
PROGRAMMING
SOFTWARE
BUGS
TESTING
METHODS
DATABASE
USABILITY

TESTIRANJE SOFTVERA

U osnovi svih definicija testiranja programa je težnja da se odgovori na pitanje:

Da li se program ponaša onako kako je zahtevano?

U bilo kom upotrebljivom softveru u praksi postoji beskonačan broj mogućih testova koje je praktično nemoguće sve izvesti, te je nemoguće u određenom vremenskom periodu, sa ograničenim resursima izvršiti totalno testiranje koje bi otkrilo sve greške u softveru.

TESTIRANJE SOFTVERA

Razvoj i implementacija softvera je zahtevan i dugotrajan proces.

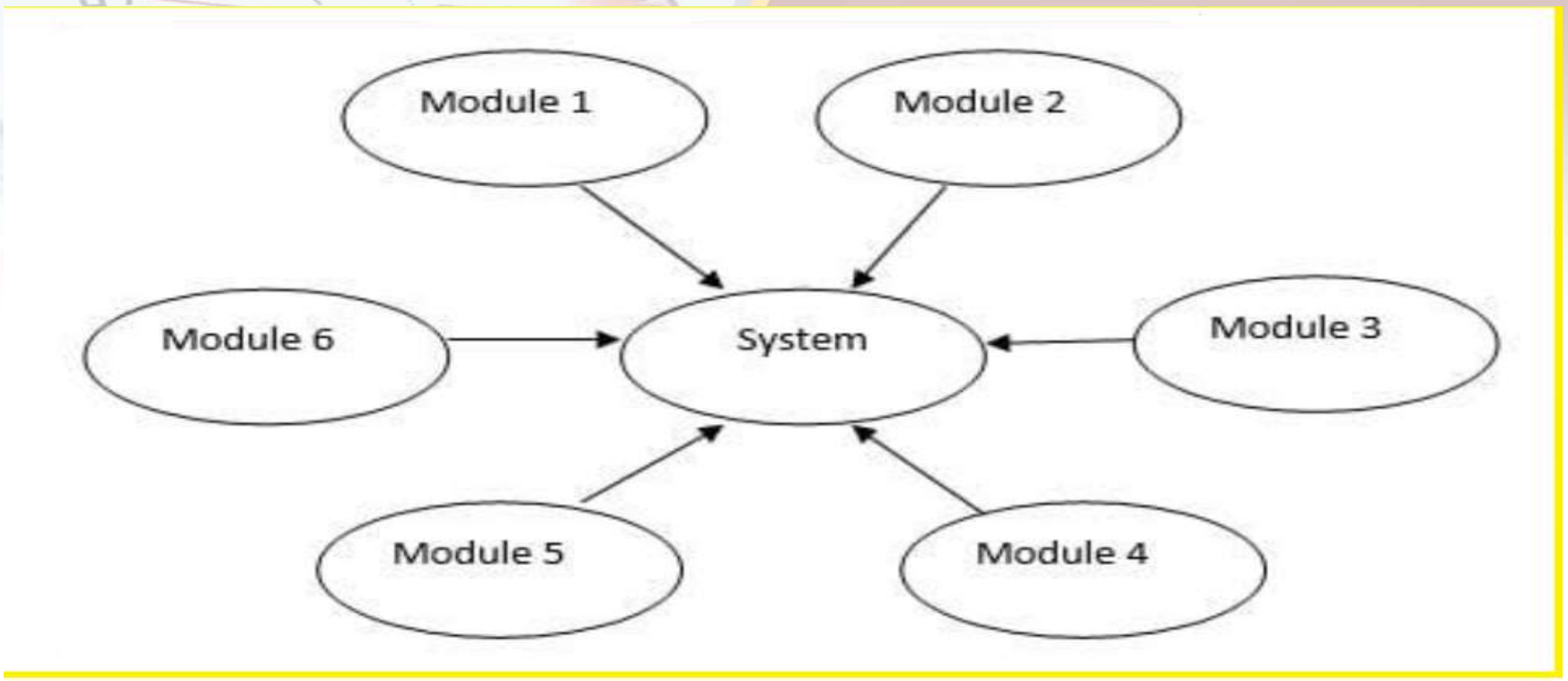
Podrazumeva korištenje raznih tehnologija i metoda.

Razvoj softvera svakim danom postaje sve kompleksniji.

Softver mora biti isporučen na vreme sa što većim kvalitetom i sa što manje grešaka.

Kao najvažniji ciljevi u Testiranju softvera ističu se prevencija, otkrivanje grešaka, zadovoljstvo krajnjeg korisnika i kvalitet samog softvera.

TESTIRANJE SOFTVERA



PROCES OTKLANJANJA BUGOVA

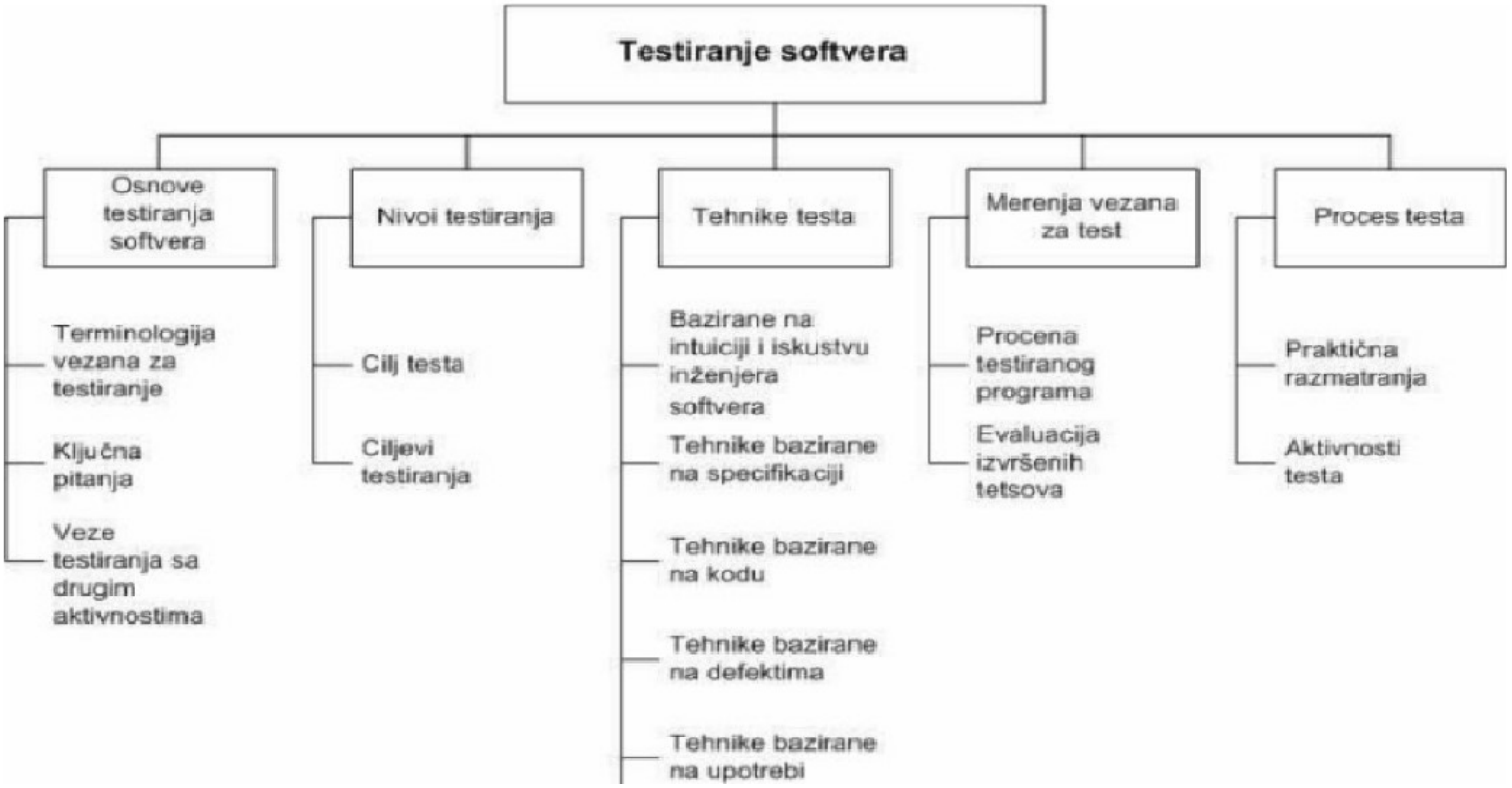
- Indentifikacija bugova
- Razumevanje bugova
- Lociranje bugova
- Ispravljanje bugova
- Ponovo testiranje svega
- VREMENSKI OKVIR + RESURSI
- TESTIRANJE I SAMO TESTIRANJE

TESTIRANJE SOFTVERA

Testiranje sistema obuhvata različite procedure, metodologije i strategije koje nude različite pristupe.

Efikasno testiranje softvera doprinosi isporuci kvalitetnog softverskog proizvoda koji zadovoljava potrebe, očekivanja i zahteve korisnika. Ukoliko radi loše, to vodi visokim troškovima održavanja i nezadovoljstvu korisnika.

TESTIRANJE SOFTVERA



Proces testiranja

- ✓ Prikupljanje nefunkcionalnih zahteva
- ✓ Izgradnja okruženja za testiranje performansi
- ✓ Programiranje slučajeva korišćenja
- ✓ Izgradnja scenarija testiranja performansi
  Sprovođenje testiranja i analiza rezultata
- ✓ Naknadna analiza rezultata, izveštavanje i sama dokumentacija

TESTIRANJE SOFTVERA

- Strategija testiranja- koja pokazuje način, tj.metod testiranja-koje tipove i koju količinu testova treba upotrebiti da bi se na najbolji način otkrile greške koje su skrivene u softveru.
- Plan testiranja- koji sadrži konkretne zadatke koji će omogućiti ostvarenje strategije testiranja.
- Slučajevi tesiranja - koji su pripremljeni u formi detaljnih primera i koji se koriste da bi se proverilo da li softver odgovara zahtevima.
- Podaci za testiranje- koji se sastoji i od ulaznih podataka i baze podataka, koji se koriste dok se izvršavaju test slučajevi,
- Okruženje testiranja- softversko i hardversko okruyenje (operativni sistem i druga softverska rešenja)u kome se celo testiranje obavlja.

IDEALNO TESTIRANJE SOFTVERA

U idealnom slučaju!!!!!

- ✓ Da koristimo što manje resursa, a da pronadjemo sto više bagova (uz što manje vremena)
- ✓ Da se naprave mali broj testova a da otkriju sve bagove
- ✓ Da ne koramo da izvršavamo testove svakodnevno
- ✓ Da svaki propušteni test daje reprezentativne rezultate
- ✓ da su testovi povezani sa Git-om ili nekim drugim alatom za verzionisanje
- .....

OVO ne postoji, samo testirati, testirati, testirati i samo testirati.

Testiranje razvoja

Sprovodi niz sinhronizovanih strategija za otkrivanje i sprečavanje defekata.

Uključuje statičku analizu koda, preglede peer kodova i analizu metrika.

Cilj je smanjiti rizik i same troškove gde god je to moguće u svakom delu razvoja sofvera.

Testiranje i samo testiranje!!!!

Kvalitet softvera

Kvalitet softvera se vrši primenom pravila, standarda, i na kraju nekih zakonskih obaveza (o kojima se mora voditi posebno računa)

- Primeri dobre prakse
  - široko prihvaćene metode rada u određenoj oblasti
- Kompanijski standardi
  - interne odredbe, procedure i smernice
  - npr. dokument o upravljanju kvalitetom, šabloni za test dokumentaciju, konvencije programskog koda

Kvalitet softvera

- Standardi upravljanja kvalitetom
  - generalni standardi koji se odnose na različite industrijske grane
  - specificiraju minimalne zahteve kvaliteta
  - ne bave se metodama implementacije kojima se kvalitet dostiže

❑ Standardi za specifičnu industrijsku granu
  - svaka industrija ima svoje standarde kvaliteta

❑ Standardi za testiranje softvera
  - odnose se na bilo koji tip softvera
  - bave se dokumentacijom i procesom testiranja

Kvalitet softvera

Test plan prema IEEE 829-2008

- Master test plan
  - sadrži generalni plan testiranja za ceo projekat
  - za svaku fazu životnog ciklusa testiranja predviđa konkretne aktivnosti, resurse i učesnike
- Level test plan
  - plan testiranja za određeni nivo testiranja
  - tako imamo *component test plan, integration test plan, system test plan*, ...
  - za svaki nivo testiranja se definiše
    - pristup, resursi i raspored aktivnosti
    - šta će biti testirano
    - konkretne aktivnosti kojima će se vršiti testiranje
    - odgovorne osobe za svaku aktivnosti

TESTIRANJE SOFTVERA

Prema QA TestLab-u postoji 7 vrsta testiranja:

- Skeniranje ranjivost (engl. Vulnerability scanning)
- Sigurnosno skeniranje (engl. Security scanning)
- Penetracijski testovi (engl. Penetration testing)
- Procena rizika (engl. Risk assessment)
- Revizija ranjivosti (engl. Security Auditing)
- Procena sigurnosnog držanja (engl. Posture Assessment)
- etičko hakiranje (engl. Ethical hacking)

TESTIRANJE SOFTVERA

Kada se vrši pravljenje nekog softvera svaka kompanija mora da testira taj softver.

Ali, i pored toga taj softver ima određene mane ili bagove.

Početna greška može da naraste tokom projektovanja a potom da se višestruko poveća tokom kodiranja.

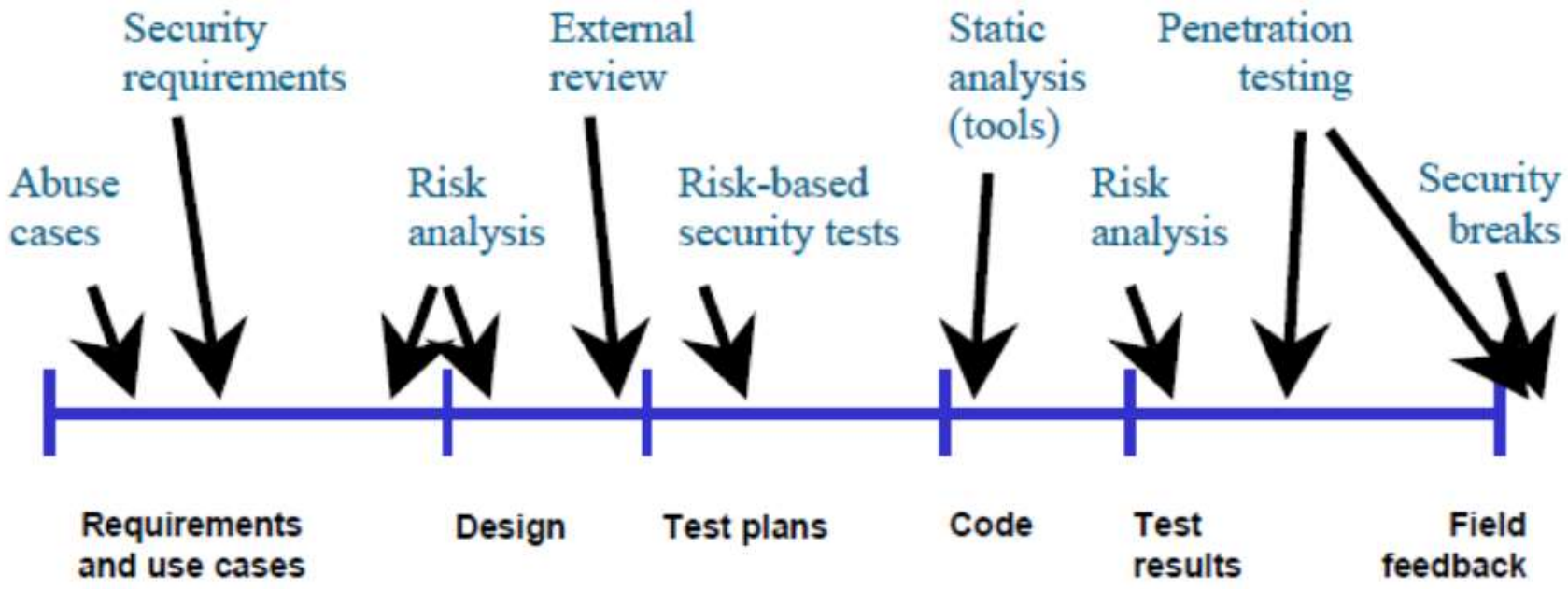
Kada se testiranjem programa utvrdi da se on ne ponaša kao što se očekuje, pristupa se analiziranju i lociranju bagova. Detaljan postupak indentifikacije bagova omogućuje se metodičkim pristupom testiranja programa.

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

Test evaluation

- Evaluacija testova (Test evaluation) predstavlja kreiranje izveštaja kojim se opisuje šta je testirano i potvrđivanje da je implementirani sistem spreman za korišćenje u skladu sa korisničkim zahtevima.
- Proces evaluacija uključuje i pregled rezultata dobijenih analizom izlaza test slučajeva.

TESTIRANJE SOFTVERA



TESTIRANJE SOFTVERA

U vezi sa testiranjem programa često se rade statičke analize (istražuju se osnovni programi, pri tome traže se osnovni problemi i prikupljaju podaci bez izvršavanja programa) i dinamičke analize (istražuje se ponašanje programa).

Postoji mnogo pristupa softverskom testiranju, koji teže kontroli kvaliteta procesa testiranja da bi obezbedili kvalitetne informacije o kvalitetu testiranja softvera.

Iako je većina istraživanih testova koncentrisana na pronalaženje efektivne tehnike testiranja, takođe je veoma bitno napraviti softver koji se lako testira.

Softver je lak za testiranje ukoliko greške uzrokuju rušenje programa, pa prema tome te greške će biti uočavane tokom samog testiranja.

Kriterijum početka testiranja

Kada se govori o testiranju softvera neki preduslovi moraju biti ispunjeni da bi se se uopšte moglo izvršiti testiranje.

- Nije dobro testirati softver koji nije spreman (resursi)

- Kriterijumi početka testiranja
  - Test alati su spremni za korišćenje
  - Test okruženje mora biti spremno
  - Test objekti postoje i nalaze se u takvom stanju razvoja da ih ima smisla testirati
  - Test podaci (ulazni) su dostupni

Kriterijum za završetak testiranja

Kada treba prestati sa testiranjem?
Dva su odgovora na ovo pitanje:

- Nikad. Korisnik nastavlja da testira nakon preuzimanja programa.
- Kada se ostane bez resursa za dalje testiranje (vremenski okvir /budžet)

Potrebno vreme testiranja bi mogao biti relevantan parametar
na osnovu kojega bi se donela odluka kada prestati sa testiranjem softvera.

TESTIRANJE SOFTVERA

Proizvođači softvera bi želeli da predvide broj grešaka u softverskim sistemima pre nego što ih primene, kako bi mogli da procene kvalitet kupljenog proizvoda i teškoće koje se javljaju u samom procesu održavanja.

Testiranje softvera je veoma kompleksan proces.

Cilj testiranje je da se ustanovi da li se softver ponaša na način kako se to od njega očekuje.

Prema tome, primarni cilj testiranja softvera je otkrivanje gešaka.

TESTIRANJE SOFTVERA

Aktivnost testiranja pokazuje da li je dati softver usklađen sa specifikacijom.

Specifikacija je ključna stvar u testiranju.

Stoga, kako se sakupljaju rezultati testiranja, pojavljuju se dokazi o nivou kvaliteta i pouzdanosti programa.

Ako testiranje često otkriva važne greške, kvalitet i pouzdanost programa mogu se smatratiti nedovoljnim i zahteva se dalje testiranje. Sa druge strane ako su greške minorne i lake za ispravljanje, onda je nivo kvaliteta i pouzdanosti programa prihvatljiv.

Testiranje nikada ne može da definitivno kaže da li je program ispravan, jer neotkrivene greške mogu ostati u programu čak i posle najobimnijeg testiranja.

TESTIRANJE SOFTVERA

Stoga je uobičajeno gledište po kome je uspešan test onaj koji ne otkrije nijednu grešku netačno, što naglašavaju navedeni ciljevi testiranja a to su:

- ✓ Testiranje je proces izvršavanja programa u nameri da se nađu greške
- ✓ Dobra test stavka ima visoku mogućnost pokrivanja greške
- ✓ Uspešan test otkriva dotle neotkrivenu grešku.

TESTIRANJE SOFTVERA

Uspeh jednog skupa test podataka je jednak uspešnom izvršenju iscrpnog testiranja programa. Jedno od velikih pitanja koje se javlja kod testiranja programa jeste reprodukcija greške (testeri otkrivaju greške a programeri otklanjaju bagove).

Očigledno je da mora postojati koordinacija između testera i programera.

Reprodukcija greške je slučaj kada bi bilo najbolje da se problematični test izvede ponovo i da znamo kad, i tačno gde se greška desila.

Dakle idealnog testa i idealnog proizvoda nema.

TESTIRANJE SOFTVERA

U poslednje vreme ulaže se dosta napora da se konstruktivni pristup primeni parcijalno, tj. na određene delove programa, ako već ne može u celini.

Jedan od pristupa je matematički dokaz korektnosti zasnovan na radovima Dijkstre, Hoare-a, Floyda.

Izvođenja u formalnim teorijama predstavljaju opšti okvir za razvoj deduktivnih metoda utvrđivanja korektnosti programa. Iz tog okvira izviru dve osnovne metode (pobijanje pridružene predikatske formule, odnosno korišćenje pravila programske logike) kao i njihove modifikacije. Rad sa formulom koja je pridružena datom programu podrazumeva prisustvo dodatnih aksioma, bez kojih pobijanje nije ostvarljivo.

TESTIRANJE SOFTVERA

Druga tehnika dokazivanja je zasnovana na pojmu simboličnog izvršenja.

Programske linije se obrađuju istim redom kao što će biti izvršavane, za razliku od Dijkstrinog redosleda "unatrag".

Obe tehnike su jednako snažne i logički ekvivalentne.

Jedna od najstarijih i najpoznatijih metoda koja se primenjuje za konstruktivno testiranje manjih programa je simboličko izvršavanje programa.

Jedan od načina njene primene je putem računara što sa druge strane nije osnovni uslov.

Samo donekle podseća na metodu simulacije rada računara, ali se od nje bitno razlikuje po cilju.

Osnovni cilj ove metode je konstruktivno testiranje.

TESTIRANJE SOFTVERA

1. Beskonačno i detaljno testiranje softvera nije moguće – (vremenski rokovi i budžet)
2. Testiranje povezanih grešaka (Defect clustering) - grupiranje grešaka odnosno (u teoriji) da mali broj modula sadrži većinu otkrivenih nedostataka.
3. Povezanost grešaka u modulima (Pesticide paradox)
4. Testiranjem dokazujemo postojanost grešaka
5. Velika zabluda je da ne postoje greške u softveru
6. Testiranje u ranoj fazi razvoja
7. Pristup testiranja prema vrsti projekta

TESTIRANJE SOFTVERA

Proces testiranja ima cilj da potvrdi funkcionalne zahteve, pouzdanost, otpornost na greške i ostale faktore kvaliteta softvera.

Testiranje softvera je aktivnost koja je važna u celom procesu razvoja softvera.

Testiranje softvera koristi određenu strategiju za izbor testova koji su izvodljivi za raspoloživo vreme i resurse.

Opadanje kvaliteta softvera – kvalitet softvera će opadati ukoliko se ne modifikuje.

Planiranje testiranja

Za process testiranja softvera vrši se pravljenje različitih planova za sam process testiranja.

Planovi obuhvataju sledeće aktivnosti:

- Ko radi testiranje
- Šta se testira
- Kada se vrši testiranje
- Kako se vrši testiranje
- Da li postoje neke dodatne aktivnosti koje su neophodne da bi testiranje bilo urađeno
- Ko obezebeđuje te dodatne aktivnosti

TESTIRANJE SOFTVERA

- ✓ Plan testiranje se isključivo vezuje za određeni projekat
- ✓ Test plan je povezan sa specifikacijom (detaljno opisuje koje tehnike testiranja se koristi, vreme testiranje, role, neophodni alati….)
- ✓ Analizira rizika (šta se dešava ako test ne prođe)
- ✓ Šta nije testirano

TESTIRANJE SOFTVERA

Razvojno okruženje - je okruženje u kom se projekat razvija i u kom se ukljanjaju pronađeni bug-ovi.

Programer je odgovoran da konfiguriše sve potrebne promene u okruženju da bi funkcija radila na očekivani način.

Test okruženje– odlučuje pod kojim će uslovima softvera i hardvera projekat biti testiran.

Production okruženje– je okruženje u kom je softver puštena za krajnje korisnike.

TESTIRANJE SOFTVERA

Kontinuirano testiranje, koje se izvodi istovremeno sa procesom razvoja, je efikasna tehnika testiranja koja se sprovodi radi automatizacije procesa testiranja. Pomaže u postizanju više ciljeva kao što su:

- Pronalaženje grešaka u ranijim fazama,
- Automatsko izvršavanje testova da bi se otkrili problemi,
- Smenjeno vreme za isporuku softvera,
- Smanjenje poslovnog rizika,
- Obezbeđivanje kontinuiranog procesa isporuke softvera,
- Poboljšava stopu izdavanja verzije softvera.

TESTIRANJE SOFTVERA

Metodologije testiranja softvera su razne strategije ili pristupi koji se koriste za testiranje softvera kako bi se obezbedilo da se softver ponaša i izgleda onako kako se očekuje.

Ne možemo znati da li u nekom delu softvera postoji neka greška ako ga ne pokrenemo i ako ga ne testiramo.

TESTIRANJE SOFTVERA

- Testiranje softvera (dinamika,konačnost, selektivnost, očekivanost)
- Veza testiranja sa drugim aktivnostima razvoja softvera.
- Nivoi testiranja
- Predmet testiranja.
- Ciljevi testiranja softvera
- Tehnike testiranja
- Kombinovanje tehnika
- Proces testiranja softvera
- Evaluacija programa koji se testiraju.
- Evaluacija testova

TESTIRANJE SOFTVERA

PONOVNA UPOTREBA TEST INFORMACIJA

Kad imamo dat tip grešaka, možemo odabrati metod testiranja (funkcionalni ili strukturni) koji će najverovatnije otkriti greške tog tipa. Ovde se postavlja pitanje kada prekinuti testiranje? Evo nekih mogućih odgovora:

- Kad nestane vremena,
- Kad dalje testiranje ne izaziva nove otkaze,
- Kad dalje testiranje ne otkriva nove greške,
- Kad ne može da se smisli nijedna nova test stavka,
- Kad se postigne zahtevana pokrivenost,
- Kad su sve greške uklonjene.

TESTIRANJE SOFTVERA

TESTIRANJE TOKA PODATAKA

Testiranje toka podataka odnosi se na oblike strukturnog testiranja koji se usmeravaju na tačke u kojima promenljive dobijaju vrednosti i na tačke u kojima se te vrednosti koriste (ili pominju).

Testiranje toka podataka služi kao "provera u stvarnosti" za testiranje putanja; zaista, mnogi zagovornici i istraživači testiranja toka podataka smatraju ovaj pristup za oblik testiranja putanje.

Tu postoje dva glavna oblika testiranja toka podataka: jedan pruža skup osnovnih definicija i jedinstvenu strukturu mera pokrivenosti testa, a drugi se zasniva na konceptu "programskog odreska".

Oba formalizuju intuitivno ponašanje (i analizu) testera, i mada oba počinju od programskog grafa, oba se vraćaju u pravcu funkcionalnog testiranja.

TESTIRANJE SOFTVERA

Promenljive koje predstavljaju podatke nekako zadobijaju vrednosti, i te vrednosti se koriste za izračunavanje vrednosti drugih promenljivih.

Još od početka šezdesetih godina, programeri analiziraju početne programe u smislu tačaka (iskaza) u kojima promenljive dobijaju vrednost i tačaka u kojima se te vrednosti koriste.

Te analize su se često zasnivale na usklađenosti spisaka broja iskaza u kojima se pojavljuju imena promenljivih. Rane analize toka bile su usmerene na skup grešaka koje su danas poznate kao anomalije definisanja/pozivanja:

•

TESTIRANJE SOFTVERA

- Promenljiva koja se definiše, ali se nikada ne koristi (niti pominje)
- Promenljiva koja se koristi, ali se nikada ne definiše
- Promenljiva koja se definiše dva puta pre nego što se upotrebi.

Svaka od ovih anomalija može se otkriti na osnovu usklađenosti programa odnosno ove anomalije se mogu otkriti onim što sa naziva "statička analiza" tj. nalaženje grešaka u programu bez njegovog izvršenja.

TESTIRANJE SOFTVERA

Test plan je u suštini dokument koji u sebi sadrži podatke koji su vezani za planiranja testiranja softvera

- Podaci a vezani za akciju aktivnosti planiranja su:
  - Izbor i način strategije testiranja
  - Izbor test okruženja i načina automatizacije testova
  - Izbor nivoa testiranja i veza između njih
  - Integracija testiranja sa drugim aktivnostima u okviru projekta
  - Načina evaluacije rezultata testiranja
  - Metrike za kriterijum završetka testiranja

NEKi nedostaci u procesa testiranja (Shortcoming of the testing process)

Nedostaci u procesu testiranja utiču da mnoge greške ostanu ne otkrivene prilikom testiranja.

Ovi nedostaci nastaju zbog sledećih uzroka:

Nepotpuni testni plan.

Nepostoji dokumentovanje otkrivenih grešaka.

Odlaganje ispravljanja otkrivenih softverskih nedostataka.

Nepotpuna korekcija ili detekcija greška zbog nemarnosti ili vremenskih ograničenja.

Refaktoring (Refactoring)

Ključna strategija u postizanju kardinalnih pravila softverske evolucije je refaktoring (eng. refactoring) koji je Martin Fowler naveo da je to "promena interne strukture softvera da bi bio lakši za razumevanje i jeftiniji za modificiranje bez promene njegovog ponašanja" (Fowler).

Ponekad se kod drastično promeni za vreme održavanja, a ponekad kod nije dovoljno dobar nakon samog procesa kodiranja.

Refaktoring (Refactoring)

Refaktoring je postupak poboljšanja softverskog programa tako što se radi interno restruktuiranje koda bez izmene ili dodavanja funkcionalnosti.

- ✓ Razvoj podrazumeva postojanje testova. Prosto, nema izmene koda ukoliko nema testa.
- ✓ Razvoj podrazumeva da se kod menja od prvog koraka.
- ✓ Da bi dodalo nešto novo u sledećem koraku.
- ✓ TESTIRATI U SVAKOM KORAKU

Fazi testiranje

Tehniku je konstruisao prof. Barton Miller sa Univerziteta u Wisconsinu 1988.

Osnovna ideja testiranja fazi metodom je generisanje slučajnih ili pseudo-slučajnih podataka kao ulaz u sistem.

Ukoliko sistem prestane da funkcioniše nakon nekog ulaza onda se taj niz podataka zapamti za dalju analizu.

Fazi testiranje

U literaturi se koristi termin negativno testiranje Negativno testiranje predstavlja efikasnu metodu za otkrivanje sigurnosnih grešaka.

Ova vrsta testiranja softvera predstavlja:

- automatizovano
- delimično automatizovano testiranje

Pre svega se onosi na granične slučajeva u softveru,

Ulazni podaci se biraju:

- nasumično,
- delimično nasumično
- ne validni podaci
- Delimično ne validni podaci

Fazi testiranje

- fazer (fuz-zer).

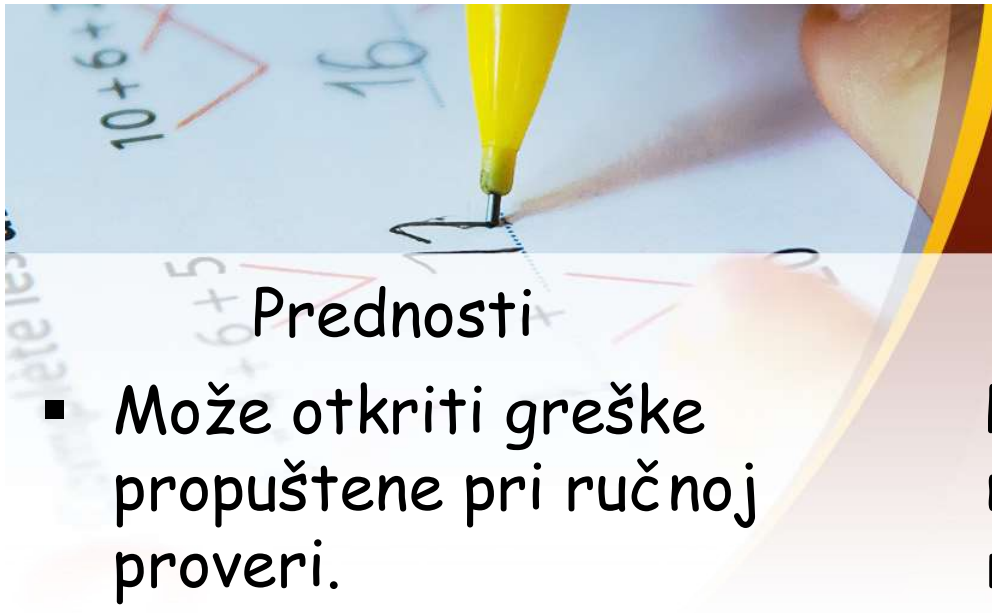
Ovde se isistira na velikoj automatizaciji i testovima.

Ne zahteva mnogo ekspertskog vremena.

Testiranje se sastoji od generisanja test slučajeva ulaznih podataka i praćenja izvršenja testiranog softvera u toku obrade.

Postoji više različitih pristupa faz testiranju.

Jedna od mana ove vrste testiranja je mogućnost da ima više false nego pozitivnih rezultata.



Fazi testiranje

Prednosti

- Može otkriti greške propuštene pri ručnoj proveri.
- Daje opštu sliku robusnosti ciljnog softvera.

Mane

Programi sa složenim ulazima mogu zahtevati mnogo više rada da bi se proizveo dovoljno dobar fazi test da bi se postigla što veća pokrivenost koda.

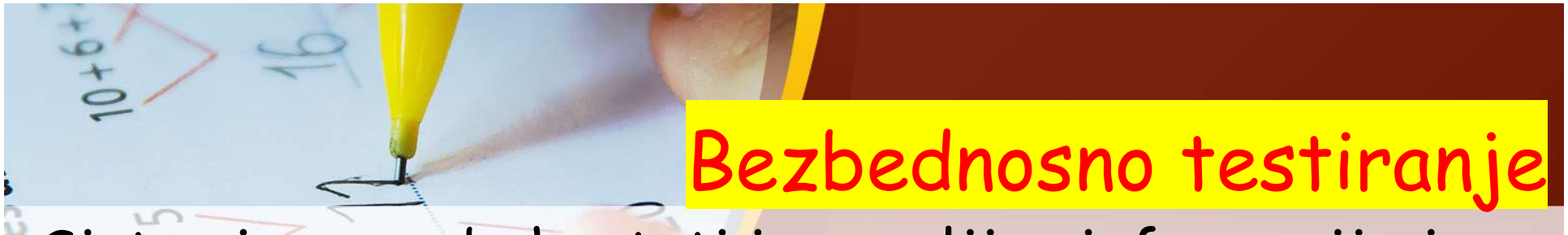
Continuous integration

Neprekidna integracija ili Continuous integration je praksa čestog spajanja (merge) radnih verzija koda u toku razvojnog procesa.

- ✓ Postupak pisanja koda se deli na manje cikluse.
- ✓ Sofverski alati za neprekidnu integraciju.

BENEFIT:

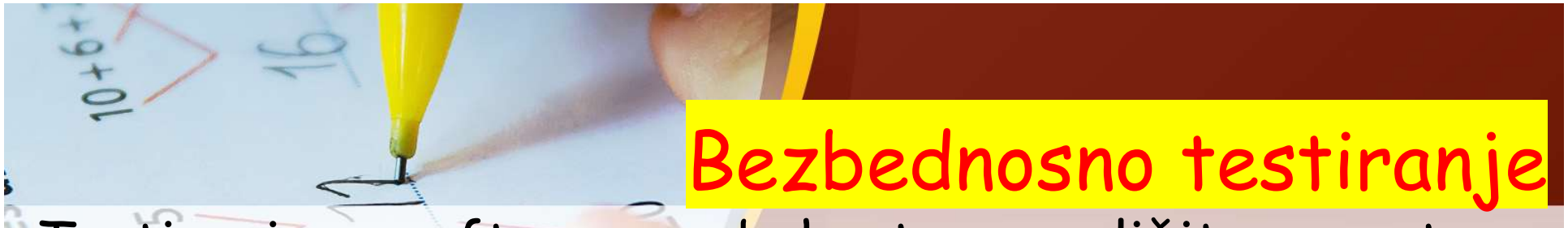
Stalna raspoloživost kompletnog izgrađenog softvera.



Bezbednosno testiranje

Sistemi mogu obuhvatati i poverljive informacije i time su potencijalno mete neovlašćenog korištenja. Ono može uključivati:

- Pokušaj proboja u sistem sa strane sa ciljem zabave ili koristi.
- Nanošenje lične štete legalnim korisnicima sistema ili ljudima čiji se podaci nalaze u sistemu.



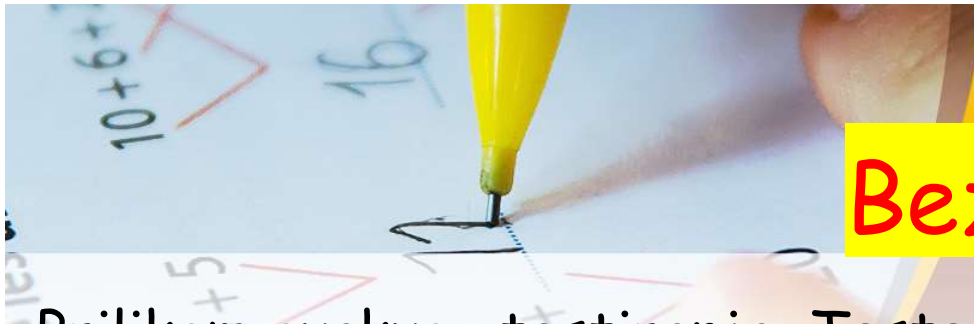
Bezbednosno testiranje

Testiranje softvera obuhvata različite vrste testiranja kako bi se obezbedilo da softver neće imati funkcionalne i nefunkcionalne nedostatke.

Bezbednosno testiranje podrazumeva izvršavanje nefunkcionalnih tipova testova kako bi se otkrili sigurnosni nedostaci.

Ono se radi kako bi se sprečili različiti hakerski napadi.

Aplikacije koje se nalaze na Internetu.



Bezbednosno testiranje

Prilikom ovakvog testiranja, Tester (ili testierski tim) pokušava da se stavi u ulogu sa "druge strane to jest osoba koja pokušava da nelegano uđe u u sistem."

Metode koje se mogu koristiti su:

- Pokušaj da se dođe do lozinke
- Korišćenje specijalizovanog softvera za napad na sistem
- Prebukiranje sistema zahtevima
- Otkrivanje grešaka u sistemu i njihovo korištenje za napad
- Upad do neobezbeđenih podataka (ako ih ima)

U literature se može naći :
"Uz dovoljno velike materijalne i vremenske resurse svaki se sistem može uspešno napasti."