
PROGRAMMING
SOFTWARE
BUGS
TESTING
METHODS
DATABASE
USABILITY

TESTIRANJE SOTFVERA

"Testiranje programa je proces izvršavanja programa i upoređivanje opserviranih ponašanja prema željenom ponašanju. "

"Primarni cilj testiranja je otkrivanje grešaka u softveru sa sekundarnim ciljem građenja samopouzdanja kod testera u slučaju kada test ne otkriva grešku ". / Myers G.

TESTIRANJA SOFTVERA

- Programi se mogu testirati u raznim fazama razvoja i različitim stepenom strogosti.
- Kao i svaki deo razvoja testiranje zahteva rad. Testiranje programa je jedna od najskupljih pa, prema tome, i najvažnijih aktivnosti u toku njegovog životnog ciklusa.
- Sprovodi se, kako u toku prvobitne realizacije tako i u fazi eksploatacije i održavanja i to prilikom svake modifikacije programa.

TESTIRANJE SOFTVERA

Konflikt između ova dva cilja se uočava kada proces testiranja ne otkriva grešku.

U odsustvu drugih informacija, ovo može značiti ili da je softver visokog kvaliteta ili da je veoma niskog kvaliteta.

Postavljeno davne 1989 godine.

[Mye,1989] *Myers G. The Art of Software Testing, još 1989 godine.*

TESTIRANJE SOFTVERA

Direktni ciljevi:

- Identifikacija i otkrivanje grešaka koliko je moguće pre produkcije.
- Postizanje što većeg nivoa kvaliteta softvera

Indirektni ciljevi :

- Skupljanje informacija o softverskim greškama

(da bi se one ispravile u novim verzijama softvera)

TESTIRANJE SOFTVERA

Da bi se izvelo testiranje velikih i složenih programa ono se mora izvesti što sistematičnije, kako bi se ostvarila.

Ovo, pak, znači da od svih metoda za testiranje jedina koja se *ne sme* primenjivati je ''ad hoc'' metoda testiranje jer ona ne može da utvrdi kvalitet i ispravnost, ni prema specifikaciji, ni prema konstrukciji, niti prema primeni.

STRATEGIJA TESTIRANJA SOFTVERA

Strategija testiranja predstavlja celokupan pristup testiranju, pri čemu se definišu nivoi testiranja, metode, tehnike i alati.

Ukoliko bi se radilo u idealnim uslovima, strategija testiranja bila bi zajednička za celu organizaciju, kao i za sve programe unutar nje.

Primena i razvoj strategije vrlo su bitni za postizanje određenog kvaliteta programa a samim tim i realizaciju projekta.

TESTIRANJE SOFTVERA

Planiranje testiranja trebalo bi počne sa ranom fazom procesa uzimanja zahteva.

To znači da test planovi i procedure moraju biti sistematski i kontinualno razvijani i po potrebi redefinisani.

Pravi stav prema kvalitetu je prevencija, mnogo je bolje izbeći probleme nego ih ispravljati.

Zbog svega navedenog, jasno je zašto je softversko testiranje tesno povezano sa drugim oblastima softverskog inženjerstva

STRATEGIJA TESTIRANJA SOFTVERA

- Prvi korak pri izboru strategije testiranja programa jeste

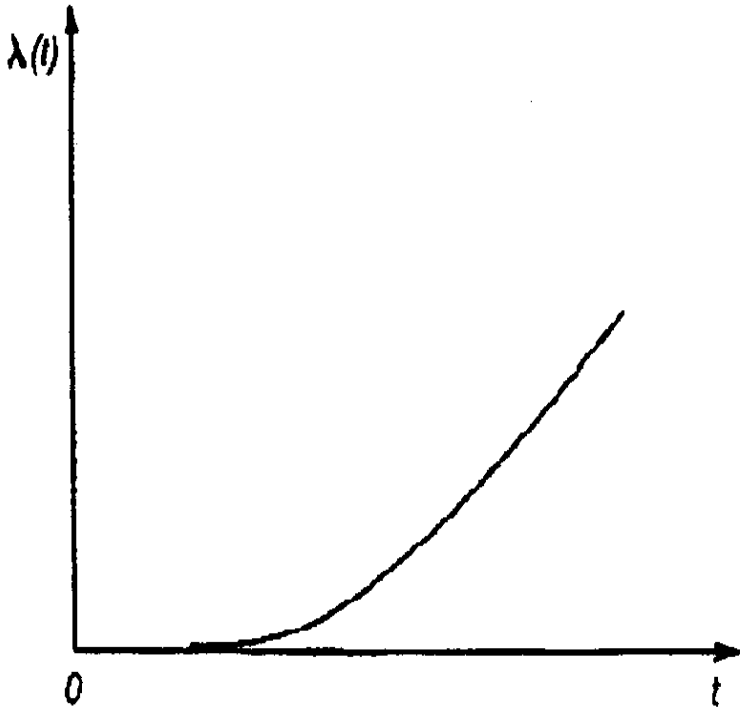
Specifikacija.

OPŠTI MODEL RASPODELE GREŠAKA U SOFTVERU

Postoji takozvana Vejbulova raspodela grešaka koja se može podeliti:

- Testiranje pokazuje kako će se program ponašati u eksploataciji.
- Sve greške imaju istu verovatnoću pojavljivanja
- Greške su nezavisne jedna od druge.
- U početku samog procesa testiranja postoji fiksan broj grešaka u softveru tj. statična pretpostavka generisanja grešaka.
- Vremenska raspodela otkrivanja grešaka podleže Vejbulovoj raspodeli.
- Broj otkrivenih grešaka u nekom vremenskom intervalu je nezavisan od broja otkrivenih grešaka u drugom vremenskom intervalu.

OTKAZI



U praksi, otkazi u zoni tzv. negativnih vremena označavali bi otkaze koji su se desili pre nego što je sistem pušten u korišćenje.
Za svaki parametar su definisane granice ispravnog rada

Intezitet otkaza
Dele se na slučajne i sistemske.

Nezavisno testiranje

Prednosti

- Nezavisni testeri su nezavisni i nepristrasni
  - Mogu da pronađu više nedostataka (greške, bagovi) nego sami programeri
  - Imaju drugačije principe i nalaze drugačije nedostatke nego programeri

Najčešće programer implicitno podrazumeva određeno (stanje, ponašanje) programa

- Testerski tim ima neke druge principe i ne polazi od istih pretpostavki (ili sam tester, ako nije u testerskom timu)
- Ne znači da nema svoje pretpostavke i ponašanje koje on implicitno podrazumeva
- Implicitno razumevanje testera je bliže nego razumevanju krajnjeg korisnika

TESTIRANJE SOFTVERA

7 PRINCIPA TESTIRANJA SOFTVERA

Prvi princip

Svaka aplikacija ili proizvod pušta se u produkciju nakon "dovoljnih metoda i tehnika testiranja" od strane različitih timova.

Da li je treba prestati sa TESTIRANJEM SOFTVERA ikada ?

NIKAD NE TREBA PRESTATI!

Testiranje softvera se ne može dokazati da softver ne sadrži greške ili nedostatke. Postoje 2 slučaja:

- Ako test ne prolazi, to znači da nedostatak postoji
- Ako svi testovi prolaze, to i dalje ne dokazuje da nedostataka nema

Što više testiramo softver smanjuje se verovatnoća da određeni nedostaci postoje, a to ne znači da nema nedostataka u samom softveru.

Drugi princip

Takozvano "Potpuno testiranje je nemoguće"

Testiranjem se identifikuje što je više moguće nedostataka. Test se smatra uspešnim ukoliko je ponašanje sistema pri njegovom izvršavanju u skladu sa korisničkim zahtevima.

- ✓ Za sve ulazne vrednosti (nije moguće da se izvrše i kreiraju svi testovi koji bi pokrili sve kombinacije ulaznih vrednost zajedno sa kombinacijama svih različitih preduslova).
- ✓ U zavisnosti od procenta celokupnog izvornog koda kome se može pristupiti, pokrivenost koda testovima može biti ograničena.
- ✓ Može se reći da Test slučaj samo uzorak iz skupa mogućih podataka
  - ✓ potrebno je pronaći podskup mogućih test slučajeva tako da u granicama dostupnih resursa obezbede što veću verovatnoću ispravnosti programa

Treći princip

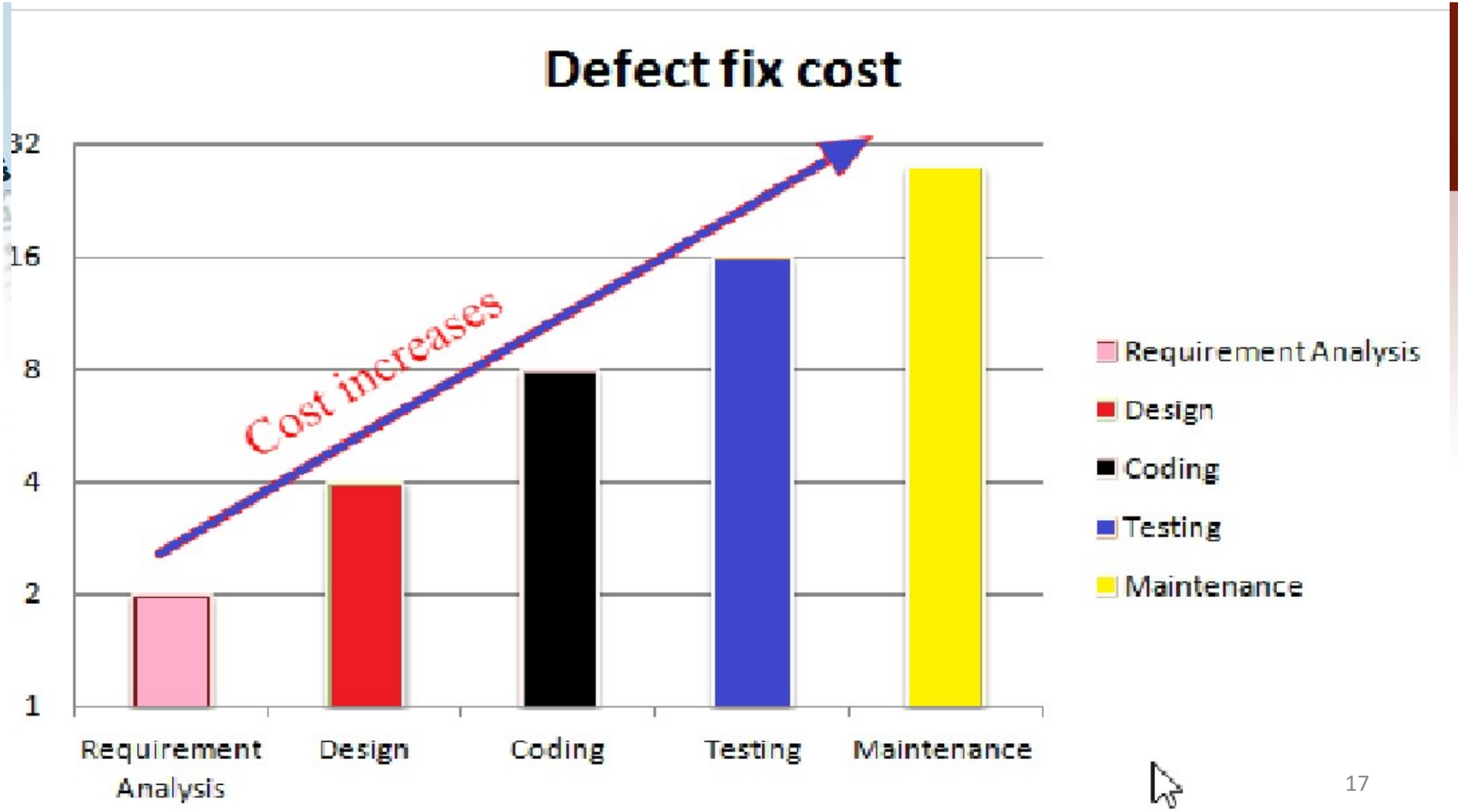
Jedna od najvažnijih osobina Testiranja softvera je rano otkrivanje grešaka.

PREPORUKA:

Rano Testiranje /Testeri se moraju uključiti u ranoj fazi životnog ciklusa razvoja softvera.

- od samog starta dizajna softvera treba uračunati i testiranje i uključiti ga u sve faze razvoja softvera
- *Rano otkrivene greške se lakše otklanjaju.*

Vrlo je bitan trenutak kada se testovi pripremaju. Što se ranije testovi pripreme to će se bolje razumeti zahtevi projekta. Ta rana priprema testova pomaže da se problemi otkriju i preduprede. Rano pripremljeni testovi povećaju kasniju efikasnost.


Defect fix cost
32
16
8
4
2
1
Cost increases
Requirement Analysis
Design
Coding
Testing
Maintenance
Requirement Analysis
Design
Coding
Testing
Maintenance

Četvrti princip

- Grupisanje nedostataka

Grupisanje i sistematizovanje podataka. Treba analizirati i dati proporciju izmedju broja očekivanih i pronađenih defekata po modulu.

- ✓ Moduli koji sadrže najviše defekata su najkritičniji i sadrže implementaciju bitnih funkcionalnosti sistema.
  - ➢ Nedostaci nisu ravnomerno raspoređeni po programu
  - ➢ Većina nedostataka se nalazi u pojedinim delovima test objekta
  - ➢ Ako se u određenom delu programa pronađu nedostaci, testirati i dalje jer najverovatnije postoji mogućnost da postoje još u tom delu programa

Peti princip

- ***Paradoks pesticida***
  - Kao što insekti postaju otporni na pesticide, tako da ukoliko više puta testove stalno ponavljamo u svakoj iteraciji testiranja, ti testovi više neće moći da otkriju nove defekte. Dakle, ako ponavljamo istu seriju testova, metoda testiranja softvera beskorisna je za otkrivanje novih nedostataka.
  - ❑ Skup test slučajeva treba redovno pregledati i revidirati
    - novi nedostaci se verovatno nalaze u delovima koda i scenarijima koje postojeći test slučajevi ne pokrivaju

Šesti princip

Testiranje zavisi od konteksta

Testiranje se prilagođava funkcionalnostima sistema

- Testiranje mora biti prilagođeno nameni aplikacije, okolnostima u kojima se koristi i mogućem riziku koji otkaz izaziva
- Sugestija (a neki smatraju da je to i pravilo)
- Različiti sistemi se ne mogu testirati na isti način
  - strategija testiranja, intenzitet testiranja, kriterijum završetka itd. moraju biti prilagođeni kontekstu konkretnog sistema

Sedmi princip

Pogrešna je ideja ili pretpostavka je da je sistem bez nedostataka upotrebljiv.

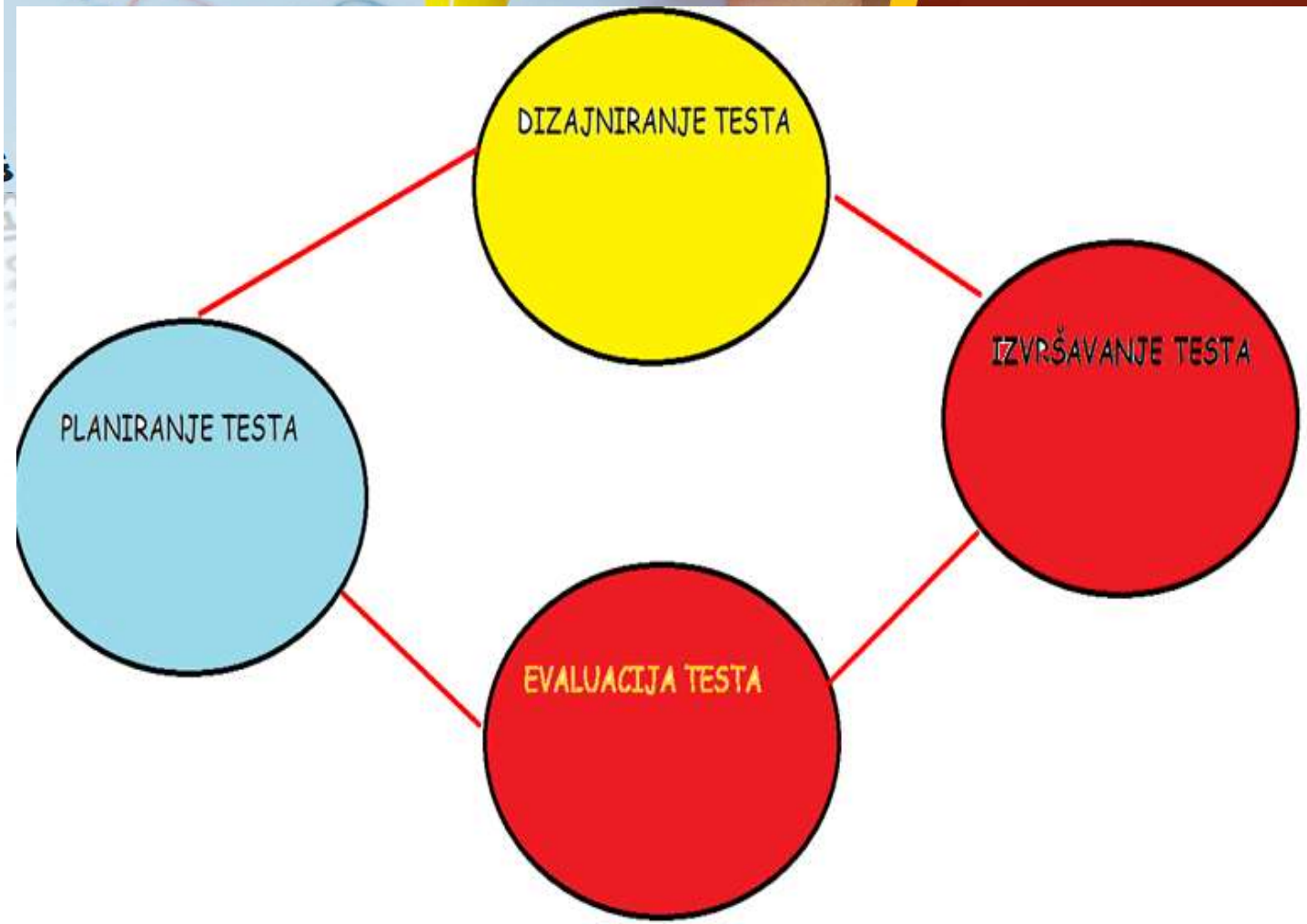
Važno je da se dobiju optimalni rezultati testa (testiranjem softvera) bez odstupanja od testnog cilja.

- Odsustvo grešaka ne garantuje da sistem radi kako treba – pronalaženje i ispravljanje defekata ne pomaže ako je sistem neupotrebljiv ili ako ne ispunjava zahteve korisnika
- Korisnik ne očekuje samo ispravan sistem nego i sistem koji je vizuelno i funkcionalno za njega intuitivan
  - intuitivnost je dosta subjektivna kategorija, pa ju je teško ugraditi u specifikaciju zahteva
  - treba uvesti korisnika u rane faze razvoja aplikacije i izradom ranih prototipa dobiti korisne povratne informacije u pogledu njegovih očekivanja

PROTOTIP

Prototip je (po literature ili nekim autorima) ogledna verzija ili simulacija samog (IT) proizvoda.

- ✓ Kada su od strane korisnika samo uopšteno definisani ciljevi razvoja proizvoda, (ali ne i detalji u pogledu ulaza, procedura i izlaza)
- ✓ Preporuka: Simulirati rad softvera da bi korisnik mogao videte kako će budući softverski proizvod raditi
- ✓ Testiranjem prototipova zajedno sa krajnjim korisnicima.
  Ishod: Tada UX timovi mogu poboljšati korisničko iskustvo.
- ✓ Kada kompanija želi proveriti efikasnost algoritama ili samog sistema.
- ✓ Greške u zahtevima i dizajnu otkrivaju se u ranoj fazi.


DIZAJNIRANJE TESTA
PLANIRANJE TESTA
IZVRŠAVANJE TESTA
EVALUACIJA TESTA

STRATEGIJA TESTIRANJA SOFTVERA

Specifikacija predstavlja osnovnu komponentu svih definicija.

Programi mogu biti veoma jednostavni ili složeni, dok specifikacija (zavisno od veličine i primenjenih programerskih metoda) može da bude u obliku jednog dokumenta ili da predstavlja složenu hijerarhiju dokumenata.

Ukoliko se posmatraju različite hijerarhije programa specifikacija može se zaključiti da se one sastoje od tri ili više nivoa dokumenata.

STRATEGIJA TESTIRANJA SOFTVERA

Uočiti razliku između:

Specifikacije programa (određuje šta program treba da radi, a često i na koji način),

Specifikacije arhitekture (prikazuje arhitekturu. programa, tj. komponente unutar programa i njihove odnose) i

Detalje specifikacije programa (opisuje kako se primenjuje svaka komponenta programa).

Ukoliko su specifikacije u hijerarhijskim odnosima, program se može testirati u različitim fazama razvoja.

STRATEGIJA TESTIRANJA SOFTVERA

Na osnovu hijerarhije programa specifikacije, evidentni su različiti nivoi testiranja:

- Testiranje jedinice (podrazumeva testiranje svake jedinice kao osnovne komponente u cilju određivanja korektnosti)
- Testiranje integracije programa korak po korak (sa sve većim i većim komponenata programa koje se integrišu ali i testiranjem sve do onog momenta kada funkcioniše kao celina)

TESTIRANJE INTEGRACIJE PROGRAMA

Cilj je :

Pronaći nedostatke u komunikaciji i konflikte između komponenti koje se integrišu.

Moguće je integrisati sve komponente odjednom i onda tek pokrenuti testiranje.

Po nekim autorima - Ovo nije baš dobar pristup za velike sisteme zato što se greška (ako postoji) teško pronalazi, ali je dobar za male sisteme, koji imaju mali broj komponenti.

TESTIRANJE INTEGRACIJE PROGRAMA

Cilj je da se nedostaci u integraciji pronađu što je ranije moguće.

Cilj je da se integracioni test sprovede čim se komponente integrišu.

- ✓ Ako postoje da se nedostaci lakše lociraju
  - ✓ pouzdano se zna da se nedostatak nalazi u komunikaciji komponenti čija se integracija testira

TESTIRANJE INTEGRACIJE PROGRAMA

Testiranje integracije bitan je aspekt testiranja softvera.

Cilj integracionog testiranja je da se moduli, koji su jedinično testirani, integrišu, zatim da se pronađu greške, da se te greške uklone i da se izgradi celokupna struktura sistema, kao što je predviđeno dizajnom.

TESTIRANJE INTEGRACIJE PROGRAMA

Integraciono testiranje (ponegde se zove integracija i testiranje i piše se kao oznaka -I&T) je faza u testiranju softvera u kojoj se pojedinačne komponente kombinuju i testiraju kao grupa.

Aplikacija se obično sastoji od nekoliko softverskih modula.

Ove module kodiraju različiti programeri.

Ali kako da znamo da li moduli dobro funkcionišu zajedno?

TESTIRANJE INTEGRACIJE PROGRAMA

- Integracijsko testiranje obično se odvija nakon jediničnog testiranja, koje uključuje testiranje pojedinačnih modula i jedinica.
- Nakon što se utvrdi da svaka jedinica radi zasebno, testiranje integracije procenjuje kako sve jedinice rade u kombinaciji.

TESTIRANJE INTEGRACIJE PROGRAMA

Testiranje integracije je u suštini inkrementacioni proces, koji obično zahteva od testera da integrišu module jedan po jedan i da vrše testiranje u svakom koraku.

Može da se koristiti i neki dodatni alati za praćenje komunikacije između komponenti

- npr. *browser* sadrži alat za prikaz podataka koji se razmenjuju koristeći HTTP protokol

TESTIRANJE INTEGRACIJE PROGRAMA

Neki od razloga zašto se testiranje integracije danas koristi su:

- Različiti programeri koriste različitu logiku kada razvijaju module čak i za istu softversku aplikaciju.
- Integracijsko testiranje je jedini način da se zasebni moduli rade zajedno kako se od njih (potencijalno) očekuje .

TESTIRANJE INTEGRACIJE PROGRAMA

- Mora se voditi računa o podacima u različitim modulima to jest da se vodi računa o struktura tih podataka. (ko ih menja, kako ih menja, vreme menjanje, postavljanje rokova…)
- Testiranje integracije je neophodno kako bi se obezbedilo da softverske aplikacije sa više modula rade zajedno kako se očekuje, da ispunjavaju zahteve korisnika i da se pridržavaju tehničkih specifikacija postavljenih na početku samog projekta.

TESTIRANJE INTEGRACIJE PROGRAMA

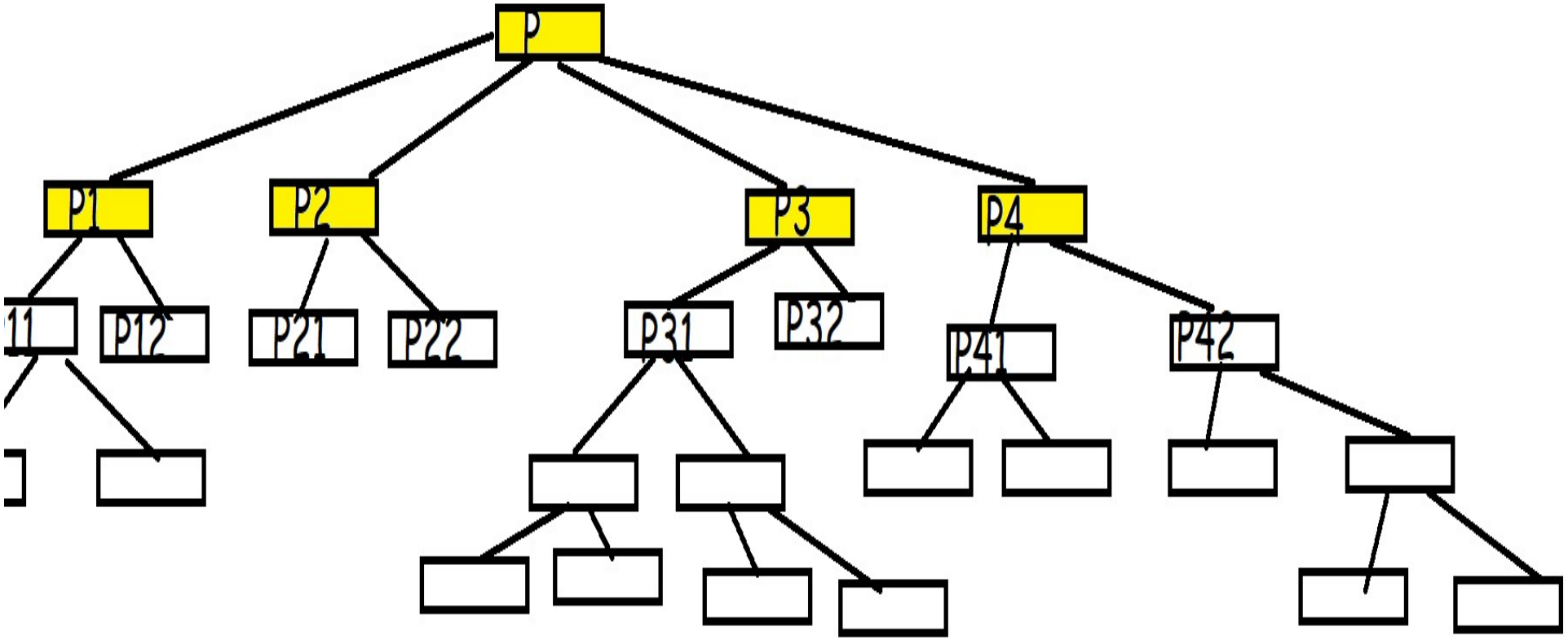
- Integracijsko testiranje zahteva mnogo resursa
- Mogu uključivati izvođenje nekoliko različitih testova istovremeno
- Testiranje integracije može biti složen proces, posebno kada se testira integracija mnogih različitih sistema, (zatim baze podataka, različita progrmska okruženja, …)
- Testiranje integracije zahteva pre svega vreme
- Jedan od najtežih situacija je faza rešavanja problema koji se pojave tokom ovog načina testiranja.

TESTIRANJE INTEGRACIJE PROGRAMA

Testiranje integracije može se izvršiti:

- ✓ Prvi pristup:Primenom pristupa odozgo nadole. (Bottom-up testing)
- ✓ Drugi pristup: Pristup odozdo prema gore, koji se izvodi sa dna kontrolnog toka. (Top-down testing)
- ✓ Treći pristup je Mešovita (Sandwich testing) integracija

TESTIRANJE INTEGRACIJE PROGRAMA



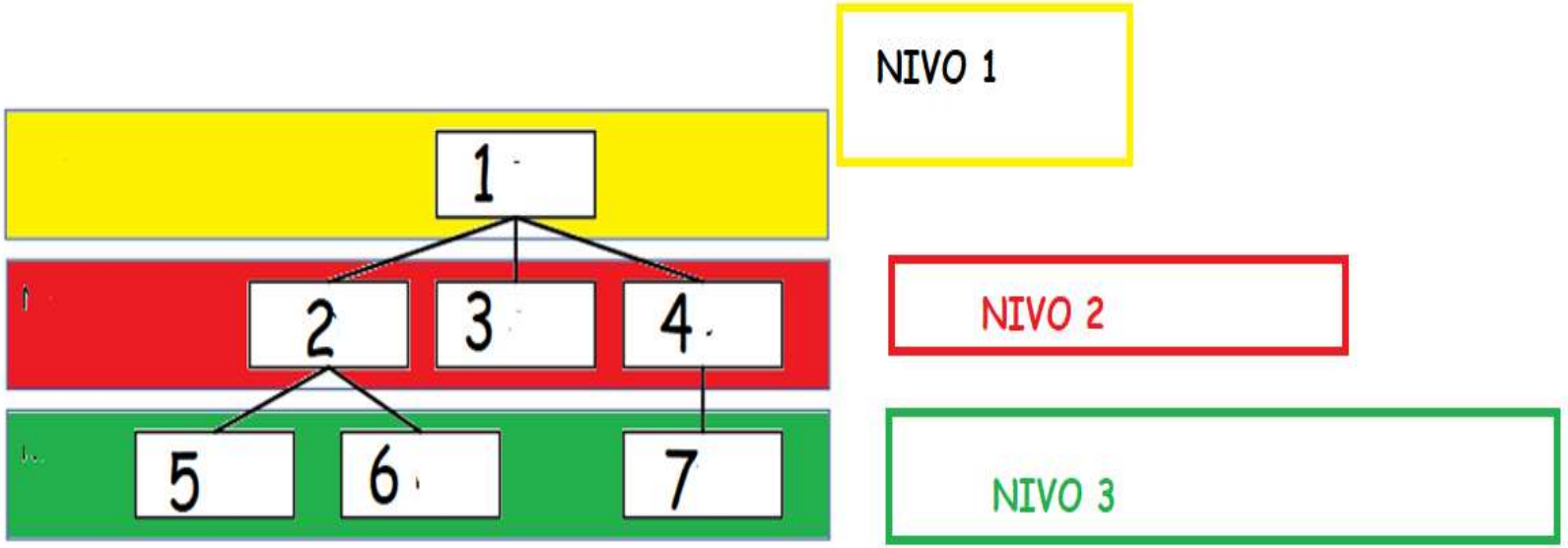
TESTIRANJE INTEGRACIJE PROGRAMA

Integracija od vrha ka dnu.
Prepoznaju se dve podstrategije u dubinu ili u širinu.
Moduli se analiziraju od vrha naa kraju.
U *top-down* testiranju, prvo se testira glavni (main) modul tj. modul na najvišem nivou u softverskoj strukturi; zadnji moduli koji se testiraju su moduli na najnižem nivou.
Glavni modul se koristi kao početni za sve podmodule.
2. Ubacuju se moduli jedan po jedan.
3. Testiranje se vrši posle ubacivanja svakog novog modula.
4. Primenjuje se i regresiono testiranje (obuhvata neka ili sva prethodna testiranja) da bi se proverilo da li dolazi do novih grešaka
Njihovo rano otkrivanje je vrlo bitno.

TESTIRANJE INTEGRACIJE PROGRAMA

- Integracija odole na gore
  - Integracija kreće sa komponentama na dnu hijerarhije
  - Komponente na nižem nivou hijerarhije koje još nisu implementirane se simuliraju
  - Kako se niže komponente implementiraju tako se verifikuje njihov rad stavljanjem u test umesto simuliranih objekata
  - prvo se testiraju moduli na najnižem nivou, a glavni (main) modul se testira zadnji.

Mešovita (Sandwich testing)

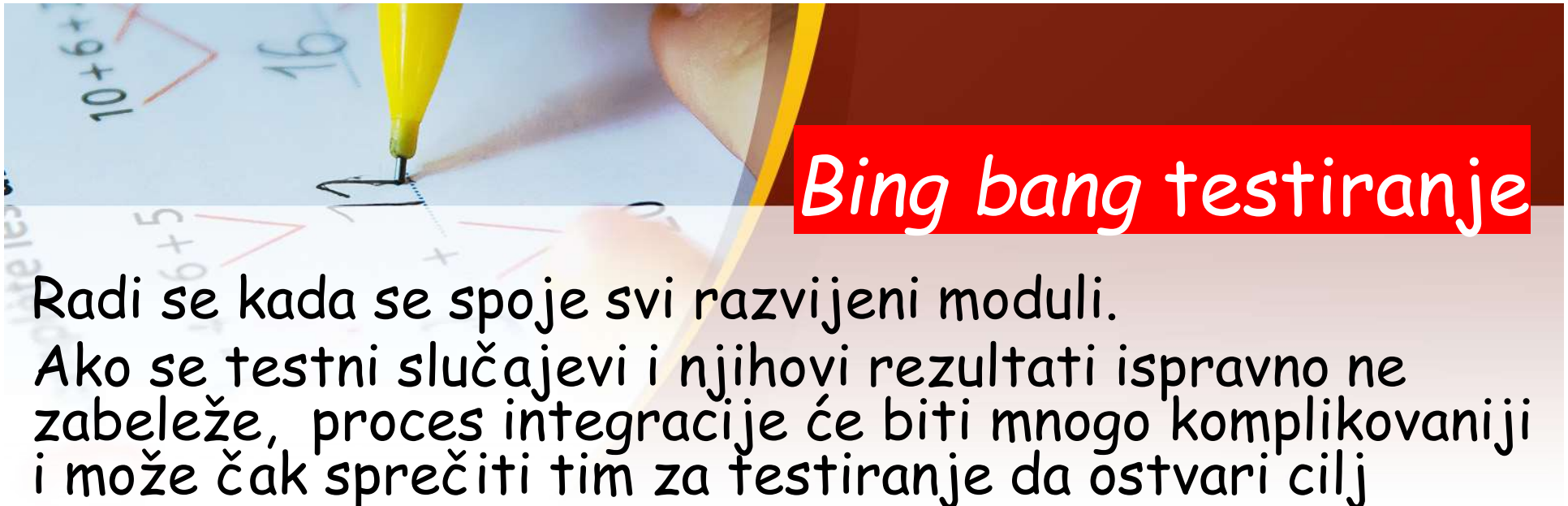


Mešovita (Sandwich testing)

- Kombinacija prva dva pristupa (od vrha ka dnu i od dna ka vrhu)
- Test primeri se relativno mogu lako uraditi (za testiranje)
- Komponente se mogu integrisati čim se implementiraju

Mana:

- Greške se teže otkrivaju



Bing bang testiranje

Radi se kada se spoje svi razvijeni moduli.

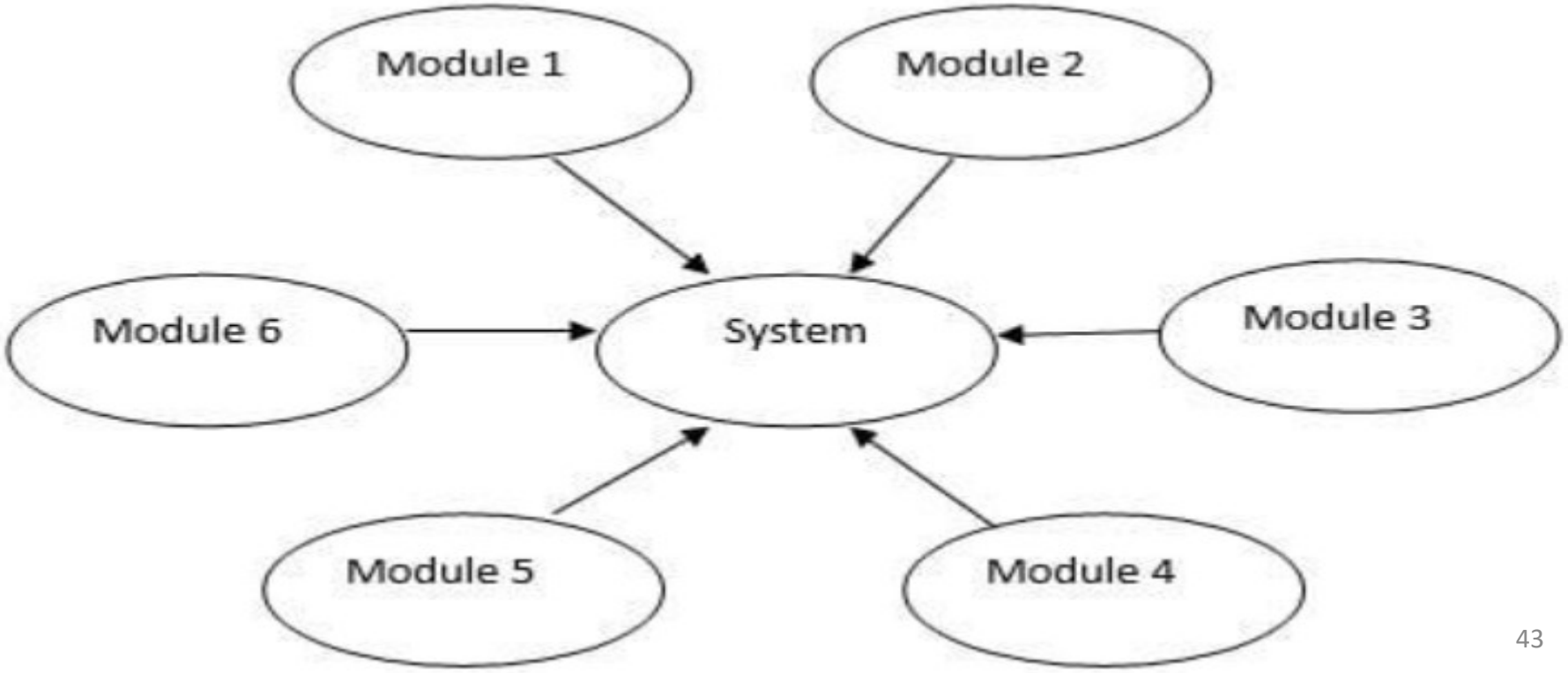
Ako se testni slučajevi i njihovi rezultati ispravno ne zabeleže, proces integracije će biti mnogo komplikovaniji i može čak sprečiti tim za testiranje da ostvari cilj testiranja integracije.

Jedan tip Big Bang testiranja zove se :

Usage Model testiranje.

Usage Model testiranje se može koristiti u testiranju i softvera i hardvera

Bing bang testiranje



STRATEGIJA TESTIRANJA SOFTVERA

- Sistemsko testiranje (System testing) (program se integriše u proizvod koji se može nazvati "konačan proizvod" i potom se vrši testiranje da se utvrdi da li su ispunjeni svi zahtevi iz specifikacije)
- Takozvano Testiranje prihvatanja (gde se testira prihvatanje gotovog programa i često se koristi podskup sistemskih testova)

STRATEGIJA TESTIRANJA SOFTVERA

Strategija testiranja predstavlja celokupan pristup testiranju, pri čemu se definišu nivoi testiranja, metode, tehnike i alati.

Ukoliko bi se radilo u idealnim uslovima, strategija testiranja bila bi zajednička za celu kompaniju, kao i za sve programe koje se koriste u okviru same kompanije. Primena i razvoj strategije vrlo su bitni za postizanje određenog kvaliteta programa a samim tim i realizaciju projekta.

SISTEMSKO TESTIRANJE SOFTVERA

Ispituje se da li je ponašanje sistema u skladu sa specifikacijom zadatom od strane klijenta.

Ova vrsta testiranja stavlja naglasak na nefunkcionalne zahteve sistema kao što su brzina, efikasnost, otpornost na otkaze, uklapanje u okruženje u kojem će se sistem koristiti.

Testiranje sistema obavlja se u drugačijim uslovima u odnosu na testiranje jedinica koda i testiranje prihvatljivosti.

U proces testiranja sistema uključuje se ceo razvojni tim pod nadzorom menadžera (rukovodioca) projekta.

SISTEMSKO TESTIRANJE SOFTVERA

- Nakon što se u toku sistemskog testiranja uoče i isprave većina defekata (nikad se mogu otkriti svi), sistem je spreman za isporuku klijentu.
- Sistemsko testiranje je obično dobro za poređenje sistema sa nefunkcionlanim sistemskim zahtevima, kao što su bezbednost, brzina, preciznost ili pouzdanost.
- Klijent onda najčešće inicira Acceptance Testing, odnosno User Acceptance Testing (UAT).

User Acceptance Testing (UAT).

Testirajte softver sa stvarnim korisnicima.

Bez sveobuhvatnog testiranja prihvatljivosti korisnika, tražene promene mogu izgledati gotove kada je stvarni rad još uvek neophodan.

Jedina dostupna metoda za prepoznavanje takvih problema je nakon implementacije kada je klijenti pokušavaju da sami otkriju neke defekte.

Korisnički test (User Acceptance Test)

Acceptance test

Koristi se za indentifaciju nekih bugova u ovoj fazi testiranja softvera.

Koristi se kada naručilac i korisnik nisu iste osobe.

Krajnji korisnik verifikuje da li je sistem upotrebljiv ili nije.

Vrlo često najteži test jer:

Strah od novog i nepoznatog

- Krajnji korisnik ima najmanju sposobnost i interes da svoja očekivanja i zahteve izrazi i formalizuje
- Bez prihvatanja od strane krajnjeg korisnika sistem će teško biti uveden u upotrebu
- Test se najčešće vrši u kasnoj fazi razvoja kada je svaka izmena komplikovana i skupa.

Test prihvatanja od strane korisnika (UAT)

- Test prihvatanja od strane korisnika (User acceptance testing ili Acceptance testing) je testiranje koje vrši klijent kada je sistem spreman za isporuku, nakon što se ispravila većina defekata pronađena u sistemskoj fazi testiranja.
- Cilj ovog testiranja je da klijent stekne poverenje u sistem koji je implementiran.
- Proveravaju se funkcionalnosti prema specifikaciji zahteva i određuje se da li sistem ispunjava potrebe krajnjih korisnika.

Test prihvatanja od strane korisnika (UAT)

- ✓ Korisničko iskustvo
- ✓ Korisničko testiranje je faza u procesu testiranja u kojoj korisnici testiraju sve ulazne vrednosti i daju neka svoja zapažanja.
- ✓ Korisničko testiranje je veoma bitno, čak i kada je Testiranje sistema i softvera već urađeno.
- ✓ Korisnikovo radno okruženje ima velike efekte na pouzdanost, performanse, upotrebljivost i robustnost sistema.

Test prihvatanja od strane korisnika

User Acceptance Testing

Test prihvatanja od strane korisnika je obavezan zbog:

- ✓ U slučaju da su developeri pogrešno protumačili zahteve i samim tim pogrešno implementirali funkcionalnosti
- ✓ Izmene u samoj specifikaciji ne budu iskomunicirane sa developerima kako treba na pravi način, mogu izazvati velike probleme (posebno ako imamo vremenske rokove i probleme u samom budžetu projekta)

TEST prihvatanja od strane korisnika

User Acceptance Testing (UAT).

Nepisano pravilo:

Potrebe samog korisnika: Treba obratiti pažnja da je akcenat dat na određenom nivou kvaliteta softvera, a ne samo na funkcionalnosti.

Preporuka: Korisnici treba da isprobaju različite uređaje kako bi bili sigurni da se test slučajevi ponašaju isto.

To je faza u procesu testiranja u kojoj sami korisnici aktivno učestvuju.

Potencijalni problem: Odabir tima za testiranje: Tim za testiranje sastoji se od krajnjih korisnika koji vrše testiranje u realnom vremenu i realnom okruženju.

Izvođenje testnih slučajeva i sve se mora dokumentovati.

UAT je jedan od poslednjih i (mnogi autori navode kao) jedan od najkritičnijih postupaka softverskog projekta. On se mora testirati pre nego što se novo razvijeni softver - bude pušten u produkciju.

Alfa testiranje

Alfa testiranje je vrsta testiranja prihvatljivosti softvera.

Ono se koristi kako bi se identifikovali svi mogući problemi pre objavljivanja proizvoda.

Fokus ovog testiranja je simuliranje rada stvarnih korisnika.

Tada se testiraju sve opcije u softveru koje tipični korisnik može uraditi u stavrnom okruženju i samom načinu rada.

ALFA I BETA TESTIRANJE SOFTVERA

Alfa testiranje se izvršava u stvarnom okruženju i obično su testeri interni članovi.

Ova vrsta testiranja naziva se alfa jer se koristi relativno rano, pri kraju razvoja programskog rešenja i pre beta testiranja.

Po nekom nepisanom pravilu, alfa tesiranje se vrši u najranijoj fazi razvoja softvera, međutim, u nekim se slučajevima može primeniti na završenu ili blizu završetka sofvera.

Beta testiranje obavljaju stvarni korisnici programskog rešenja u "stvarnom okruženju" i može se smatrati oblikom eksternog korisničkog testiranja prihvatljivosti.

Drugi naziv Field Testing – beta testera

Beta verzija programskog rešenja objavljuje se sa ograničenim brojem krajnjih korisnika kako bi se dobila povratna informacija o kvalitetu samog projekta (proizvoda.) . To je konačni test pre slanja, (postavljanja na web sajt) ka krajnim korisnicima

ALFA I BETA TESTIRANJE SOFTVERA

Pitanje koje se često postavlja je :

Da li je sigurno istalirati neku beta verziju softvera ?

Ispitivanje softvera je još uvek u razvoju.

Nepisano pravilo: Alfa testiranje je prva faza testiranja softvera nakon razvoja.

Beta testiranje se vrši nakon što softver prođe alfa testiranje. Beta verzija softvera se isporučuje korisnicima, koji ga instaliraju i koriste u realnim uslovima.

Obično se beta testiranje vrši u dve faze: zatvoreni beta test i otvoren beta test.

ALFA I BETA TESTIRANJE SOFTVERA

- Cilj: Testirati sve delove softvera,
  - Pronaći sto je više grešaka pre same produkcije,a to je i promarni cilj testiranja softvera
- Softver dobijaju stvarni korisnici pre produkcije.
- Korisnici u toku beta perioda mogu testirati aplikaciju i slati feedback,
- Beta testeri često pronađu greške koje nisu do tada primećene (naročito iz korisničke perspektive), i moguće je popraviti te probleme pre produkcije. (ako su te primedbe opravdane)
- Što se više ovakvih grešaka ispravi, veći je kvalitet softvera nakon produkcije
- Kranji cilj: Zadovoljstvo korisnika (customer satisfaction)

Buddy testiranje

Buddy testiranje je tehnika testiranja na kojem su angažovana obično dva člana tima koji rade na istom programskom rešenju na istom uređaju.

Jedan od članova tima će raditi sa raznim scenarijima da li nešto radi ili ne a drugi će sve to unositi u dnevnik testiranja.

Jedan član tima trebao bi biti tester, a drugi član razvojnog tima može biti analitičar.

Faze u procesu testiranja prihvatljivosti

- Definisanje kriterijuma prihvatljivosti
- Plan testiranja prihvatljivosti
- Izvođenje testova prihvatljivosti
- Pokretanje testova prihvatljivosti
- Analiza rezultata testova
- Odbijanje/prihvatanje sistema

PLAN TESTIRANJA

- Plan testiranja prihvatanja koji podrazumeva plan testiranja prihvatljivosti programa. Ukoliko plan testiranja ne predstavlja poseban dokument, on je povezan sa planom sistemskog testiranja.
- Plan sistemskog testiranja koji podrazumeva plan integrisanja i testiranja sistema. Uključen je u plan testiranja prihvatljivosti ukoliko ne predstavlja poseban dokument.

Evaluacija rezultata Testiranja softvera

- Na kraju neke faze testiranja radi se evaluacija rezultata testiranja.
- U ovoj fazi treba definisati mere (metrike) za evaluaciju kvaliteta testiranja i realizovati Izveštaj o proceni kvaliteta testiranja.
- U ovoj fazi se može uraditi i analiza pronađenih defekata sa ciljem da se ukaže na česte i zajedničke greške razvojnog dela tima, da se preporuče naknadne aktivnosti (kursevi, doobuka korisnika, promena tehnologije, ...).
-

SISTEMSKO TESTIRANJE SOFTVERA

- end to end test

Sistemsko testiranje bazira se na testiranju kompletnog softvera, da bi se proverilo da li sve komponente rade zajedno kao celina (end to end test), to znači da se softver testira u realnom sistemu korišćenja.

Sistemski testovi testiraju potpuno integrisani sistem da bi se potvrdilo da sistem ispunjava sve zahteve korisnika.

SISTEMSKO TESTIRANJE SOFTVERA

- Cilj je testirati funkcionalne i nefunkcionalne zahteve.
- Treba utvrditi u kojim delovima sistem netačno, nepotpuno ili nekonzistentno zadovoljava zahteve

Takođe, treba utvrditi i zahteve koji su:

- Neevidentirani
- zaboravljeni u samoj specifikaciji zahteva. (u praksi se može desiti da takvih zahteva ima jako puno)

SISTEMSKO TESTIRANJE SOFTVERA

Testiranje različitih aspekata ponašanja sistema:

- Stabilnost - u trenucima neočekivanog povećanja opterećenja,
- Pouzdanost - kada je sistem u recovery ili failure modu, bezbednost - od potencijalnog hakovanja sistema performanse - vremenski odziv sistema kada je opterećen različitim nivoima opterećenja.

Bilo koje od ovih karakteristika mogu da se testiraju istovremeno sa drugim karakteristikama ili svaka posebno.

STRATEGIJA TESTIRANJA SOFTVERA

- Konstruisanje testova
- Veoma je bitno da se pre primene softvera osmisle testovi kako bi se testiranje softvera izvršilo prema unapred definisanim pravilima.
- Nakon primene testova (u oviru svakog nivoa) vrši se procena rezultata tih testova.
- Ukoliko se detektuje neki potencijalni problem, ili se na testovima izvrši revizija pre ponovljene primene ili se izvrše određene izmene programa testiranje se mora ponavljati.

STRATEGIJA TESTIRANJA SOFTVERA

Svaki naredni nivo podrazumeva da se u predhodnim koracima ili nivoima eliminisao potencijaloni problem i da on ne postoji više.

Tokom rada, softver zahteva stalno održavanje što podrazumeva ponavljanje testiranja kao i njihovu prilagođavanje i proširivanje.

Ponovna primena testova koji su se pokazali kao uspešni koriste se u cilju utvrđivanja eventualnih posledica koje se mogu javiti usled izvršene promene softvera.(dodatni zahtevi)

STRATEGIJA TESTIRANJA SOFTVERA

Testovi na najvišem nivou se razrađuju tako što se odredi da li je i koliko softver verodostojan u primeni zahteva iz specifikacije.

Na nižim novima,testovi se razrađuju u pravcu provere da li testovi zadatog softvera uključuju sve zahteve (detalje) specifikacije i njegove arhitekture

Podrazumeva se da je potrebno i zvanično i nezvanično proveravanje svake faze razvoja testova.

Testovi se razvijaju u onolikoj meri u kojoj se razvijaju i programi.

STRATEGIJA TESTIRANJA SOFTVERA

Posle specifikacije sledeći korak je dokumentacija za testiranje.

Testovi koji su definisani kao kvalitetni testovi sastoje se iz više faza.

Cilj je da razrade sistem testova, kako bi sve bilo urađeno do detaljnih postupaka testiranja.

Faze su poznate kao:

- Strategija testiranja,
- Planiranje testiranja,
- Izrade test primera i
- Razrada samog postupka testiranja

Svaki test se razrađuje na osnovu programske specifikacije.

STRATEGIJA TESTIRANJA SOFTVERA

Tek sada možemo napraviti plan testiranja programa

Planom testiranja programa obuhvaćeni su segmenti programa koji će se testirati i u kojoj fazi razvoja, redosled testiranja, na koji način će se strategija testiranja primenjivati na svaku jedinicu i konačno tip testiranja programa.

Kao i za ostale faze testiranja, tako i za plan testiranja važi pravilo koje podrazumeva da on bude dat za ceo projekat ili u drugom slučaju da to bude čitava hijerarhija planova za različite nivoe specifikacije i testiranja.

PLAN TESTIRANJA

Plan testiranja materijalizuje strategiju testiranja, opisuje resurse koji će se koristiti, samo okruženje, odnosno okruženje u kojem će se testiranje izvršiti. Treba voditi računa o ograničenjima (ako postoje) koja će se primeniti.

Takođe treba voditi računa o izvođenja testova, kao i o njihovom redosledu.

Dnevnik testiranja

Planovi testiranja su različiti za različite nivoe.

STRATEGIJA TESTIRANJA SOFTVERA

- Plan testiranja integracije programa: u okviru kojeg su integrisane sve programske komponente. U nekim slučajevima on je deo specifikacije arhitekture.
- Plan testiranja jedinica podrazumeva plan za testiranje svake komponente ponaosob.
- Može biti deo detaljnih specifikacija.

STRATEGIJA TESTIRANJA SOFTVERA

- Realizacijom plana testiranja postoji mogućnost potvrde specifikacije proizvedenog softvera. U okviru programske specifikacije:
- testiranje prihvatljivosti i testiranje sistema.

- Sledeća faza jeste razvoj test stavki

STRATEGIJA TESTIRANJA SOFTVERA

- Razvoj testa nakon napravljenog plana testiranja za određeni nivo zahteva određivanje skupa test stavki ili test pitanja za svaki objekat koji je obuhvaćen testom na datom nivou. Svaki objekat sadrži određeni broj test stavki koji se može odrediti a samim tim i testirati na svakom nivou.
- Test stavke mogu biti u obliku posebnog dokumenta (specifikacija testiranja ili opisane u obliku samog teksta)

STRATEGIJA TESTIRANJA SOFTVERA

Specifikacija testiranja prihvatanja (kao poseban dokument ili u okviru plana testiranja sistema) određuje test stavke za integraciju i testiranje sistema.

Specifikacija testiranja integracije programa u okviru specifikacije arhitekture određuje test stavke za svaku fazu integracije programskih komponenti koje se testiraju.

STRATEGIJA TESTIRANJA SOFTVERA

- Specifikacija testiranja jedinice (kao deo detaljnih specifikacija) određuje test stavke za testiranje komponenti ponaosob.
- Za uspešno sistemsko testiranje i testiranje prihvatljivosti potreban je veliki broj pojedinačnih test stavki. Indeks takozvanih "unakrsnih referenci" za zahteve i test stavke kao deo specifikacije testiranja neophodan je radi utvrđivanja koji zahtevi su testirani sa kojim test stavkama.

STRATEGIJA TESTIRANJA SOFTVERA

Positive and negative testing. Jako bitno testiranje softvera.

- Testiranje pozitivnih i negativnih vrednosti svih ulaza.

Razlike između pozitivnog i negativnog testiranja se sastoje u tome što se kod pozitivnog testiranja proverava da li program radi ono što treba, a kod negativnog testiranja da li program radi nešto što ne bi bi trebalo da radi.

- U oba slučaja je bitno odrediti test stavke.

STRATEGIJA TESTIRANJA SOFTVERA

Primena skupa test stavki, kao poslednja faza razvoja testova predstavlja u određenoj meri relativan postupak, i u tom smislu se poredi sa razvojem programskih jedinica sa višeg nivoa funkcionalnog opisa.

STRATEGIJA TESTIRANJA SOFTVERA

- Postupci koji se vrše za svaku test stavku moraju biti unapred određeni.
- Bez obzira koji se postupak testira postupak testiranja je odlučujući pri odabiru postupka uvođenja test stavki.
- Prilikom sprovođenja postupka testiranja, svaki korak se mora ispoštovati, pri čemu se ne sme oslanjati na pretpostavke.

STRATEGIJA TESTIRANJA SOFTVERA

- Poslednja faza jeste dokumententacija rezultata testova.
- Sprovođenjem testova do kraja , za svako izvršenje testa.
- Sve izlazne vrednosti se dokumentuje u određenom fajlu u kome su smešteni rezultati testa.
- U cilju procene uspešnosti testiranja, tj. da bi se utvrdio ishod testiranja, rezultati iz datog fajla se uporede sa kriterijumima zadatim iz same specifikacije.

STRATEGIJA TESTIRANJA SOFTVERA

- Dnevnik testiranja je neophodan kako bi se dokumentovali načini izvršenja testa.
- Takva vrsta dnevnika sadrži osnovne podatke vezane za to kad je koji test izvržen, za ishod izvršenja kao i osnovna zapažanja tokom testiranja.
- Vođenje dnevnika testiranja prilikom testiranja jedinice i integracije programa nije uvek uobičajen, ali je sugestija da se mora imati.

STRATEGIJA TESTIRANJA SOFTVER

- Pregled rezultata testiranja i dokumentacija analiza sadržani su u izveštaju o testiranju.
- Izveštaj se daje u bilo kojoj fazi testiranja.
- Izveštaj o testiranju prihvatljivosti je često u formi ugovora i njime je utvrđena saglasnost o prihvatanju programa.

STRATEGIJA TESTIRANJA SOFTVERA

Kod testiranja softvera najbitnije je odrediti niz test stavki za ono što se testira.

Pre toga moramo odrediti koje informacije se moraju nalaziti u test stavkama.

STRATEGIJA TESTIRANJA SOFTVERA

- Najočitija informacija su ulazi kojih ima dve vrste: preduslovi (okolnosti koje postoje pre izvršenja test stavke) i stvarni ulazi, koje identifikujemo nekim metodom testiranja.

Mnogo je efikasnije dobro razmisliti o vrstama grešaka koje su najverovatnije (ili najštetnije) i tada odabrati metode testiranja koji će verovatno moći da otkriju takve greške.

Nema smisla testirati greške koje verovatno ne postoje.