***KLASE SOFTVERSKIH SISTEMA***

1. KAKO SE **ODNOSE PREMA OKRUŽENJU**
2. PREMA **INTERNIM KARAKTERISTIKAMA** (kako upravljaju kontrolom toka,izvršavaju akcije od kojih se sastoji obrada)
3. **GENERIČKE OBLASTI PRIMENE** (osobine tih softvera i razvoj softvera za neke primene)
4. ODNOSI PREMA OKRUŽENJU
5. **paketni** (sistemi sa paketnom obradom)

* imamo odredjeni skup ulaznih podataka koje predamo sistemu na obradu i na izlazu on da rezultate
* tako su funkcionisali i prvi računari
* danas se koristi jer je prilično zgodan za automatizaciju
* takođe veoma ih je pogodno ulančavati, gde je svaki zadužen za jedan korak u celokupnoj obradi (UNIX – jednostavni programi čijim kombinovanjem možemo graditi komplikovane sisteme)

1. **interaktivni**

* omogućavaju da korisnik ili nešto drugo iz okruženja utiče na to kako će se dalje odvijati ono što softver radi (klik na dugme, unos u neko polje dobijamo rezultate)
* prednosti : krajnjim korisnicima je bolji jer mogu da menjaju podatke po potrebi da bi dobili traženi rezultat
* mane:izazovi pri implementaciji (osmišljavanje komunikacije sa korisnicima, kako ona utice na trenutni rad sistema,..)

1. **real-time** 
   * nije dovoljno da dobijemo tacan rezultat, vec i da taj rezultat dobijemo na vreme
   * nekada zahteva zrtvovanje kvalitet podatka da bi bili sigurni da ih dobijamo na vreme
2. **Embedded sistemi**
   * oni gde je softver ili hardver deo nekog drugog proizvoda i u odnosu na taj proizvod obično ima neki visok stepen autonomije
   * često smo ograničeni po pitanju hardvera(brzine procesora, rasoploživosti memorije (ograničen broj upisa u neku lokaciju), potrošne energije (napajanje solarnim panelima, veoma je bitno kada će system raditi da ne bi ostali bez energije)
   * često su on i sistemi u realnom vremenu
3. **Distribuirani sistemi**
   * čiji se delovi mogu izvršavati na vise računara istovremeno, pri čemu ti delovi medjusobno saradjuju, ti racunari moraju biti povezani mrežom da bi razmenjivali podatke
   * mogu biti takvi da imaju
     1. **centralizovano upravljanje** - jedan je zaduzen dali zadatke drugim delovima - **čvrto spregnuti sistemi**
     2. **decentralizovani** - svaki od delova ima neku svoju lokalnu autonomiju po pitanju isporučivanja obrade - **slabo spregnuti sistemi**

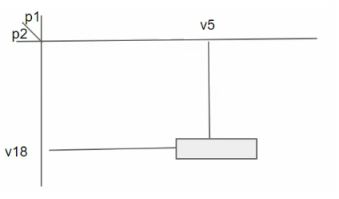
(korisni ako zelimo da obezbedimo visoku otpornost sistema da otkaz pojedinih delova ne znaci dace i taj sistem prestati da funkcionise)

* mogu se projektovati uz oslonac na neku arhitekturu npr. klijent-server, dvoslojna ili troslojna arhitektura,..

1. **konkurentni sistemi**
   * sistem se takmici za pristup nekim deljenim resursima sa drugim sistemima koji se izvrsavaju na istom hardveru (cilj da imamo više korisnika i svaki bude opslužen i obavi svoj posao)
2. INTERNE KARAKTERISTIKE

- kako softver ima organizovanu kontrolu toka (if, for,..)

1. **table driven (tabele odlučivanja)**

* uslovi od kojih zavisi tok izvršavanja aplikacije prilično kompleksni, te implementacija završava u ugnjezdenim ifovima, koja je veoma podložna greškama u implementaciji,ali i čije čitanje, razumenje pa i održavanje je teško
* umesto toga – tabele odlučivanja (matrične strukture) gde se na osama nalaze vrednosti parametara, za neku kombinaciju parametara u datom trenutku na preseku možemo dobiti akciju koju treba izvršiti u tom slučaju
* prednost
  + lepši pregled sta će se desiti
  + linearan rast vremena izvršavanja bez obzira na to koliko ma mogućih vrednosti parametara, dobili smo na vremenu reakcije
* gubimo – žrtvujemo veličinu zarad brzine izvršavanja programa (moramo zauzeti memoriju za te tabele

1. **process-driven (procesno upravljani sistem)**

* podrazumevaju da imamo način da opišemo korake procesa(proces naručivanja proizvoda i njegovog slanja, proces izrade diplomskih radova,..) i onda imamo aplikaciju koja podržava/izvršava proces (očekuje da korisnik pošalje formular, notifikuje korisnika,...)
* podrazumeva da u trenutku izvršavanja na osnovu nekih podataka možemo uticati na tok izvršavanja aplikacije (jedan od jezika za opis procesa Business Process Modeling Language ..)

1. **sistemi bazirani na znanju** **(knowledge-based)**

* podrazumevaju da sa jedne strane imao mehanizam da predstavimo znanje i sa druge neki mehanizam koji će na osnovu tog znanja moći nešto da zaključuje (formalni način opisa znanja – npr zdravstvo, zadamo simptome i onda na osnovu njih dobijemo neke moguće uzročnike)
* opredeljujuci (ogranicavajuci) faktor za osmisljavanje mehanizama za predstavljanje znanja je nivo ekspresivnosti – što je veći što slobodnije mozemo pustiti nekog da opisuje znanje njemu je jednostavnije,veća sloboda predstavljanja, ovi mehanizmi su tada komplikovaniji, za previše ekspresivne predstave ne može uopste da se radi rezonovanje
* implementacija: neuronske mreze, fuzzy programiranje,..

1. GENERIČKE OBLASTI PRIMENE

- gde su namenjeni da se koriste softverski sistemi, od toga zavise zahtevi koje treba da ispune

- neke aplikacije koje treba da podrze neke uobicajene kancelarijske poslove, takve aplikacije nece imati iste zahteve kao one od kojih zavisi avio saobraćaj ili proizvodnja u ind i sl.

- osobine u nekom sistemu i kako one utiču na razvoj softvera

**SOFTVERSKI PROIZVODI**

Razvoj softverskog proizvoda

1. prikupljanje zahteva od korisnika - specifikacija zahteva koje taj softver treba da ispuni
2. specifikacija dizajna na osnovu 1. (dizajn – način na koji će nešto biti kreirano,spec. arhitekture, vrste softvera, na koji od onih klasa-načina će softver funkcionisati)
3. implementacijana osnovu 2.

Specifikacija zahteva je nešto što spada u domen problema koji rešavamo a spec. dizajna i implementacija u domen rešenja.

**Zahtevi** mogu biti:

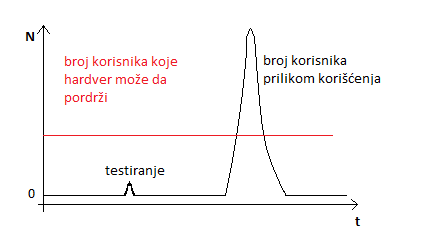
1. **funkcionalni** – šta ta aplikacija treba da radi, sta ona rešava po pitanju rešavanja problema(želim da moja aplikacija podrži proces poručivanja iz restorana, da prikaže cene, da pošalje račun korisniku,..) ali prikupljanje ovih zahteva ne mora biti trivijalno
2. **nefunkcionalni**

**Nefunkcionalni (opšti) zahtevi**:

1. **dostupnost**

* stepen u kom je softver dostupan što većem broju korisnika tj. grupi korisnika kojima je namenjen
* (geografski - da ga mozemo isporučiti fizički ili preko mreže,način interakcije korisnika sa softverom - dostupnost na različitim jezicima, prilagodjavanje interfejsa ljudima koji slabije vide, ..)
* prilagoditi softver što većem broju korisnika utiče na dostupnost

1. **adaptivnost** – odnosi se na mogućnost opstanka pri promenama u okruženju (verzije operativnog sistema, verzije drugog softvera od kojih zavisi naš, to ce njegovo trajanje eksploatacije biti duže)
2. **raspoloživost**

* deo vremena u kom je sistem funkcionalan dok on odgovara na naše akcije kod interaktivnih
* odnost ukupnog vremena koliko je sistem bio funkcionalan u odnosu na trajanje posmatranog intervala
* u praksi: aplikacija može da podrži odrednjeni broj korisnika
* problemi: kako je testirano, moglo se pretpostaviti veliki broj korisnika, a testirano je u rezimu gde sistem nije do kraja opeterćen
* rešenja:
  + jači server,i opet nećemo bitno poboljšati stvari
  + prilagoditi arhitekturu sistema prema očekivanom ponašanju, omogućiti da se aplikacija izvršava na vise servera i kapacitet tog sistema može biti proizvoljan, čak i da u napred ne znamo da će biti toliko korisnika dobro je da arhitektura takva da možemo distribuirati izvršavanje na više servera jer možemo uticati na bolje performanse
  + da radi sa promenljivim brojem servera – čime naši serveri ne bi stojali većinu vremena prazni, hardverske resurse možemo iznajmljivati po potrebi i kapaciet sistema možemo tokom vremena prilagođavati
* ovakve stvari moramo definisati na početku razvoja softvera i to odredjuje arhitekturu sistema (prvo utiče na spec. dizajna a onda i na implementaciju)

1. **kompatibilnost** – naš sistem ne remeti funkcionisanje drugih sistema u svom okruženju
2. **interoperabilnost** – mogućnost međusobne saradnje različitih sistema

* nivoi interoperabilnosti: od izolovanih (nemaju mogućnost saradnje), pa do onih koji su namenjeni da budu integrusani sa drugim sistema (dele podatke ili funkcionalnosti između sebe)
* kod sistema koji ne sarađuju međusobno, neke od podataka moramo da unesemo na više mesta, do problema dolazi kada je potrebno te podatke izmeniti/ažurirati na više mesta kada često nešto zaboravimo ili pogrešimo (problem KONZISTENTNOSTI podataka izmedju razlicitih sistema)
* interoperabilnost može biti:
  + na SINTATICKOM nivou - drugi sistem razume strukturu podataka iz našeg sistema ili obrnuto
  + na SEMANTICKOM nivou - naš sistem razume značenje podataka u nekom drugom sistemu ili obrnuto

1. **korektnost** – označava ispravno funkcionisanje algoritma u odnosu na specifikaciju (algoritam daje dobre, tačne rezulate)
2. **otpornost na greske** – sistem može postepeno da gubi funkcionalnosti u uslovima otkaza nekog dela sistema, ali na nama je kako ćemo tim greškama rukovati, da li će korisniku biti korisna poruka o gresci ili će se u trenutku kada se desi greška strušiti i neće dozvoli nikakav oporavak
3. **robusnost** –na ispravan način indikuje da je došlo do greske,veoma loše je kada se greška ignorise, te korisnik nije svestan da je do greske došlo
4. **pouzdanost** – sistemi visoke pouzdanosti nastavljaju da rade nakon pojave otkaza i njihova funkcionalnost se ne menja u nepredvidjenim okolnostima uključujuci otkaz hardvera, greške u softveru,..
5. **prenosivost** – omogućiti da se softver korisiti u različitim okruženjima, softverskim ili harverskim platformama (java)
6. **ponovna iskoristivnost** **delova razvijenog softvera -**  programer ima velike koristi od toga, uštede u vremenu razvoja (na duže staze imaju koristi i korisnici) , zgodna mogućnost da kada se uoči greška u implementaciji to izmenimo na jednom mesto i da ta izmena bude vidljiva na onim modulima koje smo razvili, greška ne mora biti u implementaciji može biti i u prikupljanju zahteva
7. **sigurnost** - nećemo nekoga ugroziti u okolini tj svoje korisnike, zdravlje, život, imovinu, moramo imati proces prema kome ćemo identifikovati, analizirati i upravljati rizicima koje imamo po pitanju toga da nas softver nekoga ugrozi
8. **bezbednost**

* zaštita informacija,njihove privatnosti, pristupa informacijama, vlasništva informacija od kradje, oštećenja , da nenamerni uticaji oštete inf da izmene ili smanje raspoloživost, ali te mere koje preduzimamo zarad bezbednosti ne smeju smanjiti raspoloživost i dostupnost ovlašćenim korisnicima
* ona može biti ugoržena:
  + u fazi razvoja
  + u fazi distribucije
  + tokom korišćenja
  + tokom održavanja ili unašređenja

1. **mogućnost održavanja** – koliko će biti jednostavno izmeniti sistem (zato što otklanjamo greške, javili su se novi zahtevi, da bi pojednostavili održavanje ili da ga uskladimo sa promenama u okruženju)
2. **mogućnost testiranja**

* stepen do kog softver ili neki njegov deo podržava tesitranje, nekada potreban dodatan napor u razvoj funckionalnosti da bi olakšali tesitranje, da bi omogućili da se neki deo može izvršavati odvojeno u nekom testnom okruženju
* odnosi se i na to da li će softver manifestovati greške u procesu tesitiranja, nećemo otkriti da greška postoji ako nema manifestaciju

1. **mogućnost korišćenja** – lakoća sa kojom korisnici mogu primeniti softver da bi rešili problem zbog kog softver postoji:
   * koliko je lako naučiti njegovo korišćenje - dokumentacija, sistem pomoći, obuke
   * efikasnost u korišćenju softvera - koliko vremena je nepohodno potrošiti za postizanje cilja
   * pamćenje – ako korisnik ne koristi softver odredjeni period da li će se lako setiti kako se šta radi
   * greške – da li će funkcionisanje našeg softvera dovesti do toga da neko pogreši u svom radu, koliko softver sprečava da pogreši i koliko mu pomaže da ne pogreši
   * zadovoljstvo – koliko su zadovoljni dok koriste softver
2. **efikasnost -** koliko sistem troši svoje resurse da zadovolji svoju namenu (memorija, procesor)
3. **performantnost** - koliko će softver biti brz, moramo na početku znati da je brzina izvršavanja nesto od čega zavisi uspešnost softverskog proizvoda pa utiče i na to kakva će biti specifikcija, dizajn, arhitektura, testiranje performansi u odgovarajućim uslovima skladnim sa načinom korišćenja