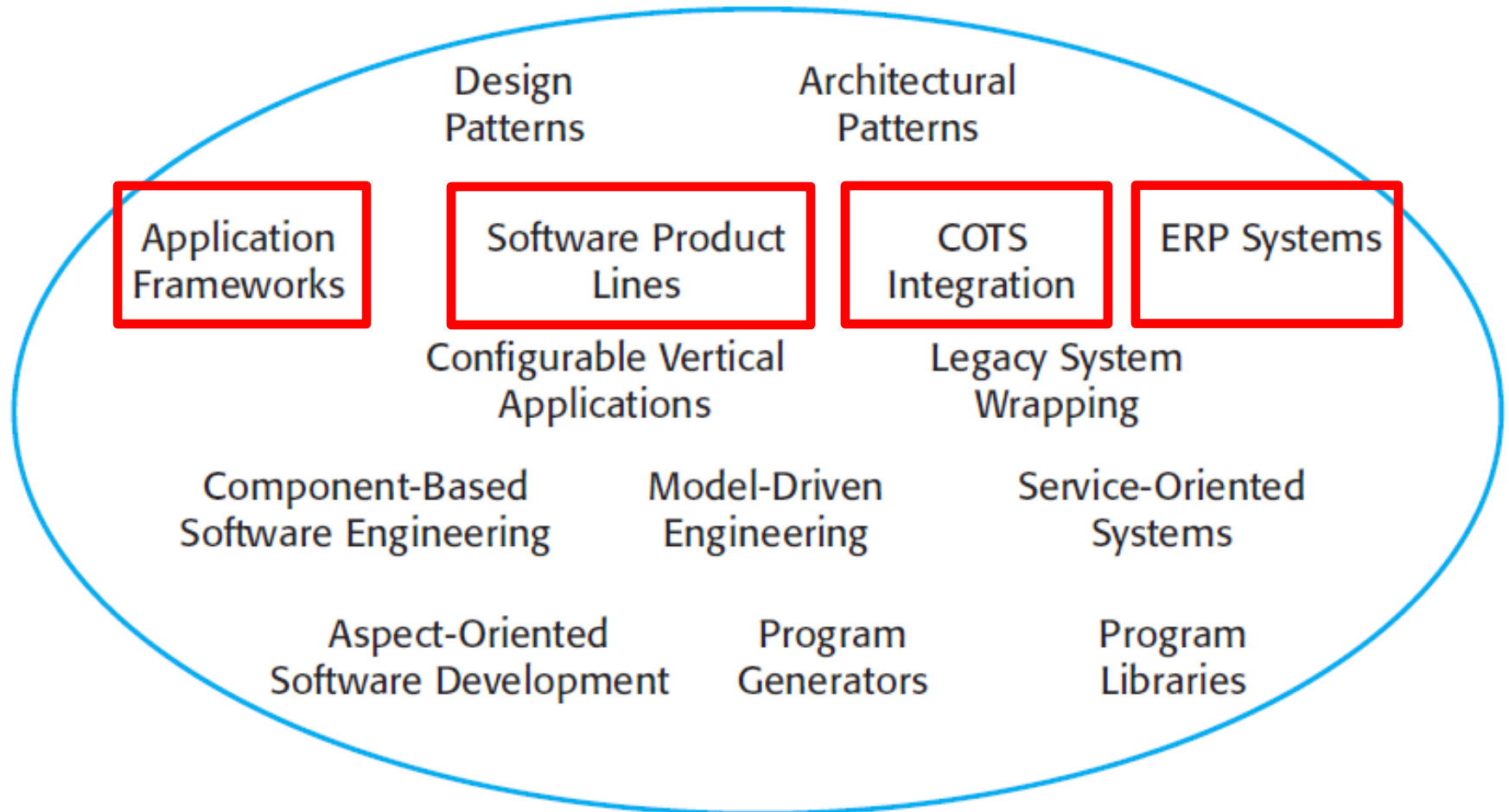
# 3/6 „PEJZAŽ“ VIŠESTRUKÉ UPOTREBE SOFTVERA




# 3/6 „PEJZAŽ“ VIŠESTRUKA UPOTREBE SOFTVERA

- Neke stvari su nam već poznate jeeej!
  - Design patterns ( uzorci, šabloni, obrasci za projektovanje )
  - Architectural patterns ( uf... )
  - Frameworks? ( okviri?)
  - ...
- Za neke nikad nismo čuli ( barem ja ) ☹
  - Software product lines ( proizvodne linije )
  - ERP
  - COTS
  - ...

# 3/6 „PEJZAŽ“ VIŠESTRUKÉ UPOTREBE SOFTVERA



# ŠTA OD OVOGA KORISTITI?

- Ako su nam rokovi kratki onda COTS i slične gotove sisteme.
  - Ako je planirani životni vek našeg softvera dug, onda navedeno iznad treba izbegavati.
  - Ono što znamo da koristimo.
  - Ako je upotreba kritična onda je bolje ne koristiti softver za koji nemamo uvid u izvorni kod.
  - Ako za domen aplikacije već postoji neko generičko rešenje onda koristimo to rešenje.
  - Koristimo ono što platforma na kojoj razvijamo podržava.
- 

## 4/6 FRAMEWORKS

*“An integrated set of software artefacts (such as classes, objects and components) that collaborate to provide a reusable architecture for a family of related applications.” – Schmidt 2004*

- Svi smo ih koristili ali konkretna definicija toga šta je tačno framework ( okvir ) je pomalo problematična i nekad je teško odvojiti sam okvir od svih ostalih komponenti koje idu sa njim u „paketu“.



## 4/6 FRAMEWORKS

- Framework je skup konkretnih i apstraktnih klasa objekata u jednom objektno orijentisanom jeziku. Odatle sledi da je konkretan framework vezan za konkretan programski jezik.
- **Kako onda .NET framework podrzava više različitih jezika?**
- **Trik pitanje**, ne podržava, to su različite implementacije jednog framework-a. Svaki jezik koji se pokreće na .NET CLR-u mora da implementira sve klase koje sadrži .NET framework.





## 4/6 FRAMEWORKS

- EJB, Qt, ...
- Web frameworks:
  - .NET, spring, hibernate, django, codeigniter, ...
  - Često se oblikuju oko nekih uzoraka za projektovanje.
- Prednosti, **mane?**



## 5/6 PROIZVODNE LINIJE SOFTVERA

- **Proizvodne linije softvera ( software product lines)** su skupovi aplikacija sa zajedničkom arhitekturom koje mogu da dele neke komponente, gde je svaka aplikacija takva da odgovara drugačijim zahtevima.
- Sistem se prilagođava potrebama klijenta.
- Najčešće ovakvi generički skupovi aplikacija nastaju vremenom, unutar organizacija koje se bave proizvodnjom aplikacija u istom domenu.
- Odnos između PLS i Framework-a?



# 5/6 PROIZVODNE LINIJE SOFTVERA

- Kako specijalizujemo jedan ovakav sistem?
  - Za konkretnu platformu
  - Za okruženje
  - U odnosu na funkciju
  - U odnosu na procese
- **Magični redosled**: Arhitektura, instanca, specijalizovana instanca.

# 5/6 PRIMER RM ARCHITEKTURE

## Interaction

User Interface

## I/O Management

User  
Authentication

Resource  
Delivery

Query  
Management

## Resource Management

Resource  
Tracking

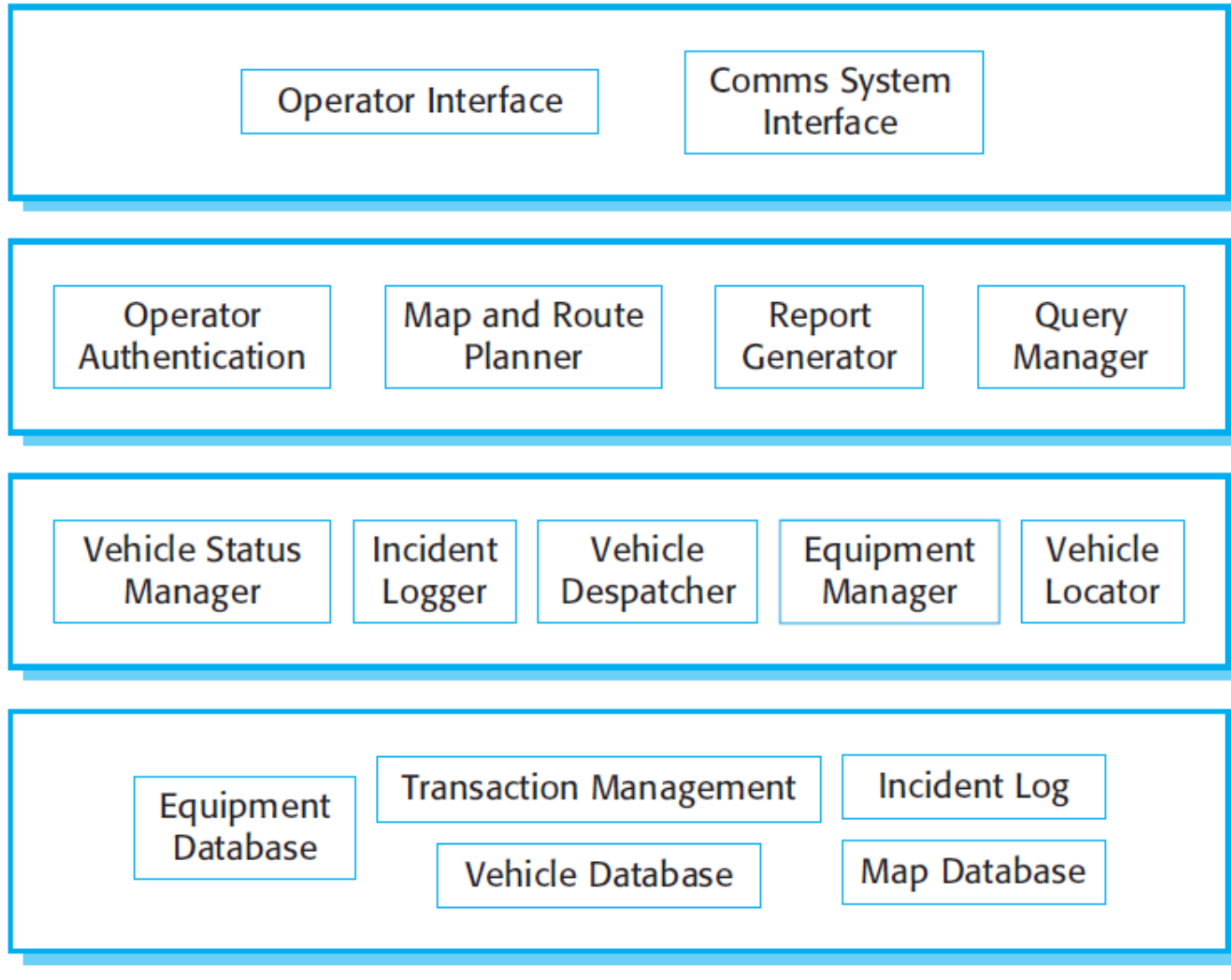
Resource Policy  
Control

Resource  
Allocation

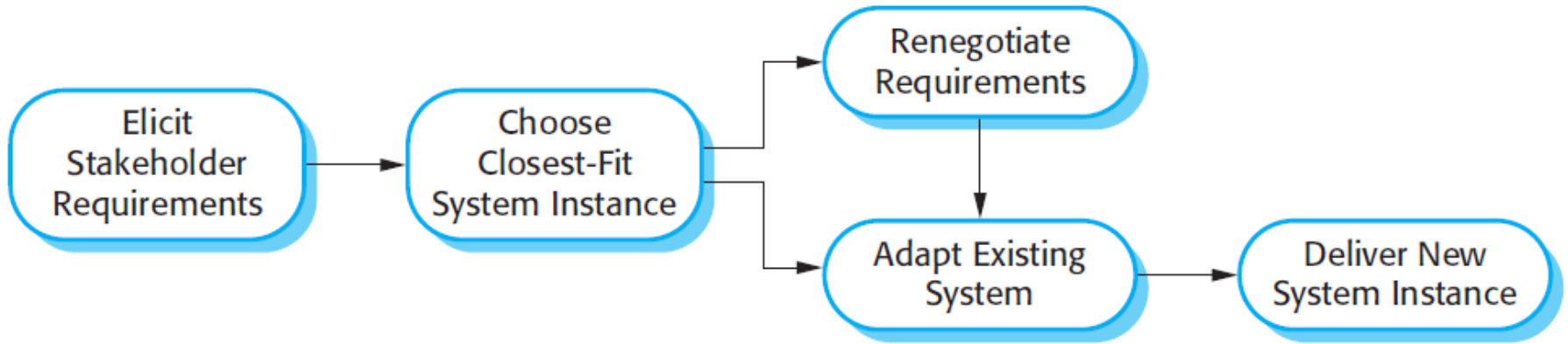
## Database Management

Transaction Management  
Resource Database

# 5/6 INSTANCA RM ARHITEKTURE



## 5/6 KAKO SPECIJALIZUJEMO?



- Prednosti i mane?

## 6/6 COTS

- **COTS** (commercial-off-the-shelf ) ne može se bukvalno prevesti a da ima smisla.
- U mom „slobodnom“ prevodu **Komercijalna gotova rešenja** ( nema nikakve veze sa knjigom „Gotova rešenja“ koja se bavi uzorcima za projektovanje ).
- To su sistemi koji se mogu prilagoditi potrebama različitih klijenata **bez menjanja izvornog koda**.
- Možemo reći da je većina standardnog softvera koji se danas koristi na neki način COTS.

# 6/6 COTS

Postoje 2 tipa:

- **COTS rešenja**
  - Generička aplikacija od jednog proizvođača koja se konfiguriše prema potrebama klijenta.
- **COTS integrisani sistemi**
  - Dva ili više COTS sistema koji se integrišu u jedan IS.

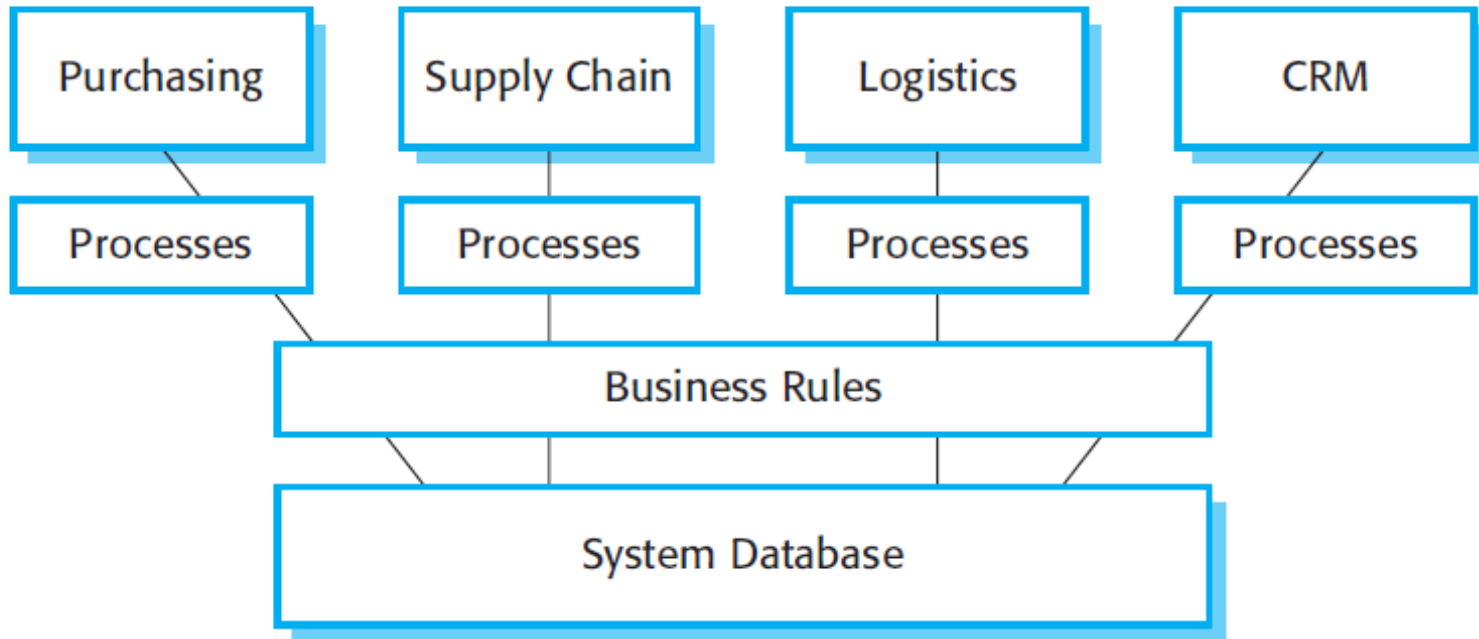




## 6/6 COTS REŠENJA

- Mogu da budu usmerena ka specifičnim poslovima, poslovnim funkcijama ali i ka celokupnim poslovnim sistemima.
- ERP ( Enterprise Resource Planning ) kao što su SAP i BEA ( Oracle ) proizvodi.
- Sadrže veći broj modula koji se integrišu i konfigurišu po potrebi.
- Proces konfigurisanja je najjobimniji deo posla.
- ***Testiranje je veliki problem.***

## 6/6 COTS REŠENJA




# 6/6 COTS INTEGRISANA REŠENJA

- Dva ili više COTS sistema a nekad i legacy sistem.
- Kad se koristi?
- **Tri pitanja:**
  - Koji od ponuđenih sistema nam je najpodesniji?
  - Kako će se vršiti razmena podataka?
  - Koje mogućnosti sistema ćemo koristiti?



# 6/6 COTS INTEGRISANA REŠENJA

## PROBLEMI:

- Nedostatak kontrole nad funkcionalnošću i performansama.
  - Problemi u interoperabilnosti.
  - Nedostatak kontrole nad evolucijom sistema.
  - Podrška korisniku od strane proizvođača.
- 

# NAPOKON ZAKLJUČAK

- Višestruka upotreba softvera je super, sem kad ne valja.
- Postoji više načina ovakve upotrebe, od „recikliranja“ pojedinih komponenti do upotrebe celih sistema.
- Problema ima puno ali benefita još više.
- Treba biti svestan postojanja već gotovih rešenja.

HVALA NA PAŽNJI! 😊