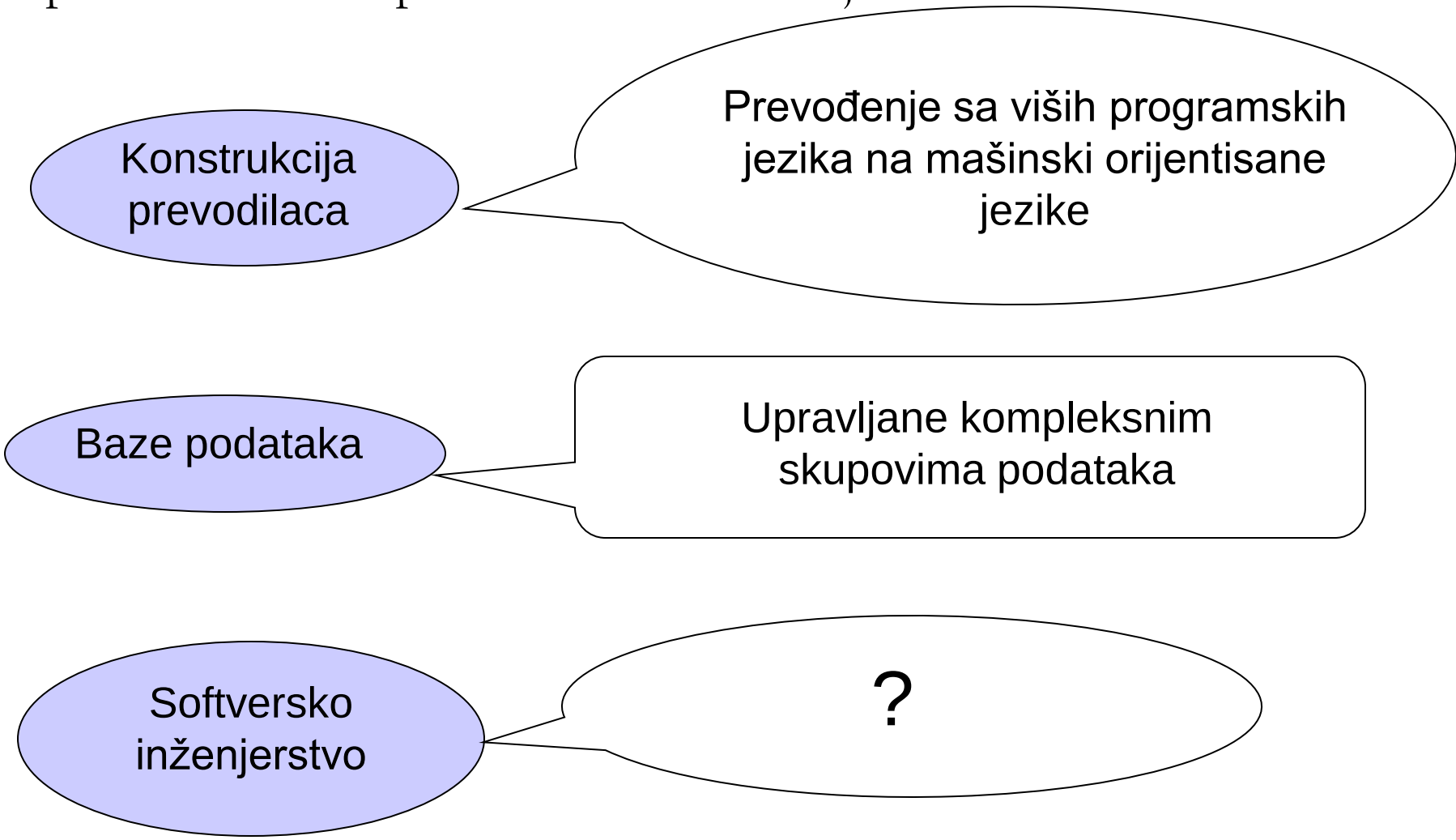
Rešavanje problema pomoću računara

Vladimir Filipović

vladaf@matf.bg.ac.rs

Softversko inženjerstvo

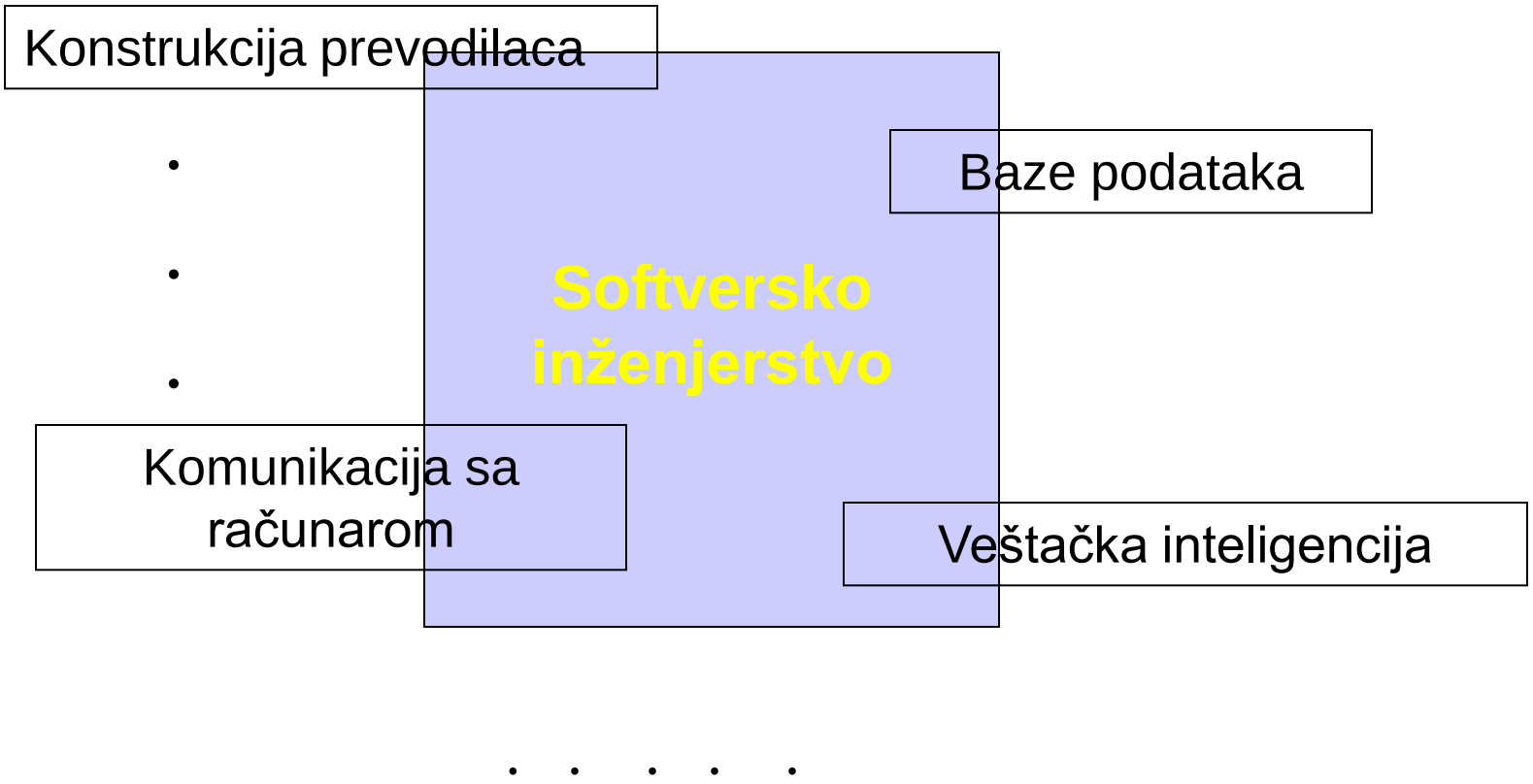
Šta podrazumevamo pod softverskim inženjerstvom?



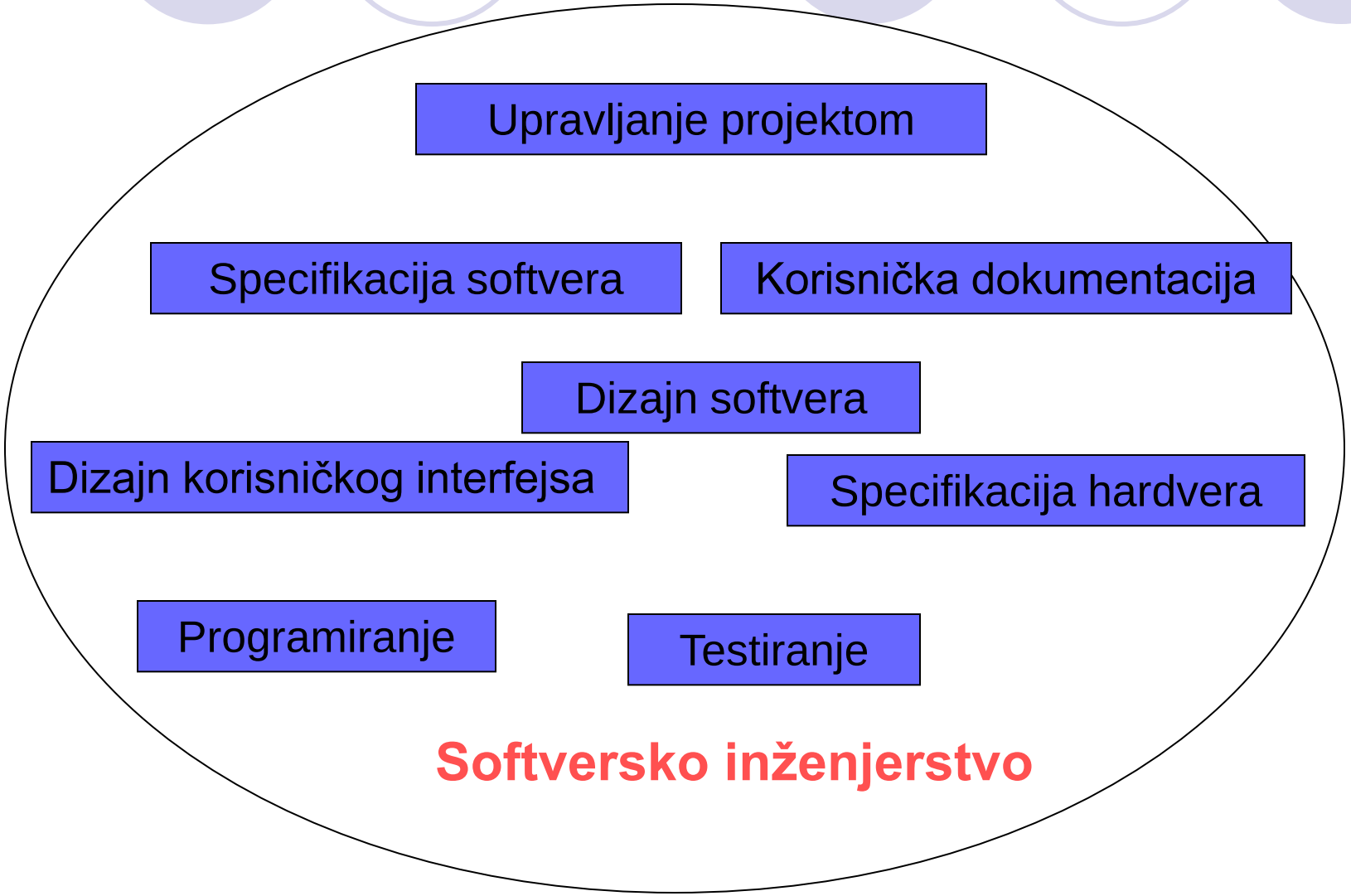
Neke definicije:

1. Softversko inženjerstvo (SI) je disciplina koja ima za cilj ekonomičan razvoj visoko-kvalitetnog softvera (Pagel, 1994.)
2. SI je inženjerska disciplina povezana sa praktičnim problemima razvoja velikih softverskih sistema. (Sommerville, 1992.)
3. SI je definisan, korak po korak proces koji olakšava specifikaciju, dizajn, implementaciju i testiranje softverskog rešenja za skup postavljenih zahteva na najekspeditivniji i najjeftiniji način (Kehoe, 1995.)

Softversko inženjerstvo nije jedna specijalna oblast računarstva, već kombinacija raznih računarskih oblasti.

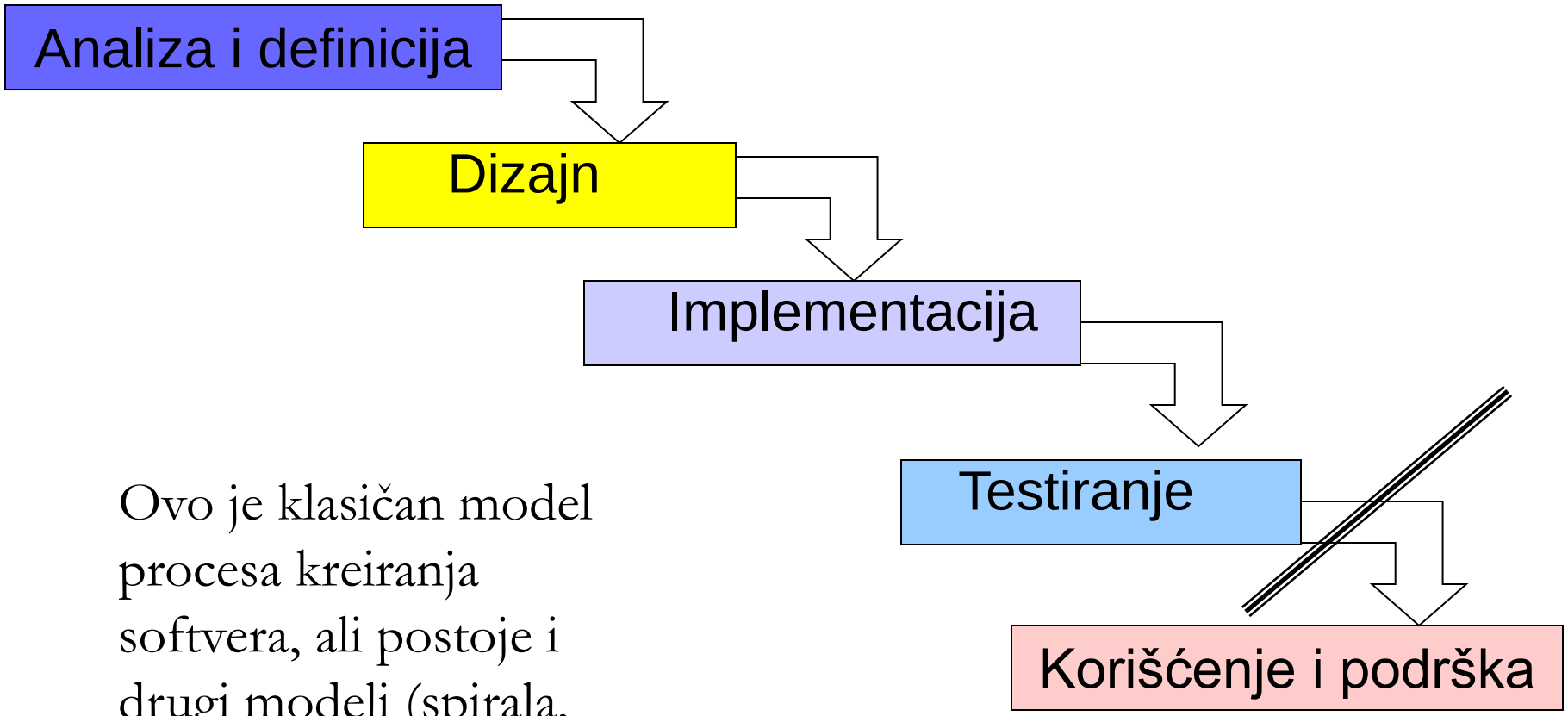


Koje oblasti pripadaju softverskom inženjerstvu?



Procesi (aktivnosti) u kreiranju softvera

Jedan od modela za opis glavnih aktivnosti u SI je tzv. "Vodopadni model".



Ovo je klasičan model procesa kreiranja softvera, ali postoje i drugi modeli (spirala, kodiraj-pa-popravi, evolutivna isporuka

Analiza i definicija softvera

Ciljevi faze analize i definicije su:

- Analiza problema koji treba da bude rešen
- Definicija zahteva nad softverskim proizvodom (šta sve treba da zadovoljava softver koji treba da se kreira).

U fazi analize razlikuju se dve podfaze:

Faza planiranja (studija izvodljivosti: rečnik (glosarijum), preliminarna specifikacija zahteva, procena cene, projektni plan)

Faza definisanja (definicija proizvoda: specifikacija zahteva, model proizvoda, korisnički interfejs, uputstvo korisniku)

Rezultat faze analize i definicija treba da bude nekoliko dokumenata, kao što su: struktura projekta, osnova ugovora, osnova dizajna, …

Zahtevi

Zahtevi predstavljaju najvažnije elemente faze analiza. Šta podrazumevamo pod zahtevima? Zahtevi predstavljaju verbalni opis onoga šta proizvod treba da zadovoljava.

Možemo razlikovati:

1. Funkcionalne (operacione) zahteve (funkcionalnost, podaci tj. logičko viđenje), korisnički interfejs)
2. Zahtevi aplikacionog okruženja (aplikaciona situacija, korisnički profil)
3. Tehnički zahtevi (jezik implementacije, operativni sistem, hardver)
4. Performansni zahtevi (efikasnost, obim podataka)
5. Zahtevi validnosti (pripremanje testova, specijalni slučajevi za testiranje)

6. Zahtevi kvaliteta (prijateljski orijentisan softver, pouzdan,…)
7. Zahtevi realizacije (model procesa, dokumentacija, regulisanje krajnjih rokova, cena)

Pojedini zahtevi zavisiće i od oblasti primene softvera (to su specifični zahtevi vezani za oblast primene).

Na osnovu **preporuka IEEE**, treba navesti i neke opšte opise, kao što su:

- Perspektiva proizvoda
- Funkcije proizvoda
- Karakteristike korisnika
- Ograničenja
- Pretpostavke i zavisnosti

Objektno-orijentisana analiza

Glavni aspekti OO analize su:

- Model-paradigma (dekompozicija softverskog sistema u objekte)
- Jezik modeliranja (UML - Unified Modeling Language)
- Metodologija (Kako razviti model?)
- Alati (Together, Rational Rose, …)

Za realizaciju OO analize ključni pojmovi su objekat i klasa.

Kao rezultat OO analize treba da se dobije pseudo-kod u kojem su opisani: glavni zadatak klase, atributi i glavne operacije.

Treba opisati i kako se klase grupišu u pakete.

Dizajn softvera

Dizajn softvera je drugi proces u "Vodopadnom modelu"

Za fazu dizajna važna su sledeća pitanja:

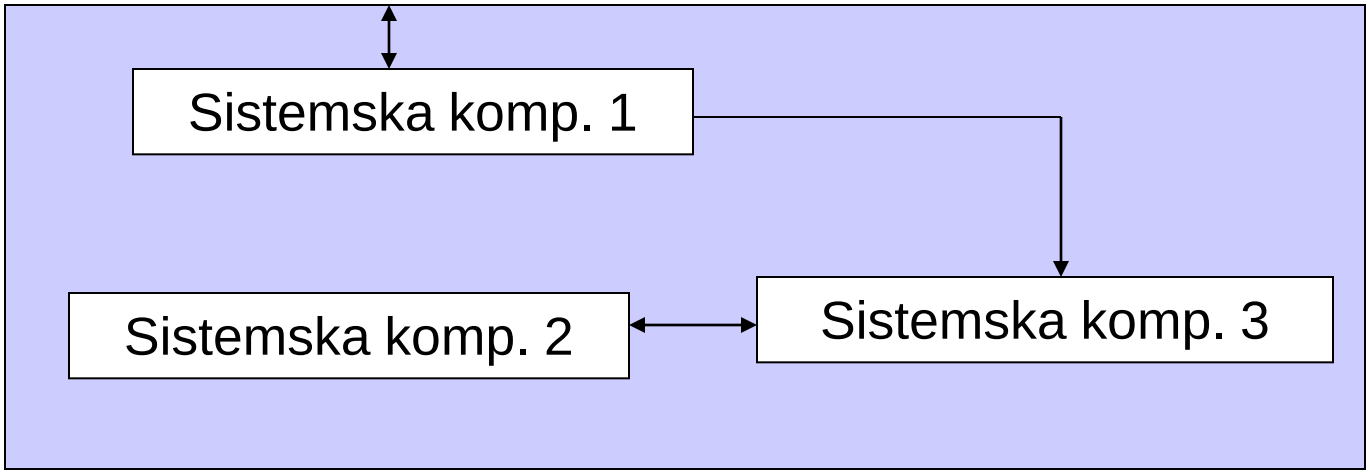
- Dokumenti dizajn faze
- Kriterijumi kvaliteta
- Metodi dizajna
- Uticajni faktori na softversku arhitekturu
- Slojevita softver arhitektura
- Klijent/server softverska arhitektura
- Web softverska arhitektura

U okviru softver-dizajna, možemo razlikovati:

Dizajn arhitekture i

Dizajn implementacije

Dizajn arhitekture sastoji se od navođenja sistemskih komponenti i relacija među njima. (Kao komponente arhitekture pojavljuju se: atomi, funkcije, moduli, klase, podsistemi, …)



Za dobre softverske arhitekture karakteristična je slaba povezanost i jaka kohezija (svi elementi komponente služe rešavanju zajedničkog zadatka).

Interfejs (između komponenti) treba da je sto jednostavniji. Na taj način su komponente lakše za razumevanje i modifikaciju.

Kako glavni metodi dizajna pojavljuju se: funkcionalna apstrakcija i apstrakcija podataka.

Najbitniji uticajni faktori na arhitekturu softvera su: definicija proizvoda i implementacija jezika.

Slojem nazivamo podsistem čije komponente logički treba da budu zajedno. Kod kompleksnih sistema, normalno je da se proizvod sastoji iz više slojeva. Slojevi mogu biti uređeni na razne načine - bitno je da su sve komponente unutar sloja jednako dostupne.

Objektno-orijentisani dizajn

U OO-dizajnu razlikujemo:

1. Fazu arhitekture

    Ova faza se sastoji od OO-dizajna modela.

    Mogu se razlikovati sledeći modeli:

        proširen

        modifikovan

        optimizovan

        prilagođen okolini

2. Faza implementacije sastoji se od modifikacija koje treba napraviti prilikom prilagođavanja ciljnom jeziku.

Implementacija i testiranje softvera

Implementacija se realizuje korišćenjem (najčešće) jednog programskog jezika. Međutim, ima i kompleksnih implementacija gde se može koristiti veći broj jezika. Kasnije se prevedeni moduli povezuju međusobno.

Testiranje softvera - značajna faza u kojoj se ispituje da li softver ispunjava postavljene zahteve.

Tester softvera - posebno zanimanje.

Dijkstra: "Testiranjem se ne može dokazati da je program ispravan, već se samo može dokazati da u programu ima grešaka".

Glavni postupak prilikom testiranja je izbor podataka za koje je unapred poznat rezultat i provera da li se pomoću softvera zaista dobija taj rezultat. Problem nastaje kada je teško naći takve podatke.

Društveni aspekti softverskog inženjerstva

- Odgovornost stručnjaka
- Rizici i pouzdanost
- Intelektualna svojina
- Internet - softver - društvo
- Veštačka inteligencija i čovek.