„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

Predgovor.....i

Sadržaj.....iii

Uvod.....1

Terminologija u vezi sa greškama i testiranjem 1

Uloga testiranja u životnom ciklusu softvera 2

Predviđanje rezultata testa 3

Potreba za projektovanjem test primera 3

Kriterijumi selekcije testova 4

Testiranje tehnikama crne kutije.....7

METOD DELJENJA NA KLASE EKVIVALENCIJE 8

Kako odrediti klase ekvivalencije? 9

Multidimenzionalno particioniranje 11

Određivanje test primera na osnovu klasa 13

Zadatak 1: Prodavnica obuće 14

Zadatak 2: Problem trougla 16

Zadatak 3: Generator ocena 19

Zadatak 4: Sajam antikvitetnih knjiga 28

METOD GRANIČNIH SLUČAJEVA 40

Zadatak 1: Mtest evidencija studenata 45

Zadatak 2: Generator ocena (sa graničnim vrednostima) 50

Zadatak 3: ETF Alumni 58

METOD UZROČNO-POSLEDIČNOG GRAFA 76

Konstrukcija grafa 77

Transformacija grafa u tabelu odlučivanja 80

Tehnika senzitizacije putanja 85

Zadatak 1: Novčane transakcije 87

Zadatak 2: Kombinacije uzroka 90
Zadatak 3: Potprogram IME 93
Zadatak 4: Tabela odlučivanja po Majersu 99
Zadatak 5: Premija osiguranja 101
Zadatak 6: Bankovni računi 107
KOMBINATORNO TESTIRANJE 111
Strategije kombinatornog testiranja 112
Konstrukcija kombinatornih tabela 114
Zadatak 1: Metod testiranja svih parova 120
Zadatak 2: Testiranje IPO procedurom 129
TESTIRANJE ZASNOVANO NA MODELU STANJA 134
Kriterijumi testiranja modela stanja 136
Pokrivanje stanja/prelaza 137
Pokrivanje izmena 139
Pokrivanje skupova N-izmena 141
Tabela prelaza konačnog automata 142
Validnost kriterijuma testiranja zasnovanog na modelu stanja 144
Zadatak 1: Bankarske usluge 148
Zadatak 2: E-Student portal 155
Zadatak 3: Planinarski satovi 160
Testiranje strategijama bele kutije 163
TEHNIKE ZASNOVANE NA KONTROLI TOKA 164
Pokrivanje iskaza 164
Pokrivanje odluka ili grana 165
Pokrivanje uslova 166
Pokrivanje odluka i uslova 167
Pokrivanje višestrukih uslova 167
Minimalno pokrivanje višestrukih uslova 167
Zadatak 1: Pokrivanje iskaza 168

Zadatak 2: Pokrivanje odluka 170
Zadatak 3: Pokrivanje uslova 172
Zadatak 4: Deljenje sa nulom 173
Zadatak 5: Pokrivanje odluka i uslova 174
Zadatak 6: Pokrivanje višestrukih uslova 175
Zadatak 7: Višestruki uslovi 179
TESTIRANJE ZASNOVANO NA PROGRAMSKIM PUTANJAMA 181
Graf toka kontrole (CFG) 181
Testiranje potpunim pokrivanjem putanja 182
Pokrivanje bazičnih putanja 184
Testiranje petlji prema Bajzeru 188
LCSAJ sekvence (JJ-path) 191
Zadatak 1: Testiranje ugneždene WHILE petlje 194
Zadatak 2: Uspeh studenata (pokrivanje linearno nezavisnih putanja) 197
Zadatak 3: Skokovi sa GoTo (LCSAJ) 200
Zadatak 4: Stepenovanje (LCSAJ) 201
Zadatak 5: FOR petlja i LCSAJ 203
Zadatak 6: Uspeh studenata (granično testiranje unutrašnje putanje) 205
TESTIRANJE METODOM TOKA PODATAKA 209
Zadatak 1: Ocene na predmetu (c- i p-upotrebe) 212
Zadatak 2: Trgovinska radnja i čokolade 217
Zadatak 3: Binarno pretraživanje 219
Zadatak 4: Sportska radnja i popusti 222
Zadatak 5: Konverzija decimalnog iz binarnog 226
Zadatak 6: Konverzija decimalnih u hex 228
Zadatak 7: Razne strategije bele kutije 230
MUTACIONO TESTIRANJE 232
Zadatak 1: Aritmetičke operacije i mutanti 262
Zadatak 2: Relacioni operatori u bankarskom softveru 265

Zadatak 3: Konverzija decimalnog broja ... 268
Zadatak 4: Brojač alfabetskih karaktera ... 273

Integraciono testiranje ... 278

Integraciono testiranje zasnovano na dekompoziciji ... 279
Integraciono testiranje na osnovu grafova pozivanja ... 285
Zadatak 1: Integracija programa od vrha ka dnu ... 288
Zadatak 2: Integracija programa - svi pristupi ... 290
Zadatak 3: Novi dan ... 295

Testiranje objektno orijentisanog softvera ... 304

UVOD ... 304
UNUTAR KLASNO TESTIRANJE ... 306
Testiranje obrade izuzetaka ... 312
Ponovno korišćenje testova za podklase ... 312
MEĐUKLASNO TESTIRANJE ... 313
Interakcije u međuklasnim testovima ... 318
Predikcija rezultata testa ... 319
Zadaci iz testiranja OO softvera ... 322
Zadatak 1: Uređivač teksta ... 322
Zadatak 2: Primena klasa ekvivalencije u unutar klasnom testiranju ... 331
Zadatak 3: Testiranje redefinisanih metoda ... 334
Zadatak 4: Nasleđivanje i Yo-Yo graf ... 335

Testiranje konkurentnih programa ... 338

Komunikacija niti ... 338
Sinhronizacija niti ... 342
Testiranje i otklanjanje grešaka u višenitnim programima ... 343
Praćenje, testiranje i ponovno izvršavanje za brave ... 344
SYN-sekvence za brave ... 347
Detekcija zastoja ... 348

Zadaci sa testiranjem konkurentnih programa 350
Zadatak 1: Primena Lockset algoritma 350
Zadatak 2: Deljena promenljiva 351
Zadatak 3: Kontrolor reprodukcije 353
Zadatak 4: Java VM detekcija zastoja 354
Testiranje grafičkih korisničkih interfejsa 356
Manuelno testiranje GUI 357
Slučajno (stohastičko) testiranje 358
„Nasnimi i pusti“ (Capture and Replay) tehnika 358
Testiranje zasnovano na modelu GUI 360
Graf toka događaja 364
Prediktori za testiranje GUI 365
Zadaci sa testiranjem GUI 366
Zadatak 1: Konstruisanje FSM za GUI tekst editora 366
Zadatak 2: Konstruisanje EFG za iskačući prozor 370
Zadatak 3: Krug i kvadrat 372
Literatura 375

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.

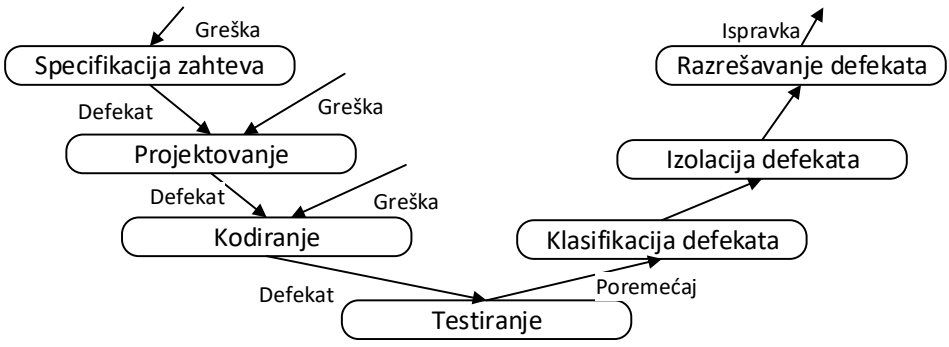
Testiranje softvera je, prema Glenfordu Majersu (*Glenford Myers*), proces izvršavanja programa sa osnovnim ciljem nalaženja defekata. U širem smislu, radi se o ispitivanju softverskog proizvoda ili servisa sa ciljem objektivnog utvrđivanja kvaliteta. Testiranje softvera je deo šire oblasti osiguranja 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.

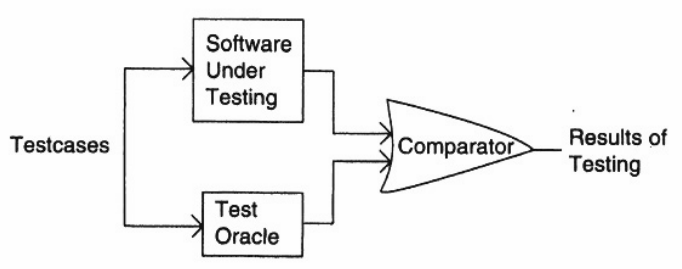


Slika 1. Životni ciklus softvera

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...]