





## ***Zašto testiranje softvera ?***

Do danas nije razvijena metodologija testiranja softvera, kao najvažnija aktivnost u procesu obezbeđenja kvaliteta softvera, koja garantuje da će program biti bez grešaka, ali sistematiziran i planirani postupak testiranja softvera, kroz izvršavanje programa sa pažljivo odabranim skupom ulaznih podataka, značajno povećava poverenje u korektnost programa.

- Testiranje softvera odavno nije samo faza u procesu razvoja softvera već paralelni podproces.

Pošto je stalan pritisak, u raznim poslovnim primenama, za upotrebom novih, produktivnijih programskih jezika i razvojnih alata, sve obimniji programski kod se proizvodi u vrlo kratkom vremenskom periodu



# TESTIRANJE SOFTVERA

## ***Zašto testiranje softvera?***

Zato što su veliki gubici kompanija koje razvijaju softver upravo zbog velikog broja defekata u isporučenom softveru. Prvenstveni zadatak test inženjera je otkrivanje problema u softveru sa ciljem da se oni otklone pre predaje softverskog proizvoda kupcu.

Od test inženjera se zahteva da otkrije što je moguće više problema i to što više onih vrlo ozbiljnih čije posledice mogu biti katastrofalne sa materijalnog i bezbednosnog aspekta.

Zato je sa svih **aspekata** potrebno da se proces testiranja softvera učini što **efikasnijim** i uz što **manje troškove** ukoliko je to moguće.



# TESTIRANJE SOFTVERA

- Bez softvera danas je nemoguće zamisliti savremeni svet. Softveri su postali nezamenljiva komponenta u donošenju poslovnih odluka, naučnim istraživanjima i radu mnogih inženjera. Kompanije koje se bave razvijanjem softvera najveće gubitke beleže upravo ukoliko taj isporučeni softver sadrži određene greške.
- Kako bi se tako nešto izbeglo, osnovni zadatak test inženjera jeste da što pre **otkriju problem i da ga otklone pre nego što proizvod stigne do potrošača.**
- Od test inženjera očekuje se da proces testiranja softvera učine što efikasnijim i uz što manje troškova obezbede normalan rad određenog programa.

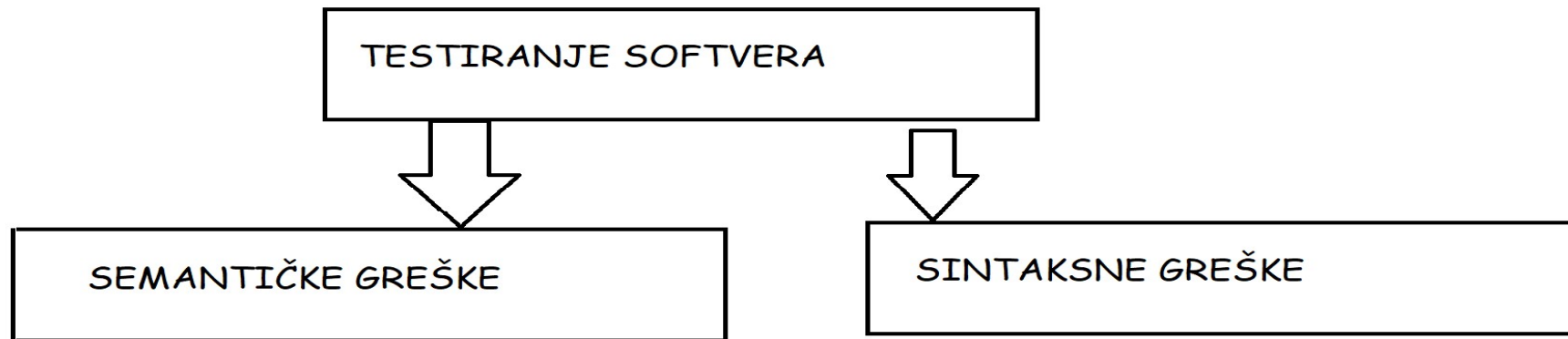


# TESTIRANJE SOFTVERA

- Softversko testiranje danas predstavlja aktivnost koja obuhvata kompletan proces razvoja i održavanja i kao takva čini veoma važan deo cele konstrukcije softvera.

Testiranje softvera nije aktivnost koja počinje samo nakon kompletiranja faze kodiranja, već aktivnost koja se izvodi zbog evaluacije kvaliteta proizvoda i njegovog poboljšanja, putem identifikovanja defekata i problema.

## Semantičke greške



### Semantičke greške obuhvataju:

- ☐ Logičke greške (greške nastale zbog primene pogresne logike u programiranju)
- ☐ Greške u kodiranju (naprimer: zamena slova koja dovodi do promene imena promenljivih )



## Sintaksne greške

Sintaksa u programskom jeziku uključuje skup dopuštenih fraza jezika, dok semantika izražava povezano značenje tih fraza.

Sintaksa definiše stroga pravila pisanja reči nekog jezika.

To su greške u zapisu programa koje nisu u skladu sa pravilima programskog jezika



## Softverska greška

Propusti u razvoju softvera se često nazivaju greškama ili bagovima.

"Greška (Bag) može izazvati neuspeh - tj. da dobijeni rezultat odstupa od očekivanog." - MAYERS

Kada se pogleda literature onda postoje sledeći termini za testiranje softvera koji se odnose za pojam greške:

- ✓ Greška (eng. *error*),
- ✓ Defekt (eng. *fault*),
- ✓ Otkaz (eng. *failure*).





## Greške / Bag

### Potencijalni izvori grešaka:

Greške se mogu generisati tokom razvoja projekta, komponenti, koda i dokumentacije.

Obično se nalaze tokom izvršavanja ili održavanja softvera na najneočekivanijim i najrazličitijim tačkama u njemu.

Greške su uzrokovane nerazumevanjem zahteva korisnika; netačna specifikacija zaheva u projektnoj dokumentaciji i drugo



## Softverska greška

- Šta je softverska greška?

Prva kompjuterska „bubica“ – HARVARD 1945 godine

Greškom se još može smatrati i pogrešan potez koji korisnik sistema može da napravi, nakon čega sistem proizvodi pogrešan rezultat.

Grešku ili nedoslednost u funkcionisanju softverskog sistema ili samog računara nazivamo *bug*.

Potrebno je razlikovati vrste grešaka koje e mogu pojaviti (error, bug, defect, failure).

**Greška (Error)**

Ljudske prirode koje pravi čovek , prilikom specifikacije zahteva ili implementacije programa.



## *Softverska greška*




- GREŠKE

Poželjno je sve greške detektovati što ranije u fazi razvoja softvera jer je ispravljanje grešaka jeftinije i brže u ranijim fazama razvoja softvera.

- OPET- VREMENSKI FAKTOR JE VEOMA BITAN!!!



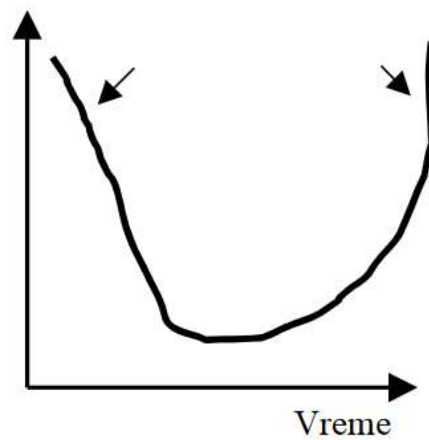
# *Softverska greška*

Error / Mistake	Defect / Bug/ Fault	Failure
Found by 	Found by 	Found by 
Developer	Tester	Customer

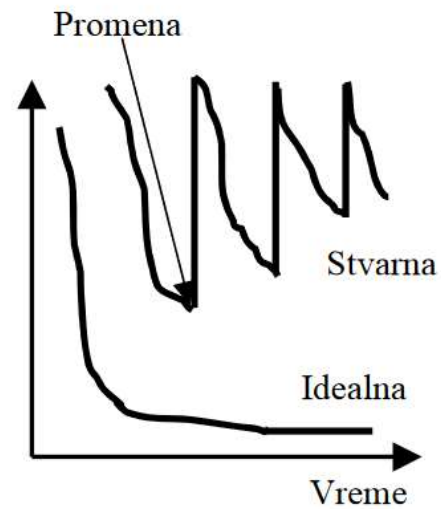
# Softverska greška

## Promene

- zahtevi se menjaju
- optimizacija



otkaz hardvera



otkaza softvera

A close-up photograph of a yellow pen tip pointing at a math problem on a piece of paper. The problem includes the expression  $10 + 6 +$  and the number 16. The background of the slide is a dark red gradient.

## Softverska greška

- ❑ Greške zbog nepoštovanja standarda
  - ✓ Postavljaju se standardne procedure za izradu softvera (greške nastaju zbog nepoštovanja i nepredržavanja zadatih procedura)
  - ✓ Nekada mogu da utiču na rad programa, ali nekad i ne moraju
  - ✓ Iako program radi, postoji mogućnost da softverske greške stvaraju probleme pri testiranju softvera i njegovom održavanju (logika rada)



## TESTIRANJE SOFTVERA- TERMINOLOGIJA

- **Greška (Error)**
  - - Napravi je čovek, prilikom specifikacije zahteva ili kodiranja programa
- • **Mana, defekt (Fault)**
  - - posledica greške (na primer, programu nešto nedostaje, ili ima funkciju ali neispravno - "bug")
- • **Otkaz (Failure)**
  - - Nemogućnost sistema da obavi zahtevanu funkciju - javlja se aktiviranjem (izvršavanjem) defektnog koda..

A yellow pen is pointing at a math problem on a piece of paper. The problem is  $10 + 6 + 16$ . The pen is pointing at the number 16.

## Softverska greška (error)

**Šta je softverska greška?**

Greškom se još može smatrati i pogrešan potez koji korisnik sistema može da napravi, nakon čega sistem proizvodi pogrešan rezultat.

**Softverska greška** (error) je deo koda koji je delimično ili totalno pogrešan najčešće kao rezultat gramatičke (sintaksne) greške.

**Logičke greške** (greske nastale zbog primene pogresne logike u programiranju)





## Softverska greška (error)

Postoje više podela bagova, [Jorg, 1995]: **GREŠKA**, (Kada ih naćine prilikom kodiranja, greške nazivamo "bagovi". Greške imaju težnju širenja) Po Jorg-u napoznatija podela je na:

- **hardverske**
- **softverske.**

Kada se testiranjem programa utvrdi da se on ne ponaša kao što se očekuje, pristupa se analiziranju i lociranju bagova.

Detaljan postupak indentifikaciji bagova omogućuje se metodićkim pristupom testiranju programa.

Potruga za bagovima olakšava testiranje programa, ali ne može adekvatno da ga zameni.

Poznato je takođe da nikakvo testiranje ne može da otkrije sve bagove.

U vezi sa testiranjem programa često se rade statićke analize (istražuju se osnovni programi, pri tome traže se osnovni problemi i prikupljaju podaci bez izvršavanja programa) i dinamićke analize (istražuje se ponašanje programa u izvršenju i tako se dobijaju podaci vezani za puteve izvršavanja, vremenske profile i pokrivenosti samog testiranja programa.



## Softverska greška (error)

- **Šta je softverska greška? (1 ili 5 metara!)**

Grešku možemo posmatrati kroz nekoliko primera.

Ako uzmemo npr. razliku od 5 metara između izračunatog rastojanja dve tačke i tačnog rastojanja, to predstavlja grešku koju možemo pripisati loše prikupljenim zahtevima, odnosno podacima.

Ako uzmemo npr. razliku od 1 metara između izračunatog rastojanja dve tačke i tačnog rastojanja, to predstavlja grešku koju možemo pripisati loše prikupljenim zahtevima, odnosno podacima.

Isto tako pogrešno odabrana instrukcija ili pogrešan rezultat neke operacije u programu predstavlja grešku koja može dovesti do pogrešnih podataka u sistemu.

A close-up photograph of a yellow pen tip pointing at a piece of paper with handwritten math problems. One problem is  $10 + 6 =$  and another is  $16$ . The background is a solid dark red color.

# TESTIRANJE SOFTVERA

***Zašto je bitno praviti izveštaje o greškama?***

1. Proces izveštavanja o greškama i preduzetim akcijama mora biti precizno definisan.
2. Česti su slučajevi otežanog identifikovanja grešaka usled nerazumevanja zahteva.
3. Postoje i problemi kada neke izmene u kodi (nove opcije) nisu pravilno definisane.
4. Sam proces prihvatanja zahteva i ispravljanja mora biti vođen po unapred definisanoj proceduri.
5. Mora se dati precizan odgovor na pitanja ko prijavljuje grešku, kako je prijavljuje, kako je opisuje, kakvog je tipa, kakav je status greške, ko prihvata i dalje delegira zahtev itd...



## Softverski defekti

U literaturi u nekim knjigama se navodi kao termin Nedostatak (*fault, defect, bug*)

**Softverski defekti** (*faults*) su softverske greške koje uzrokuju nepravilno funkcioniranje softvera tokom specifične primene.

**Fault je softverski defekt** koji prouzrokuje failure, Failure je neprihvatljiva programska operacija u odnosu na programske zahteve.

**DEFEKT**, (Defekt je rezultat greške.

Preciznije rečeno, defekt je prikaz greške, pri čemu je prikaz način izražavanja, kao što je opisni tekst, dataflow dijagrami graf hijerarhije, osnovni kod itd.)



## Softverski defekti

### Nedostatak (fault, defect, bug)

- ✓ mana u programu koja može da onemogući program da izvrši namenjenu funkcionalnost
  - ✓ ukoliko se pri izvršavanju programa izvrši ovaj deo koda

### Nedostatak se manifestuje pojavom otkaza

- u toku testiranja ili
- Kada se softver koristi



## Softverski defekti

Statistike pokazuju da određeni broj softverskih defekata se preokrene u softverske otkaze u ranim ili kasnijim fazama primene aplikacije.

Ostali softverski defekti ostaju skriveni, nevidljivi za korisnike softvera, ali uvek postoji mogućnost da se aktiviraju ako se situacija izmeni.



# Softverski defekti

- Defekt

- greška u softveru koja uslovljava da se softver ne ponaša kao što je očekivano. Defekti u programskom kodu često se nazivaju bagovi (engl. Bug ).

- Pronalaženje defekata

- proces analize softvera kako bi se pronašlo šta uzrokuje pronađenu programsku grešku. Često se naziva debugovanje (engl. Debugging).

Pošto se programska greška pronađe i ispravi, sistem postaje stabilniji.





## Reliability

- **Pouzdanost softvera** (reliability) se definiše kao verovatnoća izvršavanja operacija softverskog sistema bez greške (*failure-free operation*) za specifično vreme u specifičnom okruženju.

Pouzdanost se definiše kao niz atributa koji predstavljaju mogućnost softvera da održi svoj nivo performansi pod određenim uslovima u određenom periodu vremena i ona je jedan od ključnih faktora kvaliteta softvera.(ISO 9126)





## Reliability

- ✓ IEEE 982.1-1988 definiše upravljanje pouzdanošću softvera kao proces optimizacije softvera kroz tri aktivnosti u kontekstu ograničenja u projektu kao što su resursi, vreme i performanse:
- ✓ Prevencija greške (error)
- ✓ Detekcija i otklanjanje fault-ova
- ✓ Merenje da bi se dobila maksimalna pouzdanost za dve aktivnosti (prevencije greške i Detekcija i otklanjanje fault-ova)



# OTKAZ

**OTKAZ**, (Otkaz se dešava kada dođe do defekta.)

Procena **pouzdanosti** zasniva se na analizi otkaza softvera i njihovog uticaja na celokupan system.

**Otkaz** se može definisati i kao odstupanje ponašanja, softvera u odnosu na zahteve ili specifikaciju softverskog proizvoda.

**Otkaz** softvera je evidentan događaj postojanja greške u softveru tj. posledica, manifestacija greške u softveru.

**Otkaz** softvera može biti posledica grešaka u kodu, pogrešnog korišćenja, pogrešne interpretacije specifikacije koju softver treba da zadovolji ili nestrupnosti programera, primene neodgovarajućih testova ili drugih nepredviđenih problema.

**INCIDENT**, (Kada se desi otkaz, korisnik to možda neće ni primetiti.

Incident je simptom praćen otkazom koji obaveštava korisnika o dešavanju



# TESTIRANJE SOFTVERA

- Potencijalni bagovi, u prošlosti, sadašnjosti i budućnosti
- Aritmetički bagovi
- Deljenje sa nulom
- Gubitak preciznosti zbog zaokruživanja
- Logički bagovi
- Beskonačna petlja
- Beskonačna rekurzija



## TESTIRANJE SOFTVERA - TERMINOLOGIJA

### ✓ Test slučaj

- stanje opisano akcijom koju treba izvršiti, podacima koje treba poslati sistemu, kao i očekivanim odgovorom od strane softvera. On opisuje šta će biti testirano (stavke), korake i podatke koji će biti korišćeni u testiranju softvera. Umesto ovog termina u literature se koristi i termini test primer ili slučaj testiranja.

### ✓ Test scenario

- sekvenca akcija koje treba izvršiti tokom testiranja sistema kako bi se proverili jedan ili više test slučajeva.

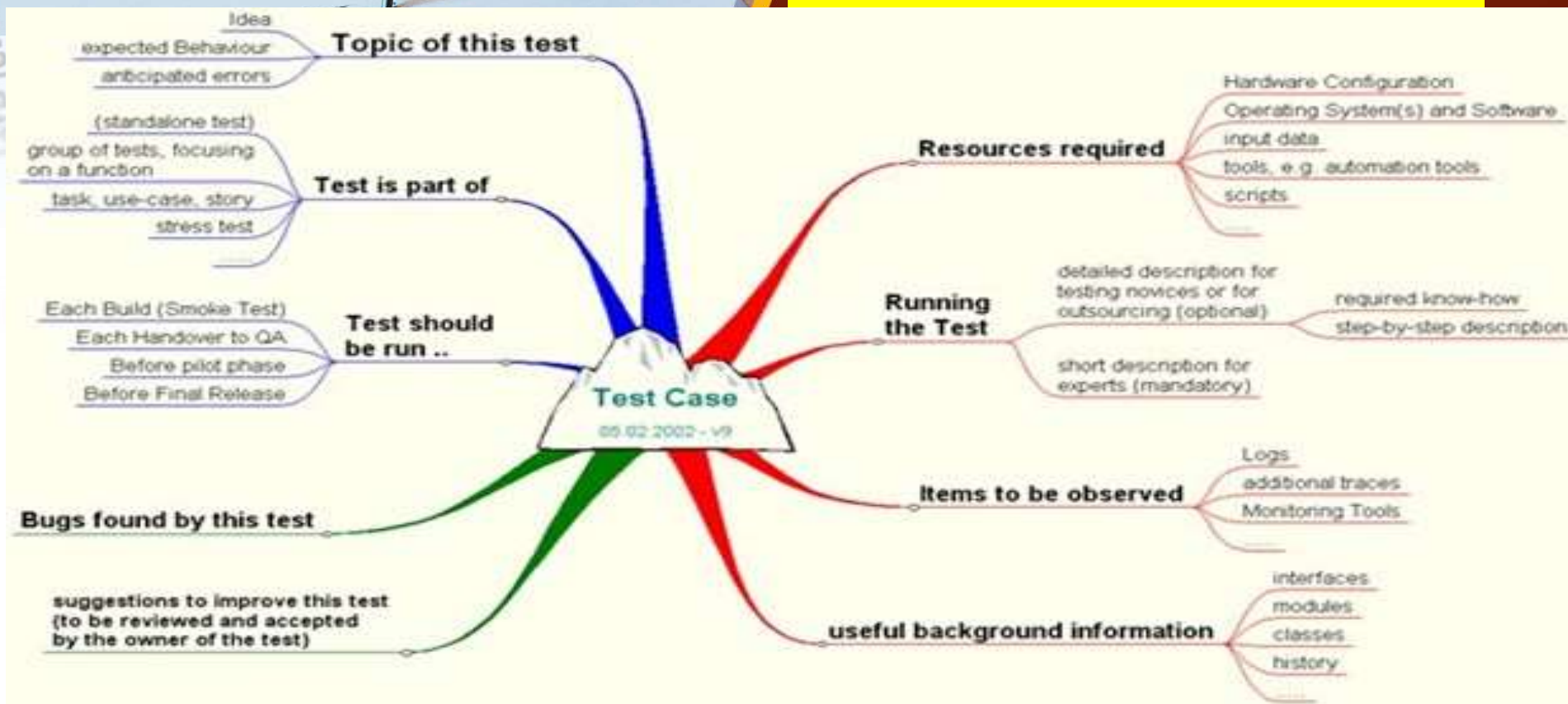
Često se umesto termina test scenario koristi termin test procedura ili test skript



## TEST SLUČAJEVI

- ✓ Faza gde se određuju softverski i hardverski uslovi testiranja određenog proizvoda.
- ✓ Identificiranje test slučajeva može potrajati i ponekad moramo ponoviti test i po nekoliko puta.
- ✓ Testni slučajevi sadrže niz radnji koje izvodimo kako bismo poboljšali određenu funkciju ili funkcionalnost.

# TEST SLUČAJEVI







## TEST SLUČAJEVI

- Testni slučajevi su jedan su od najvažnijih delova životnog ciklusa razvoja softvera koji je odgovoran za razvoj programa. ( i na kraju da li on radi ispravno ili ne).
- Ovde postoje i takozvani test scenariji -To je skup skup testnih slučajeva, koji određuju pozitivne i negativne aspekte projekta kako bi se dala ocena softvera a u isto vreme identifikovali potencijalni nedostaci u program.

## TEST CASE

## VERSUS

## TEST SCENARIO

Test Case	Test Scenario
It's a set of variables or conditions which determine the viability of a software application.	It's a series of test cases executed one after the other to determine the functionality of the system or application.
It is a detailed document consisting of application requirements, preconditions, test data, post conditions and expected results.	It is a detailed test procedure consisting of test cases which help find problems in the system and evaluating results.
QA team and development team write test cases.	Reviewed by business analyst/ business manager.
It is important when development is done onsite and testing is done off-shores.	It is beneficial when time to build test cases is less.
More resources are required for writing test cases which is a waste of time and money.	It's a collaborative effort which reduces complexity thereby saving time and money.





## Test objekt

### □ Test objekt

- Komponenta (Softverska) koja se testira

Testiranje je svako izvršavanje ili analiziranje test objekta kako bi se utvrdilo njegovo ponašanje

- porede se stvarno i očekivano ponašanje da bi se utvrdilo da li test objekt zadovoljava zahteve

### □ Uslovi testa (test conditions)

- koje osobine ili funkcionalnosti test objekta treba verifikovati
- verifikacija se vrši kroz test slučajeve



# TESTIRANJE SOFTVERA

- Česti bagovi- neki primeri

## Sintaksni bagovi

- Korišćenje pogrešnog operatora, na primer operatora dodele vrednosti umesto operatora poređenja jednakosti

- **Resursni bagovi**

Dereferenciranje NULL pokazivača

Korišćenje neinicijalizovane promenljive

Pristup nedozvoljenom području memorije (segmentation fault)

Preduboka rekurzija, koja ima za posledicu prepunjavanje steka



## TESTIRANJE SOFTVERA - TERMINOLOGIJA

- Programski kod
  - tekst napisan na nekom jeziku koji se može ili izvršiti direktno od strane računara ili se može prevesti u oblik razumljiv računaru.
- Dizajn testa
  - ✓ - proces analize i specifikacije test slučajeva i scenarija koji će biti korišćeni tokom testiranja.
  - ✓ U dizajn testa pored funkcionalnih test slučajeva mogu da se uključe dizajn testa sigurnosti, opterećenja, platforme, lakoće korišćenja i slično.
  - ✓ U literaturi to se još naziva projektovanje test slučajeva.



# TESTIRANJE SOFTVERA

- Poremećaj (Incident)

simptom kojim korisnik postaje svestan otkaza.

- Testiranje

postupak izvršavanja softvera sa test primerima.

Dva su različita cilja testiranja: da se nađu otkazi, ili da se demonstrira ispravno izvršavanje

Testiranje može otkriti otkaze, ali potom treba eliminisati defekte debugovati program, što je posebna aktivnost



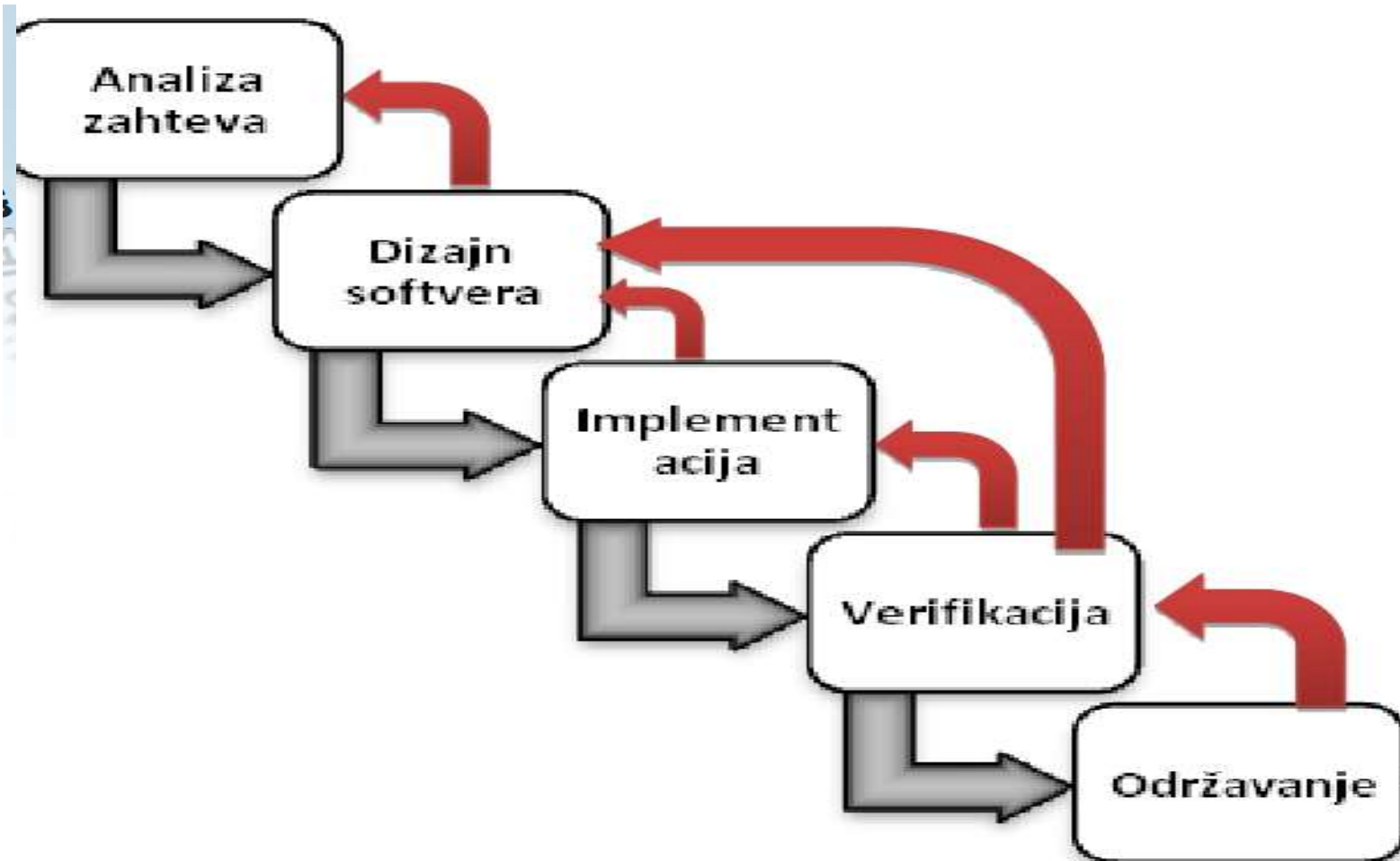
## MODEL I ŽIVOTNOG CIKLUSA SOFTVERA

Životni ciklus softvera obuhvata period od definicije zahteva do prestanka njegovog korišćenja.

- ❖ Model životnog ciklusa opisuje procese iz razvoja, korišćenja i održavanja softverskog proizvoda u toku njegovog životnog ciklusa.
- ❖ Za konkretan proizvod potrebno je izabrati konkretan model (implementirati standard).
- ❖ Ne postoji jedinstveni optimalan model za sve softverske proizvode, obično se primenjuje neki od standardnih modela ili neka kombinacija istih



# SOFTWARE TESTING LIFE CYCLE (STLC)





## MODEL VODOPADA

Faze se sekvencijalno izvršavaju dok se ne završi projekat. Faze u modelu vodopada su:

- Analiza zahteva -faza gde se sakupljaju zahtevi od krajnjih korisnika, analizira šta je potrebno raditi, kreiraju ugovori kojima se definiše šta će biti urađeno i potvrđuju zahtevi koji će biti implementirani.
- Dizajn softvera -faza gde se detaljnije analiziraju zahtevi prikupljeni tokom analize, specificira kako će se implementirati softver, kreira tehnička specifikacija i dizajn softvera. Dizajn softvera predstavlja konkretan plan kako će biti implementiran sistem od generalne arhitekture softvera do detaljnog opisa implementacije pojedinih komponenti i algoritama. Na kraju ove faze je poznato kako će se sistem implementirati

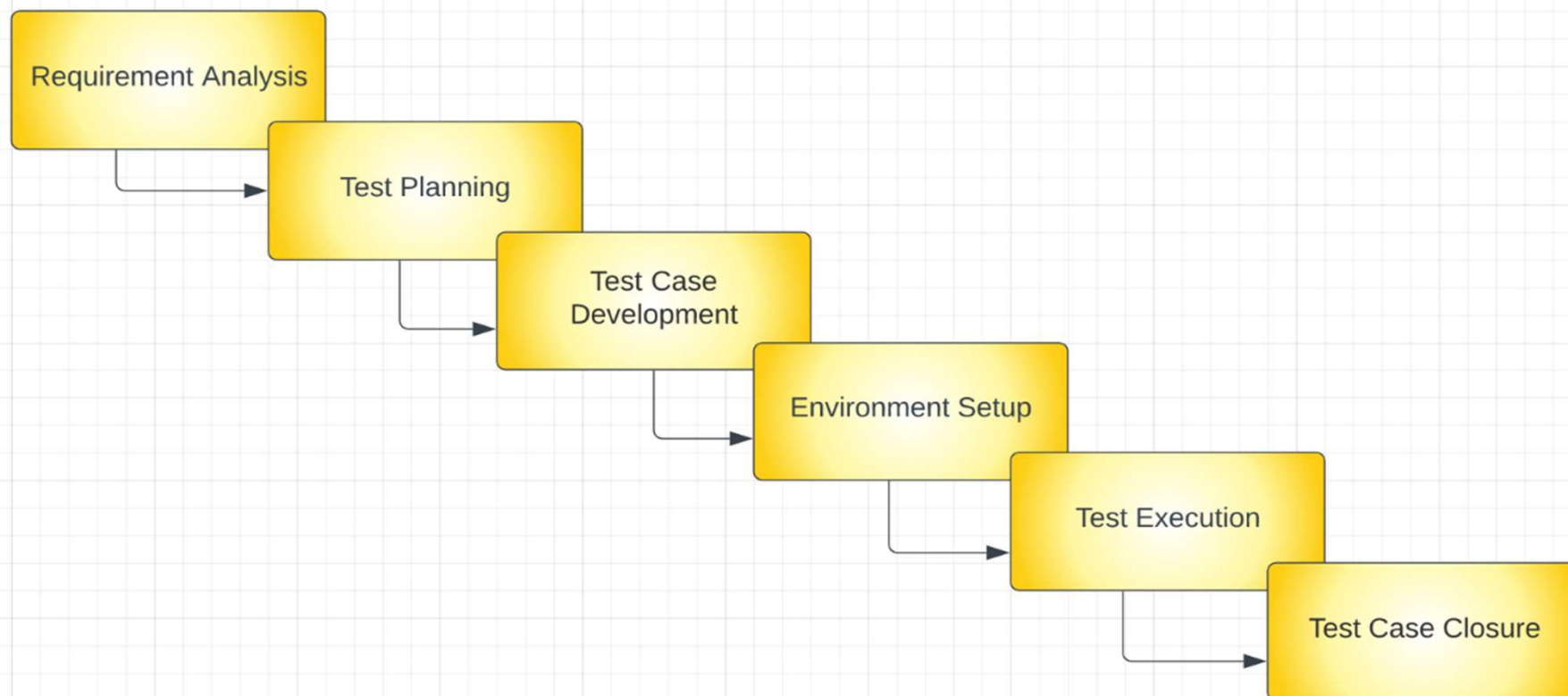




## MODEL VODOPADA

- Implementacija -faza gde se projektovani softver implementira u određenom programskom jeziku i platformi. Na kraju ove faze softver je završen i spreman za upotrebu.
- Verifikacija -faza u kojoj se planira testiranje, testira sistem koji je implementiran u prethodnoj fazi, prijavljuju i otklanjaju problemi nađeni u softveru. Na kraju ove faze softver je testiran i spreman na predaju krajnjim korisnicima.
- Održavanje -tokom faze održavanja softver je predat krajnjim korisnicima i vrše se eventualne dorade u skladu sa izmenama traženim od korisnika. Ova faza traje sve dok korisnici upotrebljavaju softver.

# Životni ciklus razvoja softvera





## Test Planning

Kada se završi faza analize zahteva može preći na fazu planiranja testiranja (*Test Planning*).

Faza planiranja podrazumeva kreiranje test plana - dokumenta plana testiranja koji služi kao strateški plan za testiranje.

Dokument sadrži detaljnu procenu vremena i troškova, potrebne resurse, podelu odgovornosti i određivanje okruženja za testiranje



## Faza analize zahteva (*Requirement Analysis*)

Podrazumeva analizu funkcionalnih i nefunkcionalnih zahteva, kao i pregled korisničkih zahteva pri čemu se definišu test zahtevi.

U ovoj fazi se definišu tipovi testova koji će se pisati, definišu prioriteti zahteva, priprema se dokument za praćenje korisničkih zahteva pomoću test slučajeva, određuju detalji test okruženja (*Test Environment*) u kojem će se izvoditi testovi i ukoliko je moguće kreira izveštaj o izvodljivosti automatizacije.



## Test Case Development

Test slučajevi se kreiraju u fazi razvoja test slučajeva (*Test Case Development*).

Testovi se zatim proveravaju tako da svaki deo softvera radi kako je predviđeno za krajnjeg korisnika.

Razvoj test slučajeva zavisi od obima zahteva i kriterijuma testiranja.

U ovoj fazi se, ukoliko je to moguće, kreiraju skripte za automatizaciju.



## Environment Setup i Test Execution

- ❑ Postavka test okruženja (*Environment Setup*) je važna jer predstavlja uslove pod kojima će se izvršavati testovi. Ova faza uključuje stvaranje test okruženja koje blisko simulira okruženje iz stvarnog sveta. Tim za testiranje koristi ovo okruženje za testiranje cele aplikacije.
- ❑ Faza implementacije testova (*Test Execution*) je faza izvršavanja napisanih test slučajeva pri čemu se proizvod validira i pronalaze greške. Ukoliko greške postoje one se prijavljuju razvojnom timu koji dalje rešava problem nakon čega se on ponovo testira.



## Test Case Closure

- ❑ Nakon faze implementacije testova sledi faza završetak životnog ciklusa testiranja softvera (*Test Case Closure*).
- ❑ U ovoj fazi tim se okuplja i analizira ceo tok trenutnog ciklusa testiranja i dolaze do zaključka koje će im u budućim projektima koristiti za poboljšanje načina testiranja.



# Životni ciklus razvoja softvera (Software development life cycle, SDLC)

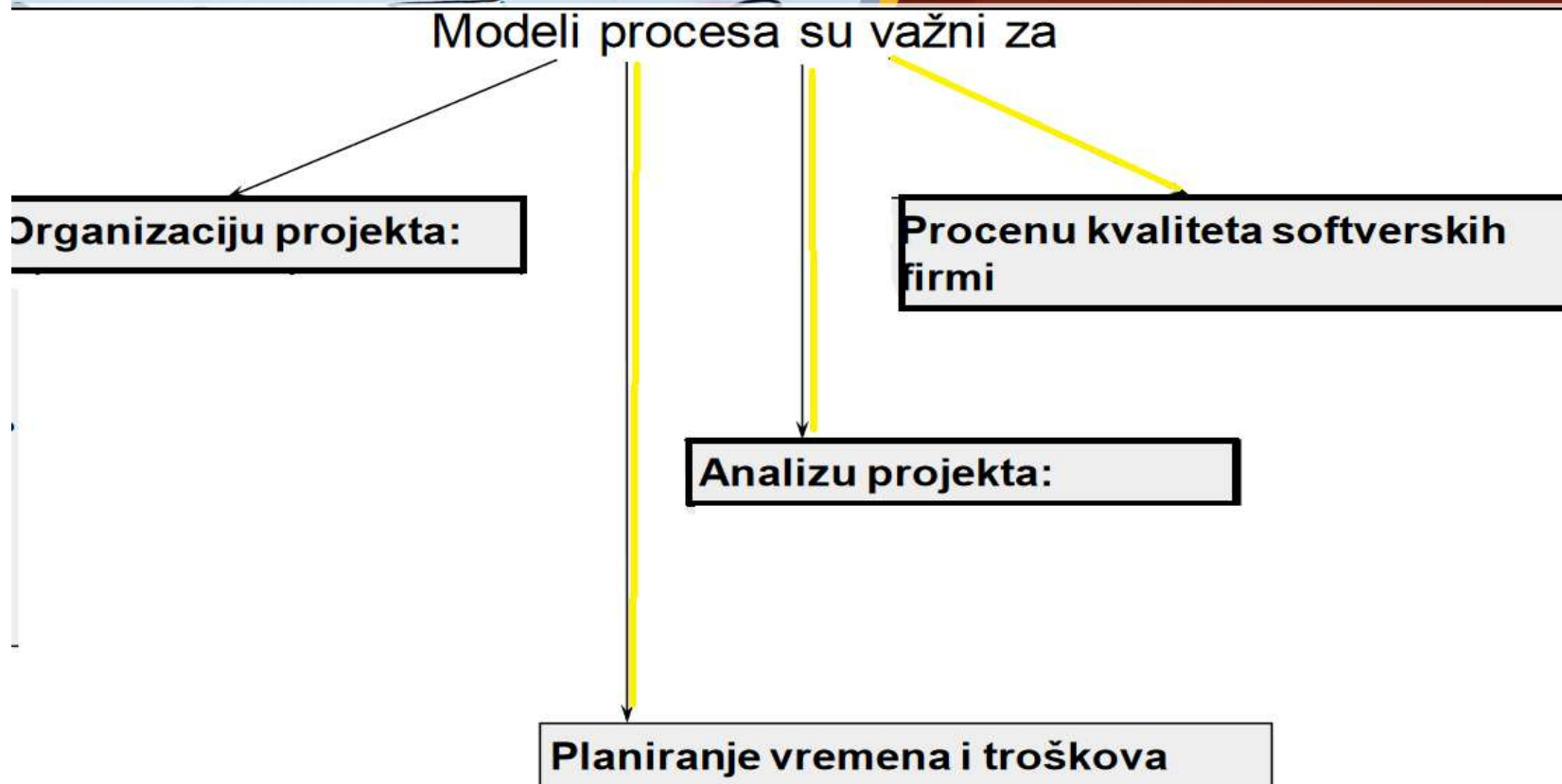
Modeli procesa su važni za

Organizaciju projekta:

Procenu kvaliteta softverskih  
firmi

Analizu projekta:

Planiranje vremena i troškova



# KASKADNI MODEL

Analiza  
zahteva

Projektovanje

Kodiranje

Testiranje

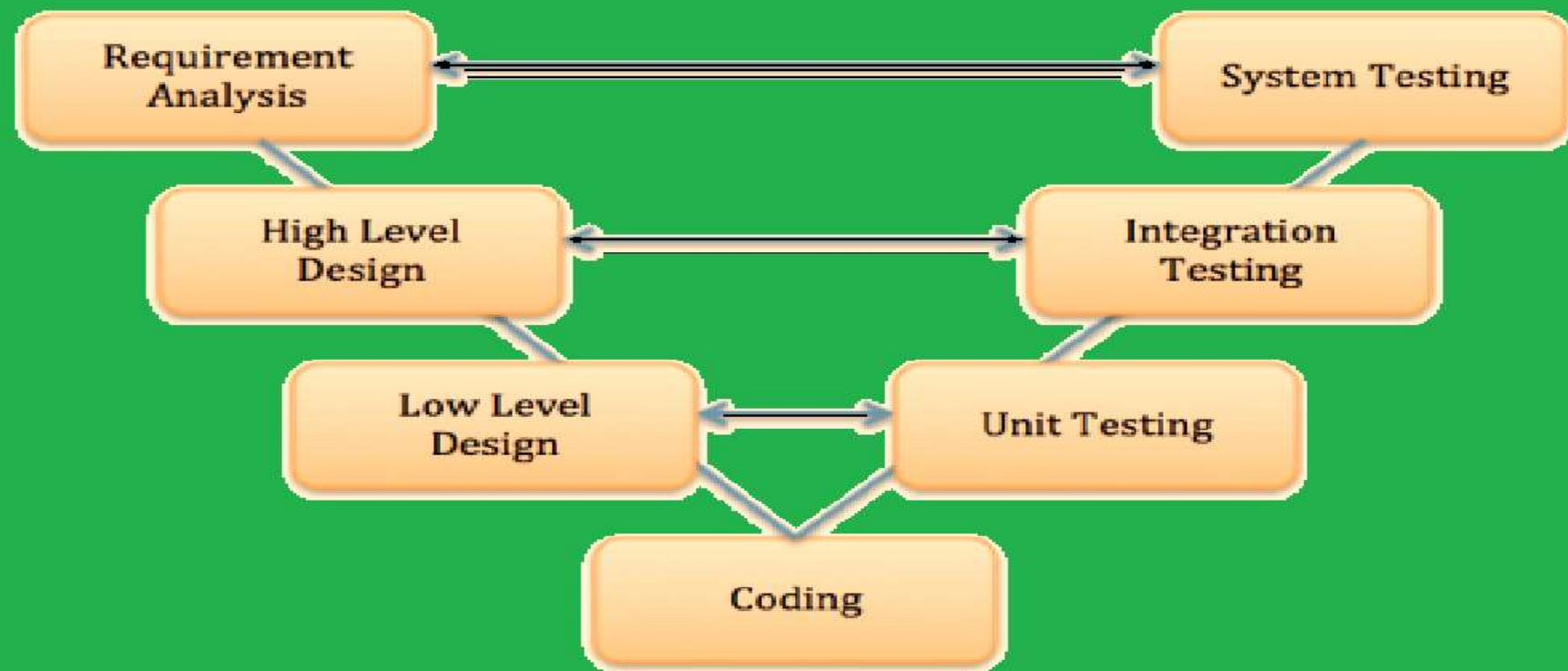
Isporuka i  
održavanje

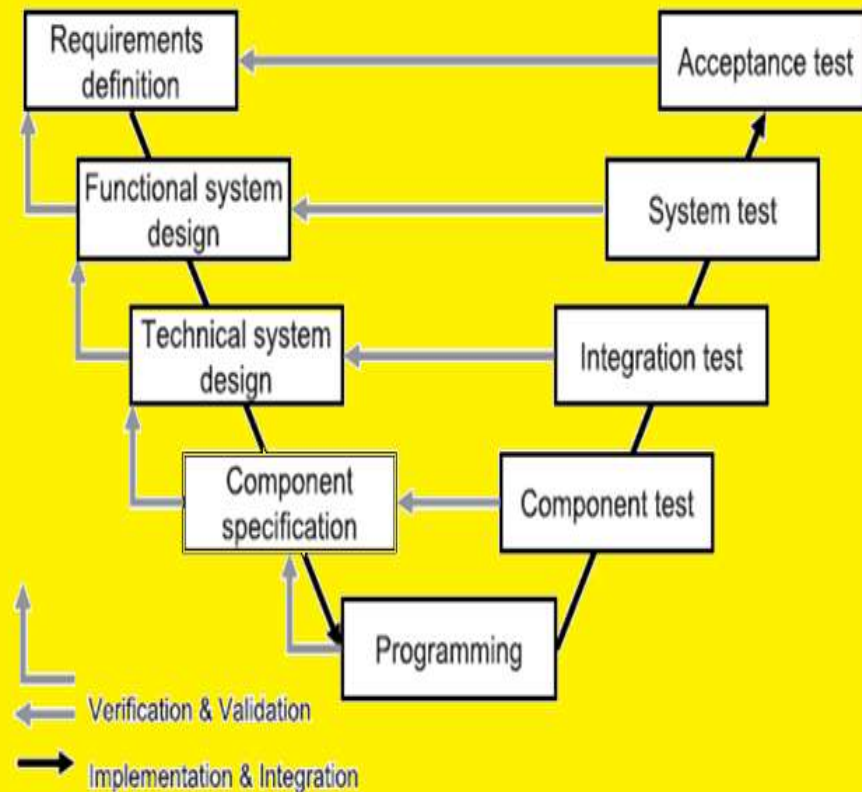
*Model vodopada, Royce, 1970.g.,  
najstariji model*

## Osobine

- ❑ veoma visok nivo apstrakcije
- ❑ kaskadna veza faza
- ❑ kritične tačke i međuproizvodi

# V model životnog ciklusa razvoja softvera





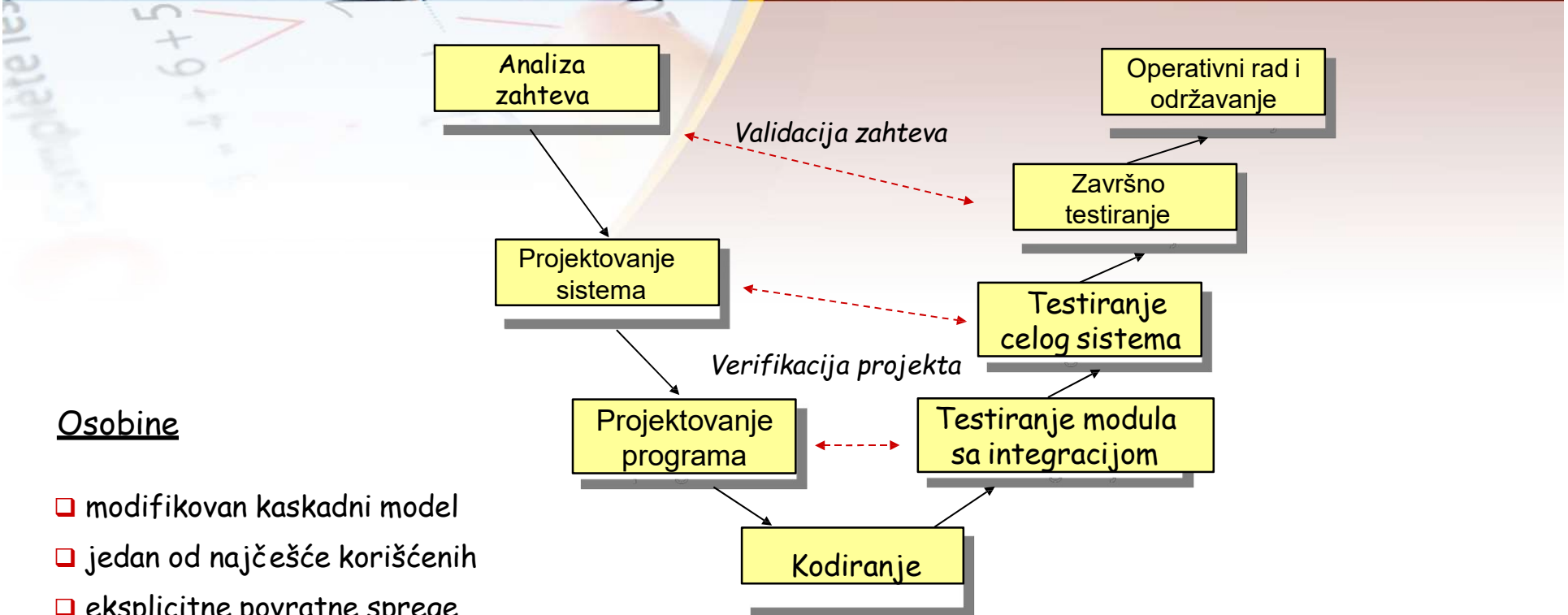
## V model

Iz ovoga se vidi koliki je značaj testiranju softvera a koliki je značaj razvoja softvera.

- ✓ Razvoj funkcionalnosti softvera je prikazan na levoj strani slike
- ✓ Proces testiranja i integracije je prikazan na desnoj strani slike
- Svaka faza u testiranju softvera verifikuje jednu od faza razvoja softvera



## V MODEL

Osobine

- ❑ modifikovan kaskadni model
- ❑ jedan od najčešće korišćenih
- ❑ eksplicitne povratne sprege

*Nemačko Ministarstvo odbrane, 1992.g*



## Aktivnosti vezane za testiranje V Model

### ❑ Testiranje komponenti

Component testing je testiranje komponenti koje nastaju spajanjem više jedinica koda

- zove se i jedinično testiranje
- nezavisno testiranje svake pojedinačne komponente da se utvrdi da li zadovoljava zadatu specifikaciju ili ne

### ❑ Integraciono testiranje

- otkriti potencijalne probleme i greške u interakciji između integrisanih celina.
- Integration testing se koristi da bi se proverile veze između različitih komponenti koje zajedno čine neki deo sistema



## Aktivnosti vezane za testiranje V Model

- ❑ Sistemsko testiranje
  - verifikacija da li kompletan sistem zadovoljava funkcionalnosti kada se svi njegovi delovi integrišu u jednu celinu
- ❑ Test prihvatljivosti (eng. *acceptance test*)
  - proverava da li sistem zadovoljava korisnikove zahteve i očekivanja





## V-modelu

U V-modelu postoje pravila kojima se definišu preduslovi za prelazak u sledeću fazu:

- ✓ Iz faze analize zahteva se može preći u fazu specifikacije samo ako su analizirani zahtevi i ako je definisano kako će se ti zahtevi testirati tokom testa prihvatljivosti.
- ✓ U fazu dizajna sistema se može preći ako je završena faza specifikacije sistema i definisano kako će se testirati kompletan sistem.
- ✓ U fazu dizajna pojedinih modula se može preći ako je dizajnirana arhitektura sistema i definisano kako će se testirati komponente tokom integracije. • U fazu kodiranja se može preći ako su dizajnirani moduli koji će se kodirati i ako je definisano kako će se ti moduli testirati.



## V-modelu

Druga značajna izmena je definisanje više nivoa testiranja kojima se proveravaju različiti delovi sistema. Nivoi testiranja po V-modelu su:

- ✓ Jedinično testiranje kojim se testiranju pojedini delovi sistema (moduli, komponente, forme).
- ✓ Integraciono testiranje kojim se testira komunikacija, povezivanje i tokovi među modulima.
- ✓ Sistemsko testiranje kojim se testira sistem u celini.
- ✓ Test prihvatljivosti kojim krajnji korisnici potvrđuju da aplikacija radi upravo ono što im treba



## Modeli razvoja softvera

- Postoji veliki broj različitih modela razvoja softvera kojima se uklapaju pomenute aktivnosti. Modeli razvoja softvera mogu se podeliti na sledeće grupe:
  - ✓ 1. **Fazni modeli** - modeli u kojima se aktivnosti razvoja dele po grupama koje se rade sekvencijalno po fazama.
  - ✓ 2. **Iterativni modeli** - u kojima se projekat deli na manje periode (iteracije) u kojima se vrše sve aktivnosti u razvoju.
  - ✓ 3. **Agilne metode** - slične iterativnim ali se na manje formalan način kombinuju aktivnosti razvoja softvera.



## FAZNI MODELI

- Fazna priroda modela vodopada zasniva se na „zdravo razumskoj“ podeli posla.
- Koji god posao je potrebno izvršiti logično je da se najpre analizira šta će biti urađeno, potom da se odredi kako će to biti urađeno, a nakon izvršenja posla potrebno je proveriti da li je posao stvarno urađen kako treba.
- Model vodopada je samo primena ovog načina rada u softverskim projektima



## Iterativni model

U iterativnom procesu razvoja potrebno je napraviti strategiju rešavanja problema.

Tokom testiranja pronalaze se programske greške, koje je potrebno rešiti, što zahteva određeno programersko vreme koje se troši na uštrb vremena planiranog za razvoj funkcionalnosti.

U slučaju da je programski kod napravljen tokom iteracije suviše nestabilan, neće biti dovoljno vremena da se završe sve planirane funkcionalnosti do roka.



## Iterativni model

- Pošto je kraj iteracije fiksna bez obzira na implementirane funkcionalnosti i probleme koji se dese potrebno je odlučiti šta treba raditi sa greškama koje se nađu.
- Neke od mogućih strategija su:



## Iterativni model

- Ukoliko su nađene greške koje se mogu rešiti u razumnom roku, vreme za rešavanje grešaka uklapa se u vreme koje će se trošiti u okviru iteracije
- Kreiranje scenarija za sledeću iteraciju
- U slučaju da nije moguće rešiti sve greške u okviru iteracije i kompletirati planirane funkcionalnosti, iteracija se proglašava za neuspešan i svi zahtevi se vraćaju u glavni log.



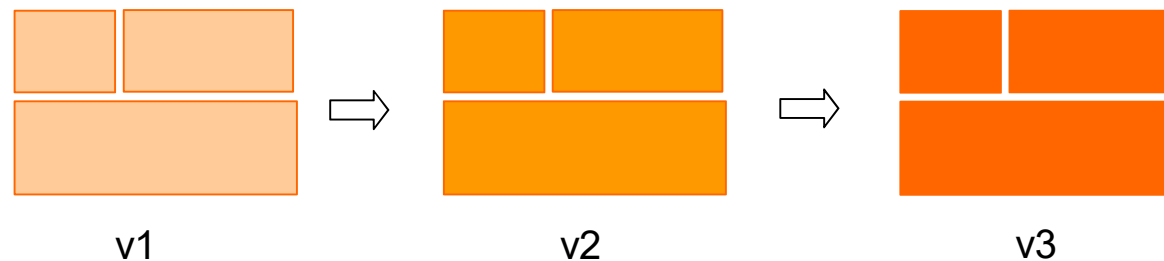


## Iterativni model

- ❖ Specifikacija je precizno definisana i potpuno jasna.
- ❖ Glavni zahtevi moraju biti definisani; međutim, neke funkcionalnosti ili zahtevana poboljšanja mogu se razviti (potrebno je vreme).
- ❖ Vremenski rok je precizno definisan ( i on je jako bitan faktor za distribuiranje na tržište)
- ❖ Koriste se nove tehnologija
- ❖ Često postoji slučaj da Programeri nisu dostupni u potrebnom broju.
- ❖ Postoje visok rizik da se neke funkcije softvera i ciljevi mogu promeniti u budućnost

# ITERATIVNI RAZVOJ

- ❑ podela sistema na podsisteme (faze, verzije) prema funkcijama
- ❑ u svim verzijama se isporučuje ceo sistem, uz menjanje funkcija svakog podsistema





## Agilne metode

Ovi modeli razvoja softvera predstavljaju novi pristup razvoju softvera, koji se bori protiv formalizacije i birokratizacije procesa razvoja softvera krutim modelima razvoja, u kojima se precizno moraju definisati sve potrebne faze i aktivnosti kako bi se kompletirao projekat

- Agilni model je pokušaj da se ukloni formalizacija procesa razvoja softvera, minimizuju prateće aktivnosti koje nisu direktno vezane za razvoj softvera i tako dobije maksimalna efikasnost



## Agilne metode

- Osnovna karakteristika agilnog razvoja softvera je iterativno/spiralni razvoj u kome se projekat, umesto na dugačke faze ili verzije kao u prethodnim modelima, deli na kratke iteracije (trajanja od dve do tri nedelje), u okviru kojih se vrše sve potrebne aktivnosti analize, dizajna, kodiranja i testiranja kao i u ostalim modelima



## Agilne metode

- U okviru svake iteracije, koja može trajati dan, nedelju ili par nedelja, ponavljaju se aktivnosti analize, dizajna, implementacije i testiranja gde je fokus samo na funkcionalnostima koje se implementiraju u trenutnoj iteraciji.
- Po završetku jedne iteracije kreće se u novu po istom šablonu. Ideja agilnih procesa je napuštanje formalizacije nastale modelom vodopada i fokusiranje na ljude, efikasnost i komunikaciju, a ne na procese, dokumentaciju i planove kao u formalnim metodama.



## Agilne metode

Agilni sistem testiranja je jedan od najpopularnijih, u kompanijama koje su prepoznale značaj faze kvalitativnog ispitivanja. Neko id koraka kada za testiranje koristi ova metoda su:

- ✓ **Planiranje sprintova**
- ✓ **Kreiranje scenarija testiranja**
- ✓ **Verifikacija kvaliteta**
- ✓ **Ispitivanje stabilnosti**
- ✓ **Ispravljanje, testiranje i isporuka softvera**

Odlika agilne metodologije jeste organizacija posla u sprintovima. U praksi to izgleda tako što se veliki projekti razbiju na manje celine, I svaki od tih delova se "daju" određenom timu. Sprintovi su uglavnom vremenski ograničeni I pre samog početka se definiše cilj koji se u tom roku mora izvršiti.



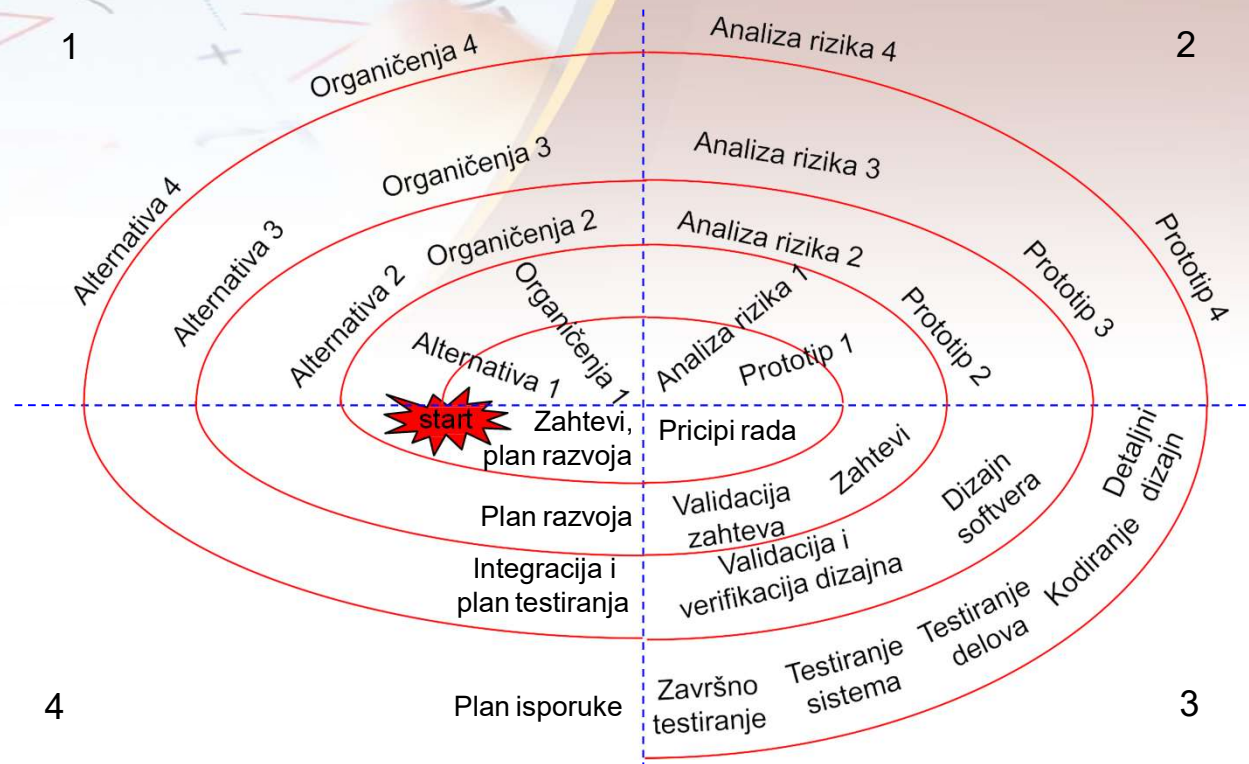
## Agilne metode

U agilnim metodama, korisnik je deo razvojnog tima i odgovoran je za donošenje odluka o prihvatljivosti sistema.

- ✓ Testovi su definisani sa strane korisnika i integrisani su sa drugim testovima na taj način da se izvršavaju automatski kada se naprave promene.
- ✓ Ne postoji odvojen proces testiranja prihvatljivosti.
- ✓ Osnovna dilema je da li je uključen korisnik radi po pravilima ili ne radi.



# SPIRALNI MODEL





# SPIRALNI MODEL

- Model predstavlja iterativni razvoj u četiri koraka:
  1. **Zahtevi i planiranje životnog ciklusa**
  2. **Planiranje razvoja**
  3. **Planiranje integracije i testiranja**
  4. **Implementacija**
- Svaka iteracija obuhvata pun krug i prolazi kroz četiri kvadranta:

kvadrant 1: određivanje ciljeva, alternativa i ograničenja	kvadrant 2: evaluacija alternativa, identifikacija i procena rizika
kvadrant 4: planiranje sledeće iteracije	kvadrant 3: razvoj i verifikacija putem testiranja



## TESTIRANJE SOFTVERA

Proces testiranja mora se odvijati tokom svih faza životnog ciklusa.

Tester će izvoditi testove na osnovu planova ispitivanja i pripremljenih test slučajeva.

Za ovaj process moraju se realizovati standardizovani propisi i procedure koje definišu način rada i aktivnosti test tima, kao i svih ucesnika u razvoju.



## Kako poboljšati sam proces testiranje softvera?

- Poboljšanja procesa testiranja je konstantan proces zajedno sa svim drugim elementima razvoja softvera. Angažuju se sve veći resursi kako velikih tako i malih kompanija u proces testiranja i njegovo unapredjenje. Postoji više aspekata koji doprinose poboljšanju procesa testiranja:
  - ☐ obuka kadrova za proces testiranja
  - ☐ automatizacija procesa testiranja



## Kako poboljšati sam proces testiranje softvera?

- ☐ automatizacija procesa testiranja
- ☐ razvoj novih alata
- ☐ razvoj novih modela testiranja
- ☐ integracije procesa merenja parametara efikasnosti i efektivnosti procesa testiranja
- ☐ analize slabih i jakih strana postojećeg procesa testiranja
- ☐ identifikacije rizika i njihovih posledica na uspeh procesa razvoja



## Šta danas predstavlja testiranje softvera?

Testiranje predstavlja važan deo životnog ciklusa razvoja softvera.

Softver se implementira prema zahtevima korisnika sa ciljem rešavanja realnog problema ili kreiranja potrebne funkcionalnosti.

Nakon implementacije, softver može u manjoj ili većoj meri odgovarati zahtevima.

Svako ponašanje softvera koje se ne slaže sa zahtevima predstavlja grešku koju je poželjno detektovati i eliminisati.



## TESTIRANJE SOFTVERA

- **Validacija i verifikacija** su izrazi koji se najčešće povezuju sa testiranjem programa. Verifikacija predstavlja proveru ili testiranje objekata (ili programa) u cilju utvrđivanja koliko odgovaraju predviđenim karakteristikama.
- **Verifikacija** obuhvata analize, inspekciju, isprobavanje, a jedan od njenih oblika je i testiranje programa.





# TESTIRANJE SOTVERA

Izrazi "verifikaciju" i "validacija" se vrlo često koristi u stručnoj literaturi i odnose se na kvalitet i analize bilo kog softvera.

**Verifikacija** programa se može osloniti na tehnike automatskog dokazivanja teoreme.

Ove tehnike imaju principe deduktivnog zaključivanja, iste one koje koriste i programeri prilikom samog konstruisanja programa.

Testiranje softvera se posmatra kao validacioni proces.



## TESTIRANJE SOFTVERA

Zašto ne bi koristili iste principe u sistemu za automatsku sintezu, koji može da konstruiše program umesto da samo dokazuje njegovu ispravnost?

Svakako, konstruisanje programa zahteva više originalnosti i kreativnosti nego dokazivanje njegove ispravnosti, ali oba posla zahtevaju isti način razmišljanja.



## Testiranje nove verzije softvera koja se pušta u produkciju

Kada se nova aplikacija pušta u produkciju može se reći da je to samo početak nove faze u životnom ciklusu softvera (proizvoda).

Tu se posebno ističe termin Održavanje softvera (proizvoda).

Naravno tu se mora voditi računa o novim izmenama u sqamom softveru (proizvoda).

Postoje :

- ☐ Implementacija novih zahteva
- ☐ Prilagođavanje softvera novom okruženju (ili zadatim uslovima)
- ☐ Ispravljanje uočenih grešaka (bagova)

Ograničavajući factor je vremenški okvir jer pri svakoj izmena softvera je potrebno da se obavi kompletan ciklus testiranja

Kada se govoro o Novim verzijama softvera one mogu biti:

- neplanirane, izazvane ispravkama nedostataka (*hot fixes*)
- planirano izdavanje nove verzije
- Prestanak korišćenja softvera zahteva testiranje arhiviranja i migracije podataka



## ISTALACIONO TESTIRANJE SOFTVERA

Jedno od najbitnijih stavki - u testiranju softvera.

U literature se navodi kao **završna faza procesa testiranja**.

Istalacioni testovi se radi zajedno sa **korisnicima**. (Ne može se testirati bez prisustva samog korisnika softvera.)

Ovaj tip testiranja se radi kako bi se osiguralo da su sve mogućnosti softvera i opcije koje on pruža pravilno instalirani.

Treba proveriti da li su sve komponente aplikacija pravilno instalirane.

Poslednja provera - Da li je sve u redu sa dokumentacijom.

Sugestija:

testira instalacije softvera na različitim sistemima i u različitim situacijama (npr. prekid napajanja ili nedovoljno prostora na disku).

Uvek treba pokušati instalaciju na veoma slabim računarima koji ima manje memorije (non-compliant),



## ISTALACIONO TESTIRANJE SOFTVERA

Preporuka a i sugestija:

Ako se želi zameniti postojeći sistem, instalacija novog sistema se vrši u paralelnom modu.

### Paralelno testiranje

Paralelno testiranje se koristi tokom razvoja, kada jedna verzija softvera zamenjuje drugu ili kada novi sistem treba da zameni stari

To znači da će oba sistema raditi istovremeno u jednom periodu.

Programeri moraju da svako dnevno prate rad korisnika na ovakvom sistemu i kad se steknu uslovi (kada se postogne komforan i olakšan rad korisnika novog sistema), stari sistem se tek tada eliminiše.

Mnoge kompanije postavljaju nekoliko servera (multi - server) na jedan sistem, koji su nezavisni jedan od drugog.



## ISTALACIONO TESTIRANJE SOFTVERA

Prilikom instaliranja, sistem se konfiguriše u skladu sa okruženjem. Ukoliko je potrebno, sistem se povezuje sa spoljnim uredajima i sa njima uspostavlja komunikaciju

Jedan dobar način za instaliranje sistema pre puštanaj u takozvanu produkciju je probna instalacija na jednom odabranom serveru, posle provere dužine trajanja, instalirati ga na drugom severu.

Sve mora da se pravilno dokumentuje i da se zapiše u dnevnik testiranja softvera.

Za to je potrebno vreme i sve što se radi se mora zapisivati u dnevnik testiranja softvera.

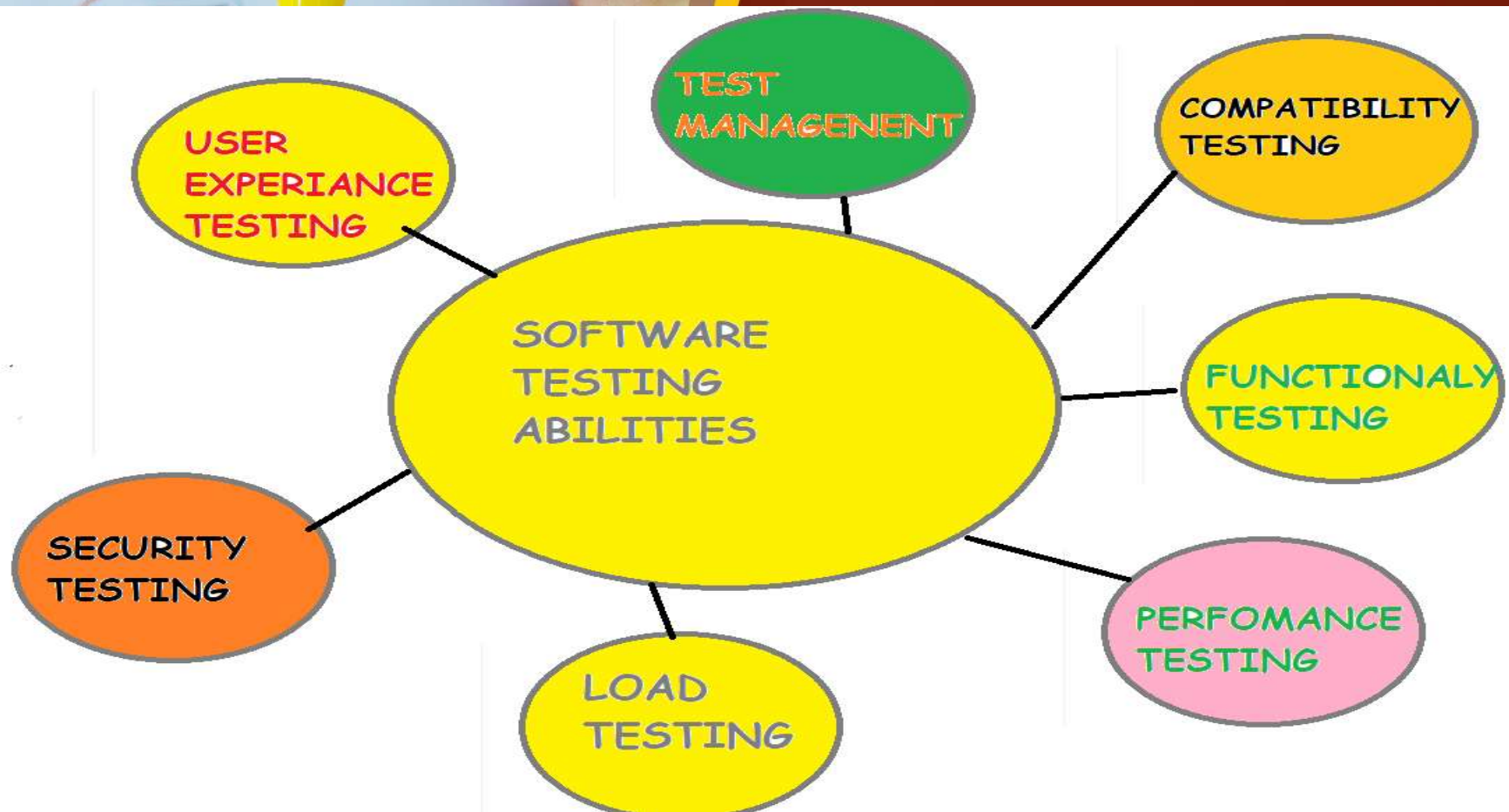


## VRSTE i Metode testiranja softvera

Zavisno od toga na koje se rizike i karakteristike proizvoda test tim mora fokusirati, može se definisati više vrsta testiranja.

- Funkcionalno testiranje softvera
- Ne funkcionalno testiranje softvera







## Funkcionalno testiranje softvera

Funkcionalno testiranje softvera obuhvata proveru da li je softver razvijen u skladu sa funkcionalnom specifikacijom.

Funkcionalno testiranje (funkcionalnost aplikacije) obuhvata testiranje na različitim nivoima, testove prihvatljivosti,

Formiraju se detaljni test scenariji i koriste se kao osnova za čitav proces funkcionalnog testiranja - najčešće BlackBox metodologijom.

Rezultat testiranja se dostavlja u formi izveštaja koji tačno navodi koje funkcije ne rade u skladu sa aplikativnim zahtevom.



## Tipovi nefunkcionalnog testiranja

- **Test opterećenja** (*load test*)
  - merenje ponašanja sistema u uslovima povećanog opterećenja (npr. broja korisnika ili količine pozvanih operacija) kako bi se utvrdilo koji stepen opterećenja sistem može da izdrži
- **Test performanse** (*performance test*)
  - merenje performanse softvera (npr. brzina obrade ili vreme odziva) za određenu funkcionalnost (obično se meri zavisno od opterećenja)
- **Test velike količine podataka** (*volume test*)
  - testiranje ponašanja sistema kada upravlja velikom količinom podataka



## TESTIRANJE OPTEREĆENJA

- To je proces testiranja softvera u kom se performanse ispituju pod određenim opterećenjem.
- Softver za testiranje opterećenja prvenstveno testira robusnost ili dostupnost softvera u normalnim i ekstremnim radnim uslovima.



# TESTIRANJE OPTEREĆENJA

Postoji puno alata koji mere performanse u uslovima opterećenja:

❑ Jmeter <https://jmeter.apache.org/>

Podržava više platformi. Napisan je u Podržava veliki broj veb protokole.

❑ Gatling <https://gatling.io/open-source/>

Napisan je u Scali, Po završetku testiranja generiše puno izveštaja.



## TESTIRANJE OPTEREĆENJA

Postoji puno alata koji mere performanse u uslovima opterećenja:

- ❑ Locust <https://locust.io/>

Prikazuje rezultate testiranja u realnom vremenu.

- ❑ Taurus <https://gettaurus.org/>

Pomoću njega testirate kod u razvoju.



## Testiranje performansi

Deo testiranja proizvoda može obuhvatiti testiranje bitnih karakteristika sistema, kao što su performanse i pouzdanost.

- ✓ Testovi treba da reflektuju profil upotrebe sistema.
- ✓ Testovi performansi obično obuhvataju planiranje serija testova kod kojih se postepeno povećava opterećenje sve dok performanse sistema ne postanu neprihvatljive.

"Stres" testiranje je oblik testiranja performansi gde se namerno testira sistem pod maksimalnim opterećenjem kako bi namerno doveli do otkaza.





## Testiranje performansi

- Kod testiranje performansi posmatraju se:
- Brzina / proverava se brzina odziva softvera
- Skalabilnost / testira se maksimalno korisničko opterećenje pod kojim softver može da rad
- Stabilnost / proverava se stabilnost softvera pod različitim opterećenjima



## Testiranje performansi

- ❑ Testiranje performansi je proces utvrđivanja brzine, odziva kao i stabilnosti kako same aplikacije tako i sistema na kome se izvršava.
- ❑ Kada se gleda na nivuo operacije razlika od jedne milisekunde je beznačajna, ali ako se ta operacija ponavlja recimo hiljda puto to već predstavlja ozbiljnu razliku.



## Testiranje performansi

- Performanse obuhvataju vreme odziva softvera, pouzdanost, upotreba resursa i skalabilnost.
- Testiranje performansi je tip testiranja koji verifikuje da se sistemski softver ponaša na odgovarajući način pod nekim očekivanim opterećenjem.
- Cilj testiranja performansi nije pronalazak novih defekata, već da se eliminišu potencijalni problemi koji utiču na performanse sistema.



## Testiranje performansi

Postoji mnogo podvrsta testiranja performansi kao što su:

- Test opterećenja (kada se opterećenje povećava u normalnim uslovima)
- Stress test (kada su opterećenja izvan granica normale)
- Spike testiranje (kad se preopterećenje povećava samo u datom trenutku), itd.

Rezultat izvršavanja ovih testova se obično daje u formi izveštaja koji analizira kapacitet servera.



## Test prihvatljivosti

Test prihvatljivosti je formalni opis ponašanje softverskog proizvoda.

Test prihvatanje generalno ima binarni rezultat, da li proizvod radi ili ne.

Testiranje prihvatljivosti ima mnogo prednosti, ali i nedostataka (skoro nikada se ne testira softver posle redovnog korišćenja, a i ne testiraju se sve funkcije interfejsa), tako da ga treba dopuniti nekom od navedenih metoda evaluacije



## Testiranje performansi

Najčešće obuhvata iskorišćenje procesorskih resursa, protok podataka i vreme odziva.

Tipični resursi koji se proveravaju su: propusni opseg, brzina procesora, iskorišćenje memorije, zauzetost prostora na disku.

U real time sistemima i ugrađenim sistemima, softver koji pruža potrebne funkcije ponekad nije prihvatljiv u skladu sa performansama.

Testiranje performansi dizajnirano je za izvođenje run-time testa performansi u softverskom okruženju jednog integrisanog sistema.

Tokom testiranja performansi, izvršavaju se testovi konfiguracije.

Testiranje performansi se pojavljuje u svim procesima testiranja.



## Testiranja performansi i testiranja opterećenja

Razlika između testiranja performansi i testiranja opterećenja je u tome što testiranje performansi utvrđuje jesu li performanse Sistema normalne, dok testiranje opterećenja proverava radni kapacitet softverske aplikacije.



A close-up photograph of a yellow pen tip pointing at a math problem on a piece of paper. The problem includes the equation  $10 + 6 =$  and the number 16. The background of the slide is a dark red gradient.

## *Peak testiranje*

Unapred definisani broj korisnika pristupa aplikaciji i koristi je neki vremenski interval, i nakon tog isteka intervala, aplikaciji pristupaju novi korisnici koji posle nekog vremenskog intervala više ne koriste aplikaciju.

Tako imate nove stres tačke sa velikim brojem korisnika.

Ideja je da se vidi da li je zadata aplikacija (koja je već opterećenja sa velikim brojem korisnika) u stanju da na neki način sa novim dodatnim korisnicima može da kvalitetno radi.



# Istraživačko testiranje softvera

(Exploratory)

- Istraživačko testiranje je softverska tehnika testiranja koja ne koristi bilo koji određeni dizajn, plan ili pristup ispitivanja.
- To je tehnika testiranja softvera u kojoj ispitivači istražuju i identificiraju različite načine ocjenjivanja i poboljšanja kvalitete softvera.



## Istraživačko testiranje softvera

Istraživačko testiranje (Exploratory testing), gde bez unapred definisanog postupka tester koristi svoju intuiciju i iskustvo u pronalaženju grešaka.

Ideja istraživačkog testiranja je da tester sami tokom testiranja aplikacije pronalaze alternativne scenarije za testiranje koji ne mogu biti unapred planirani, na ovaj način se podstiče kreativnost testera.

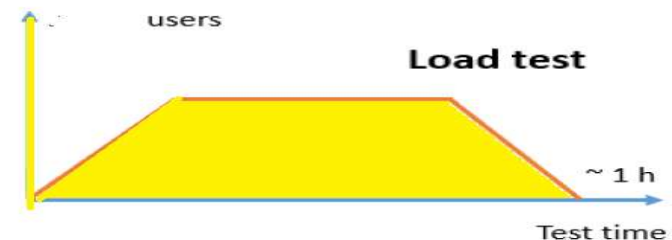
## Load testiranje

Odredi se maksimalni definisani broj korisnika koji mogu da rade na softveru (aplikaciji).

Najpre jedan korisnik koristi aplikaciju, pa drugi korisnik, pa treći korisnik i tako dalje dok ne dodjete do maksimalnog definisanog broja korisnika.

Zatim jedan po jedan korisnik napušta aplikaciju.

Ideja je da se vidi kako taj maksimalni definisani broj korisnika može da pristupi aplikaciji u isto vreme, odnosno šta će biti sa aplikacijom kada njoj pristupa taj maksimalni definisani broj korisnika u isto vreme.





## Testiranje opterećenja (Load testing)

- ✓ Testiranjem opterećenja određuju se performanse sistema u okviru realnih uslova upotrebe.
- ✓ Ovo testiranje spada u testiranje nefunkcionalnih zahteva.
- ✓ Određuje se ponašanje sistema pod normalnih opterećenjima, kao i pod najvećim očekivanim opterećenjem.
- ✓ Identifikuje se maksimalni operativni kapacitet, uska grla ako postoje i koja komponenta izaziva opadanje performansi



## Testiranje opterećenja -Load testing

- U slučaju kada se opterećenje podigne iznad razumnog nivoa, test opterećenja postaje stres test.
- Ovaj oblik testiranja najčešće se primjenjuje za klijent/server web sisteme.



## Testiranje opterećenja (Load testing)

Test opterećenja određuje:

- ✓ Maksimalni operativni kapacitet sistema
- ✓ Da li postojeća infrastruktura zadovoljava potrebe sistema
- ✓ □ Održivost sistema u slučaju povećanog korisničkog opterećenja (eng. peak load)
- ✓ Broj konkurentnih korisnika koje sistem može da izdrži





## Endurance testiranje softvera

Ova vrste testiranja softvera vrlo je slična *load* testiranju. Razika je u tome da je vreme simultanog korišćenja aplikacije od strane svih korisnika dosta duže nego kod *load* testiranja (može da bude dosta dugo od nekoliko dana ili nekoliko sati u zavisnosti za šta je namenjena aplikacija).

Ideja je da se testiraju da li svi ti korisnici mogu da pristupe aplikacije ili samo zadati unapred broj korisnika.

Kako se aplikacija ponaša sa velikim brojem korisnika (pre svega njena funkcionalnost)



## **VRSTE i Metode testiranja sofvera**

- Zavisno od toga na koje se rizike i karakteristike proizvoda test tim mora fokusirati, može se definisati više vrsta testiranja.
- Testiranje korisničkog interfejsa
- Jedinično testiranje
- Integraciono testiranje
- Sistemsko testiranje



## TESTIRANJE SOFTVERA

- *Zašto je bitno praviti izveštaje o greškama?*
  - PROBLEM AKO SE GREŠKA NE MOŽE REPRODUKOVATI,
  - KAD SE DESILA I POD KOJIM USLOVIMA
- 
- DNEVNIK TESTIRANJA