Dražen Drašković
Dragan Bojić

Testiranje softvera

Beograd, 2019.

Dražen Drašković, Dragan Bojić

T E S T I R A NJ E S O F T V E R A

Recenzenti

Prof. dr Boško Nikolić
Doc. dr Miodrag Živković

Izdavač

AKADEMSKA MISAO
Bulevar kralja Aleksandra 73, Beograd

Dizajn naslovne stranice

Prof. dr Vladana Likar-Smiljanić

Štampa

Akademska misao, Beograd

Tiraž

300 primeraka

ISBN: 978-86-7466-815-3 (štampano izdanje)

Odlukom Nastavno-naučnog veća Elektrotehničkog fakulteta u Beogradu od 15. oktobra 2019. godine, ova knjiga je odobrena kao zvaničan udžbenik u štampanom i elektronskom izdanju.

„Da, istina je da ja uvek preduzmem
više nego što mogu da ostvarim“.

- Nikola Tesla

Predgovor

Ovaj udžbenik nastao je kao rezultat višegodišnjeg nastavnog rada autora na predmetu Testiranje softvera, na osnovnim akademskim studijama studijskog programa Softversko inženjerstvo na Elektrotehničkom fakultetu Univerziteta u Beogradu. Udžbenik treba da posluži bržem i potpunijem savladavanju teorijskih i praktičnih osnova iz oblasti testiranja softvera. Ciljevi knjige su upoznavanje čitaoca sa najčešće korišćenim tehnikama testiranja u različitim domenima primene softvera i ilustracija ovih tehnika na problemima čiji je stepen složenosti takav da omogućava praćenje rešenja bez većeg napora.

Uvodno poglavlje bavi se motivisanjem čitaoca za proučavanje oblasti i uvodi osnovnu terminologiju. Drugo poglavlje opisuje različite tehnike funkcionalnog testiranja, kao što su podela na klase ekvivalencije, analiza graničnih vrednosti, testiranje zasnovano na tabeli odlučivanja, uzročno-posledični grafovi, testiranje zasnovano na modelu stanja, testiranje sintakse i kombinatorno testiranje.

Poglavlje o tehnikama strukturnog testiranja, odnosno tehnikama bele kutije, bavi se tehnikama zasnovanim na kontroli toka, zatim tehnikama zasnovanim na programskim putanjama, kao što su metod bazičnih putanja, granično testiranje unutrašnje putanje, testiranje petlji i pokrivanje sekvenci LCSAJ (eng. *Linear code sequence and jump*). Potom se uvode tehnike na bazi toka podataka kod kojih se selekcija programskih putanja vrši na osnovu lokacija u programu gde se promenljivama dodeljuje vrednost ili gde se ta vrednost koristi. Na kraju ovog poglavlja se objašnjava tehnika mutacionog testiranja. Sledeće poglavlje bavi se tehnikama integracionog i regresivnog testiranja. Opisani su različiti pristupi kao što su integracija po principu „velikog praska“, zatim pristupi postupne integracija zasnovane na hijerarhijskoj strukturi programa (od vrha ka dnu, od dna ka vrhu i mešovita integracija), postupna integracija zasnovana na grafu poziva (po parovima, po susedstvu), klijent/server komunikacija i „visokofrekventna“ integracija.

Preostala poglavlja razmatraju specifičnosti testiranja pojedinih kategorija softvera. Poglavlje o testiranju objektno-orijentisanog programskog koda razmatra kako osobine ove vrste softvera kao što su ponašanje zavisno od

stanja, enkapsulacija, nasleđivanje, polimorfizam i dinamičko vezivanje, apstraktne klase i obrada izuzetaka utiču na realizaciju testiranja. Opisani su razni metodi unutar klasnog i međuklasnog testiranja i razmotrena problematika predikcije rezultata testova.

U poglavlju o testiranju konkurentnog softvera razmatraju se osobine kao što je nedeterminizam izvršavanja i potrebe za komunikacijom i sinhronizacijom i njihov uticaj na testiranje. Opisane se tehnike za otkrivanje narušavanja uzajamnog isključivanja, praćenje i ponovnu reprodukciju izvršavanja, detekcija zastoja i testiranje dostižnosti.

Poglavlje o testiranju grafičkih korisničkih interfejsa opisuje različite pristupe testiranju ove kategorije softvera kao što su manuelno testiranje, zasnovano na poznavanju funkcija aplikacija i domena korišćenja od strane korisnika ili testera, slučajno (eng. *random input*), tehnike zasnovane na snimanju i reprodukciji korisničkih sesija i tehnike testiranja zasnovane na formalnim modelima (model stanja, varijabilni model stanja, model toka događaja).

Autori se nadaju da će ovakva knjiga, pored toga što će biti zvaničan udžbenik na predmetu Testiranje softvera, poslužiti i novim softverskim inženjerima da se upoznaju sa osnovnim konceptima i tehnikama testiranja softvera.

Autori se zahvaljuju kolegama recenzentima i kolegi Milošu Gligoriću, nekadašnjem saradniku u nastavi, za doprinose i korisne sugestije u konačnom uobličavanju ove knjige. Veliku zahvalnost dugujemo i našoj dragoj Vladani Likar-Smiljanić, profesorki u penziji Elektrotehničkog fakulteta u Beogradu, koja je korice knjige ulepšala simboličnom ilustracijom borbe za kvalitet u razvoju softvera.

U Beogradu,
20. oktobra 2019. godine

Autori

Sadržaj

Uvod

Savremena civilizacija sve više je zavisna od računarskih sistema i softvera. Takozvani industrijski softver (eng. *industrial strength software*) napravljen je da rešava neki poslovni problem korisnika, tj. važne aktivnosti zavise od korektnog funkcionisanja sistema. Loše funkcionisanje izaziva nezadovoljstvo korisnika i finansijske, materijalne, gubitke, ili čak ljudske žrtve.

Kao posledica potreba za kvalitetom, procenjuje se da u praksi od 30% do 50% ukupnog napora (troška) razvoja softvera odlazi na testiranje i proveru kvaliteta. Kao ilustracija dimenzije problema neka posluže podaci da se veličina operativnog sistema *Windows 2003* procenjivala na približno 50 miliona linija izvornog koda (MLOC), dok je veličina *Linux* kernel verzije 2.6.32 veća od 12 MLOC.

Američki nacionalni institut za standarde (NIST) procenjuje da softverski bagovi (eng. *software bug*) izazivaju 60 milijardi dolara gubitaka u američkoj ekonomiji godišnje. Jedno ispitivanje u američkom ministarstvu odbrane pokazalo je da se čak 70% otkaza opreme može pripisati softveru (u sistemima sa mnoštvom električnih, mehaničkih i hidrauličnih komponenata). Može se reći da su tradicionalne tehničke discipline znatno zrelije, a softver je često slaba tačka sistema. Specifičnost softvera je i u tome što za razliku od fizičkih sistema kod kojih se otkazi često javljaju usled fizičkih i električnih promena uzrokovanih starenjem, softver ne stari, greške se nalaze u njemu od početka, a mogu se manifestovati u vidu otkaza i posle dužeg ispravnog rada.

Terminologija u vezi sa greškama i testiranjem

U ANSI/IEEE standardnom rečniku terminologije softverskog inženjerstva, definisani su sledeći termini u vezi softverskih grešaka i testiranja:

- Greška (eng. *error*) je posledica ljudskog rada, na primer, prilikom specifikacije zahteva ili kodiranja programa.
- Mana, defekat (eng. *fault*) je nepoželjna osobina nekog radnog proizvoda koja je posledica ljudske greške (na primer, programu nešto nedostaje, ili ima neispravnu funkciju).
- Otkaz (eng. *failure*) je nemogućnost sistema da obavi zahtevanu funkciju i najčešće se javlja aktiviranjem (izvršavanjem) defektnog koda.

- Poremećaj (eng. *incident*) je spoljni simptom koji korisniku otkriva otkaz.

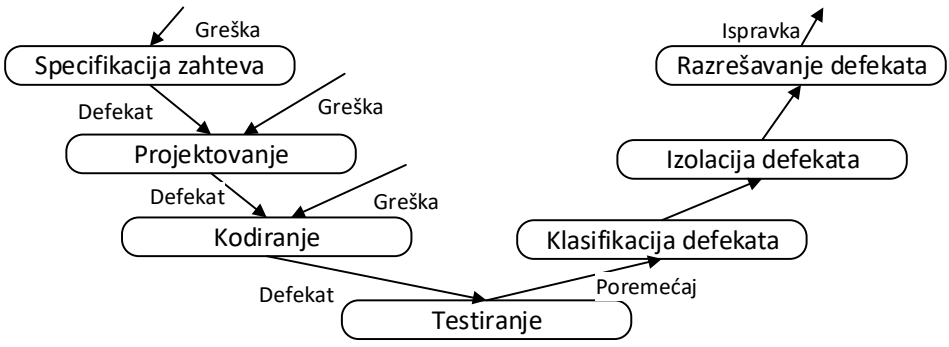
Testiranje softvera je, prema Glenfordu Majersu (*Glenford Myers*), proces izvršavanja programa sa osnovnim ciljem nalaženja defekata. U širem smislu, radi se o ispitivanju softverskog proizvoda ili servisa sa ciljem objektivnog utvrđivanja kvaliteta. Testiranje softvera je deo šire oblasti osiguranja kvaliteta (eng. *software quality assurance*, skr. SQA), koja se bavi kontrolom procesa izrade da bi se obezbedio kvalitetan finalni proizvod u predvidivom vremenskom roku.

Testiranje može otkriti otkaze, ali potom treba eliminisati defekte - debagovati program (eng. *debbuging*), što predstavlja posebnu aktivnost.

Test primer (eng. *test case*) poseduje identitet i ispituje određeno ponašanje programa. Test primer poseduje skup ulaznih vrednosti, opis stanja sistema i skup očekivanih rezultata izvršavanja programa za date ulazne vrednosti u datom stanju sistema. Serijom testova naziva se kolekcija povezanih test primera (razvijenih na primer primenom određenog metoda testiranja).

Uloga testiranja u životnom ciklusu softvera

Tipičan životni ciklus softvera polazi od specifikacije korisničkih zahteva, preko faze analize i projektovanja, programske implementacije (kodiranja), do testiranja pre puštanja softvera u rad i kasnijeg održavanja. U svakoj od ovih faza mogu nastati defekti kao posledica ljudske greške i neotklonjeni defekti iz prethodnih faza prenose se u naredne faze.

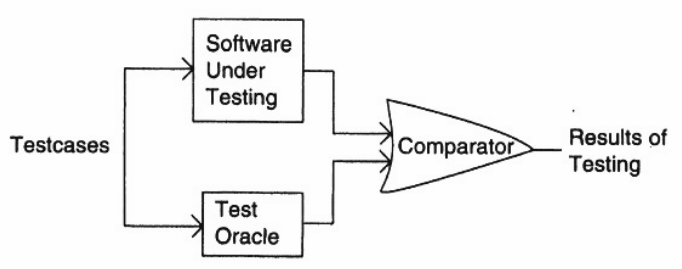


Slika 1. Životni ciklus softvera

Cilj aktivnosti testiranja (koja je prisutna pre i u toku upotrebe softvera od strane krajnjeg korisnika) je otkrivanje defekata, a zatim se defekti klasifikuju, izoluju na određeni deo softvera i razrešavaju. Aktivnost razrešavanja defekata osim što uklanja uočene defekte pruža takođe priliku za nastajanje grešaka i novih defekata.

Predviđanje rezultata testa

Prediktor testa (eng. *test oracle*) je mehanizam, nezavisan od samog programa, koji se može upotrebiti za proveru ispravnosti rada programa za test primere. Konceptualno, test primeri se zadaju programu i prediktoru i njihovi izlazi potom međusobno upoređuju.



Slika 2. Prediktor testa

Prediktor testa može biti čovek ili automat. Ljudi koriste specifikaciju programa da odluče šta je ispravno ponašanje programa. Međutim, specifikacije programa su često sa greškama, nepotpune ili dvosmislene, a i ljudi mogu napraviti previd. Automatizovni prediktori koriste formalne specifikacije ponašanja programa i tačni su onoliko koliko je specifikacija tačna.

Potreba za projektovanjem test primera

Skoro svaki netrivijalni sistem ima ekstremno veliki domen ulaznih podataka, što testiranje za svaku zamislivu kombinaciju ulaza čini iscrpnim, nemogućim ili nepraktičim. Kao primer uzmimo komandu sort na *Unix*-u, koja ima dvadesetak različitih parametara komandne linije:

sort [-cmu] [-ooutput] [-Tdirectory] [-y [kmem]] [-zrecsz] [dfiMnr] [-b]
[tchar] [-kkeydef] [+pos1[-pos2]] [file...]