QA TESTIRANJE

Seminarski rad

Prof: Dr Janoš Šimon

Strahinja Mirković 16118140

Contents

UVOD

Svi softverski problemi mogu se nazvati greškama. Softverska greška se obično javlja kada softver ne radi ono što je namenjen ili radi nešto što nije namenjen. Greške u specifikacijama, dizajnu, kodu ili drugim razlozima mogu izazvati ove greške. Prepoznavanje i ispravljanje grešaka u ranim fazama softvera je veoma važno jer troškovi ispravljanja grešaka vremenom rastu. Dakle, cilj ispitivača softvera je da pronađe greške i pronađe ih što je ranije moguće i da se uveri da su ispravljene.

Testiranje se zasniva na kontekstu i temelji se na riziku. Potreban je metodičan i disciplinovan pristup pronalaženju grešaka. Dobar ispitivač softvera mora da izgradi kredibilitet i poseduje stav da bude istraživački, rešavajući probleme, neumoljiv, kreativan, diplomatski i uverljiv.

Za razliku od percepcije da testiranje započinje tek nakon završetka faze kodiranja, ono zapravo započinje i pre nego što se može napisati prvi red koda. U životnom ciklusu konvencionalnog softverskog proizvoda, ispitivanje započinje u fazi kada su napisane specifikacije, tj. od testiranja specifikacija proizvoda ili specifikacija proizvoda. Pronalaženje grešaka u ovoj fazi može uštedeti ogromnu količinu vremena i novca.

ŠTA JE SOFTVER?

Softver je niz uputstava za računar koji izvršavaju određeni zadatak, koji se naziva program; dve glavne kategorije softvera su sistemski softver i aplikativni softver. Sistemski softver se sastoji od upravljačkih programa.

Aplikativni softver je bilo koji program koji obrađuje podatke za korisnika (proračunska tabela, program za obradu teksta, obračun zarade itd.) Softverski proizvod treba da se izda tek nakon što prođe odgovarajući proces razvoja, testiranja i ispravljanja grešaka. Testiranje se bavi oblastima kao što su performanse, stabilnost i rukovanje greškama postavljanjem testova scenarija pod kontrolisanim uslovima i procenom rezultata. Zbog toga bilo koji softver mora biti testiran. Važno je napomenuti da se softver uglavnom testira kako bi se utvrdilo da li ispunjava potrebe kupaca i da li je u skladu sa standardima. Uobičajena je norma da se softver smatra kvalitetnim ako ispunjava zahteve korisnika.

Kvalitet se ukratko može definisati kao „stepen izvrsnosti“. Kvalitetni softver obično odgovara zahtevima korisnika. Kupčeva ideja o kvalitetu može pokriti širok spektar karakteristika - usklađenost sa specifikacijama, dobre performanse na platformi / konfiguracijama, u potpunosti ispunjava operativne zahteve (čak i ako nisu navedeni!), Kompatibilnost sa svom opremom krajnjeg korisnika, bez negativnog uticaja na postojećoj bazi krajnjih korisnika u vreme uvođenja. Kvalitetni softver štedi dobru količinu vremena i novca. Budući da će softver imati manje nedostataka, to štedi vreme tokom faze testiranja i održavanja. Veća pouzdanost doprinosi nemerljivom povećanju zadovoljstva kupaca, kao i nižim troškovima održavanja. Budući da održavanje predstavlja veliki deo svih softverskih troškova, ukupni troškovi projekta će najverovatnije biti niži od sličnih projekata.

ŠTA TESTER SOFTVERA RADI?

Pored pronalaženja grešaka u softverskom proizvodu koji potvrđuju da program ispunjava specifikaciju programa, kao inženjer morate da kreirate ispitivanja, procedure, skripte i generišete podatke. Izvršavate ispitne procedure i skripte, analizirate standarde i procenjujete rezultate sistemskog testiranja. Vi takođe...

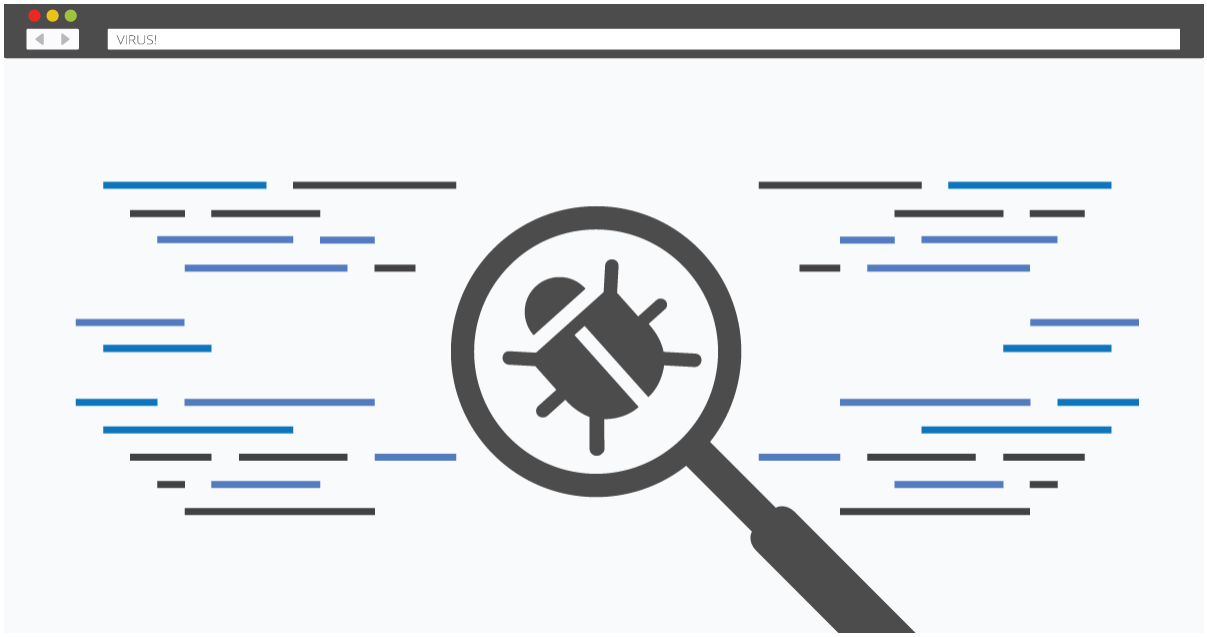
- Ubrzati proces razvoja identifikovanjem grešaka u ranoj fazi
- Smanjite rizik organizacije od pravne odgovornosti
- Osigurajte uspešno lansiranje proizvoda, uštedite novac, vreme i reputaciju kompanije otkrivanjem grešaka i nedostataka u dizajnu u ranoj fazi pre nego što se pojave kvarovi u proizvodnji ili na terenu
- Osigurava kontinuirani rast kvaliteta

Kako se softverski inženjering sada smatra profesijom tehničkog inženjeringa, važno je da inženjer softverskog testiranja poseduje određene osobine sa neumoljivim stavom kako bi se istakao. Evo nekoliko:

- **POZNAVANJE TEHNOLOGIJE**. Poznavanje tehnologije u kojoj je aplikacija razvijena dodatna je prednost bilo kom ispitivaču. Pomaže u dizajniranju boljih i moćnijih testova na osnovu slabosti ili nedostataka tehnologije. Dobri testeri znaju šta podržava, a šta ne, pa će im koncentracija na ovo pomoći da brzo razbiju aplikaciju.
- **PERFEKCIONISTA I REALISTA**. Biti perfekcionist pomoći će testerima da uoče problem, a biti realista na kraju dana zna koji su problemi zaista važni problemi. Znaćete koji zahtevaju popravak, a koji ne.
- **TAKTIČKI, DIPLOMATSKI I UBEDLJIVO.** Dobri testeri softvera su taktični i znaju kako da prenesu vesti programerima. Razgovaraju sa programerima o grešakama i objašnjavaju im zašto je to potrebno i popravljaju njihove greške. Važno je biti kritičan prema pitanju i ne dozvoliti da osoba koja je razvila aplikaciju bude zatečena nalazima.

ŠTA JE BUG?

Softverska greška može se definisati kao greška u kodiranju koja uzrokuje neočekivani kvar, grešku, manu ili nesavršenost u računarskom programu. Drugim rečima, ako program ne radi kako je predviđeno, najverovatnije je reč o grešci. U softveru postoje greške zbog nejasnih ili stalno promenljivih zahteva, složenosti softvera, grešaka u programiranju, vremenskih rokova, grešaka u praćenju grešaka, komunikacijskog jaza, dokumentacije greške, odstupanje od standarda itd.



Nejasni zahtevi za softver nastaju zbog pogrešne komunikacije u vezi sa tim šta softver treba, a šta ne sme da radi. U mnogim prilikama kupcu možda nije potpuno jasno kako proizvod na kraju treba da funkcioniše. Ovo je naročito tačno kada je softver razvijen za potpuno novi proizvod. Takvi slučajevi obično dovode do puno pogrešnih tumačenja bilo koje ili obe strane.

Stalno menjanje zahteva za softverom izaziva veliku zabunu i pritisak kako na razvojne, tako i na testne timove. Često se nova funkcija ili uklonjena postojeća funkcija može povezati sa ostalim modulima ili komponentama u softveru. Previđanje takvih problema uzrokuje greške.

Takođe, ispravljanje greške u jednom delu / komponenti softvera može prouzrokovati nastanak drugog u drugoj ili istoj komponenti. Nedostatak predviđanja u predviđanju takvih problema može prouzrokovati ozbiljne probleme i povećati broj grešaka. Ovo je jedno od glavnih problema zbog kojih se javljaju greške, jer su programeri vrlo često izloženi pritisku

vezanom za vremenske rokove; često menjanje zahteva, povećanje broja grešaka itd Dizajniranje i redizajniranje korisničkog interfejsa, integracija modula, upravljanje bazama podataka, sve to dodaje složenosti softvera i sistema u celini.

Osnovni problemi sa dizajnom softvera i arhitekturom mogu izazvati probleme u programiranju. Razvijeni softver je sklon greškama jer i programeri mogu praviti greške. Kao tester koji možete da potražite, greške u referenci / deklaraciji podataka, greškama kontrolnog toka, greškama parametara, greškama unosa / izlaza itd.

Životni ciklus baga započinje nenamernom softverskom greškom i završava se kada dodeljeni programer otkloni grešku. Grešku kada je pronađena treba saopštiti i dodeliti programeru koji može da je otkloni. Jednom kada se popravi, problematično područje treba ponovo testirati. Takođe, treba izvršiti potvrdu da bi se utvrdilo da li ispravak nije stvorio probleme negde drugde. U većini slučajeva životni ciklus postaje veoma komplikovan i teško ga je pratiti što je neophodno zbog uspostavljanja sistema za praćenje grešaka / kvarova.

NAJČEŠĆE SOFTVERSKE GREŠKE

Slede najčešće softverske greške koje vam pomažu u testiranju softvera. Ovo vam pomaže da sistematski prepoznate greške i povećate efikasnost i produktivnost testiranja softvera.

- **Pogreške u korisničkom interfejsu**: Nedostajuće / pogrešne funkcije Ne radi ono što korisnik očekuje, nedostatak informacija, prekomplikovano, zbunjujuće informacije, pogrešan sadržaj, neprikladne poruke o greškama. Problemi sa performansama - loša reakcija, ne može se preusmeriti izlaz...
- **Rukovanje greškama**: Neadekvatna zaštita od oštećenih podataka, testovi korisničkog unosa, kontrola verzija; Zanemaruje - prelivanje, upoređivanje podataka, oporavak grešaka - brisanje grešaka, oporavak od hardverskih problema
- **Greške u računanju**: Loša logika, Loša aritmetika, Zastarele konstante, Greške u proračunu, netačna konverzija iz jednog predstavljanja podataka u drugi, Pogrešna formula, netačna aproksimacija.
- **Početna i kasnija stanja**: Neuspeh postavljanja stavke podataka na nulu, inicijalizacija promenljive kontrole petlje ili ponovna inicijalizacija pokazivača, brisanje niza ili zastavice, Netačna inicijalizacija.
- **Hardver**: Pogrešan uređaj, Uređaj nedostupan, Nedovoljno korišćenje inteligencije uređaja, Pogrešno shvaćen status ili povratni kod, Pogrešni kodovi rada ili uputstva.
- **Greške u testiranju**: Neprimetenje / prijava problema, Neiskorišćenje najobećavajućeg testcase-a, Oštećene datoteke podataka, Pogrešno protumačene specifikacije ili dokumentacija, Nejasno objašnjenje načina reprodukcije problema, Neprovera nerešenih problema neposredno pre objavljivanja, Neuspeh proveri ispravke, Ne pružanje rezime izveštaja

TERMINI KOD TESTIRANJA

• **BUG**: Softverska greška može se definisati kao greška u kodiranju koja uzrokuje neočekivani kvar, grešku ili manu. Drugim rečima, ako program ne radi kako je predviđeno, najverovatnije je reč o grešci.

• **ERROR**: Neusklađenost programa i njegove specifikacije predstavlja grešku u programu.

• **DEFECT**: Defekt je odstupanje od željenog atributa proizvoda. Može biti dve vrste - kvar na proizvodu ili odstupanje od očekivanja kupaca / korisnika. To je mana softverskog sistema i nema uticaja dok ne utiče na korisnika / kupca i operativni sistem. 90% svih nedostataka mogu nastati zbog problema u procesu.

• **FAILURE**: Kvar koji uzrokuje grešku u radu ili negativno utiče na korisnika / kupca

• **QUALITY CONTROL**: Orijentisan je ka sprečavanju kvara. Osiguranje kvaliteta osigurava da se sve strane koje se bave projektom pridržavaju procesa i procedura, standarda i obrazaca i pregleda spremnosti za ispitivanje. inženjering kontrole kvalitetač kvalitet je skup mera preduzetih kako bi se osiguralo da se proizvodi ili usluge sa nedostacima ne proizvode i da dizajn ispunjava zahteve performansi.

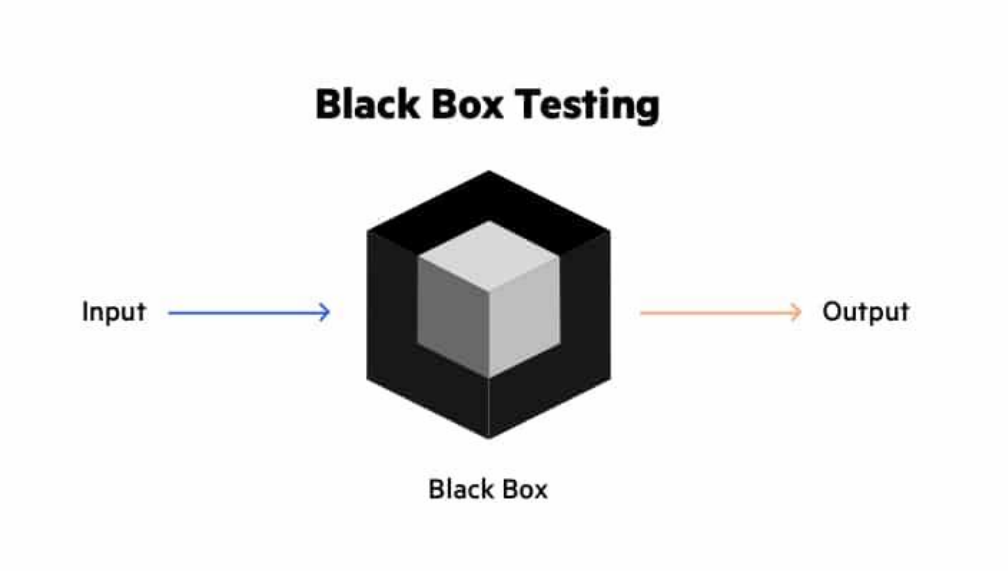
• **VERIFICATION**: Verifikacija osigurava da je proizvod dizajniran da isporuči svu funkcionalnost kupcu; obično uključuje preglede i sastanke radi procene dokumenata, planova, koda, zahteva i specifikacija; to se može uraditi pomoću kontrolnih lista, lista izdavanja, uputstava i inspekcijskih sastanaka.

• **VALIDATION**: Validacija osigurava da funkcionalnost, kako je definisana u zahtevima, bude predviđeno ponašanje proizvoda; validacija obično uključuje stvarno testiranje i odvija se nakon što se verifikacija dovrši.

BLACK BOX TESTING

Black box testing je metoda ispitivanja softvera u kojoj se funkcionalnosti softverskih aplikacija testiraju bez znanja o unutrašnjoj strukturi koda, detaljima implementacije i unutrašnjim putanjama. Testiranje crne kutije uglavnom se fokusira na unos i izlaz softverskih aplikacija i u potpunosti se zasniva na softverskim zahtevima i specifikacijama. Takođe je poznato i kao testiranje ponašanja.

Gore navedeni Black-Bok može biti bilo koji softverski sistem koji želite da testirate. Na primer, operativni sistem poput Windows-a, veb-sajt poput Google-a, baza podataka poput Oracle-a ili čak vaša prilagođena aplikacija. U okviru testiranja crne kutije, možete testirati ove aplikacije samo fokusiranjem na ulaze i izlaze bez znanja njihove interne implementacije



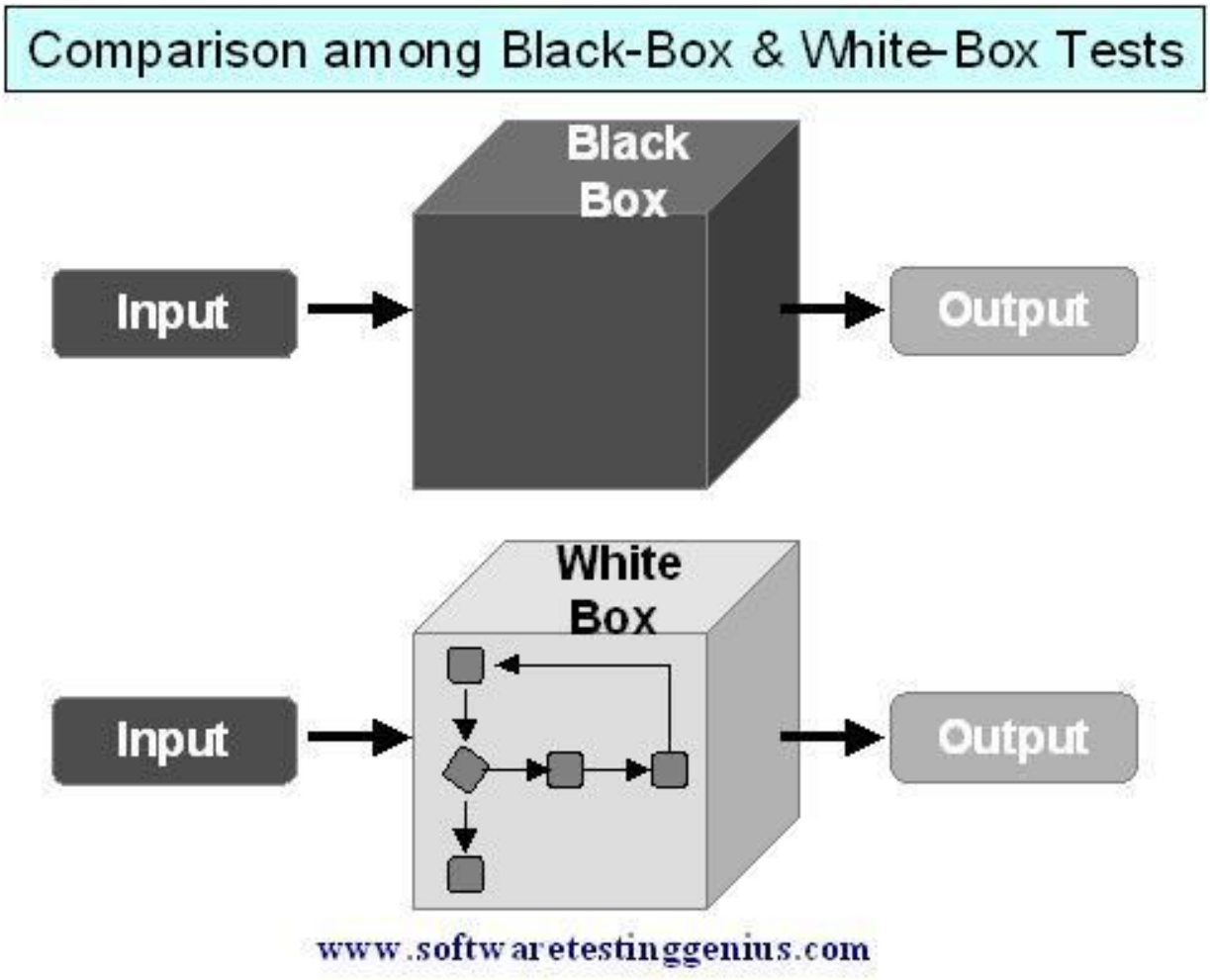
koda.

Postoji mnogo vrsta black box testiranja, ali sledeće su istaknute:

- **Funkcionalno testiranje** - Ovaj tip testiranja crne kutije povezan je sa funkcionalnim zahtevima sistema; to rade testeri softvera.
- **Nefunkcionalno testiranje** - Ova vrsta testiranja crne kutije nije povezana sa testiranjem određene funkcionalnosti, već nefunkcionalnim zahtevima kao što su performanse, skalabilnost, upotrebljivost.
- **Ispitivanje regresije** - Ispitivanje regresije vrši se nakon popravki koda, nadogradnji ili bilo kog drugog održavanja sistema radi provere novog koda koji nije uticao na postojeći kod.

WHITE BOX TESTING

White box testing je tehnika softverskog testiranja u kojoj se testiraju unutrašnja struktura, dizajn i kodiranje softvera kako bi se potvrdio protok ulaz-izlaz i poboljšalo dizajn, upotrebljivost i sigurnost. U white box testiranju, kod je vidljiv testerima, pa se naziva i Clear box testiranje, testiranje otvorenih kutija, transparentno testiranje, testiranje zasnovano na kodu i testiranje Glass box.



Prvo što će tester često učiniti je da nauči i razume izvorni kod aplikacije. Budući da testiranje bele kutije uključuje ispitivanje unutrašnjeg rada aplikacije, ispitivač mora biti veoma upućen u programske jezike koji se koriste u aplikacijama koje testiraju. Takođe, osoba koja testira mora biti veoma svesna bezbednih praksi kodiranja. Sigurnost je često jedan od primarnih ciljeva testiranja softvera. Tester bi trebalo da bude u stanju da pronađe sigurnosne probleme i spreči napade hakera i naivnih korisnika koji bi mogli ubrizgati zlonamerni kod u aplikaciju ili svesno ili ne znajući.