Programski jezici i strukture podataka

15

TESTIRANJE PROGRAMA

Testiranje programa

- Aktivnost kompleksnija od kodiranja
- Mnoštvo metoda
- Nema univerzalnog pristupa
- Ne može se dokazati da program nema grešaka
- Program koji nije testiran „nije završen“
- Nakon svake ispravke programa sledi novo testiranje

Testiranje programa

- Testiranje programa je jedna od najskupljih aktivnosti u toku njegovog životnog ciklusa.
- Sprovodi se u toku prvobitne realizacije gde može da učestvuje čak sa 30% do 40% vremena, i u fazi eksploatacije i održavanja i to prilikom svake modifikacije programa.
- Testiranjem se utvrđuje prvo kako program obavlja posao za koji je namenjen, a zatim i kako se ponaša u realnom okruženju.
- Testiranje se može definisati kao provera:
    1. usklađenosti implementacije (realizacije) programa sa njegovom specifikacijom i
    2. njegovog ponašanja u eksploatacionim uslovima.

Testiranje programa

- Provera usklađenosti - povezuje se sa osobinom tzv. **korektnosti** (pouzdanosti) programa koja se, sa svoje strane, definiše kao mera u kojoj program zadovoljava specifikaciju.
- Provera ponašanja u eksploatacionim uslovima - povezuje se sa **robustnošću** programa pod kojom se podrazumeva imunost na razne poremećaje, a prvenstveno na neregulame, pa i neočekivane, ulazne podatke.
- Ovde se odmah napominje da se pod ulaznim podacima u ovom tekstu podrazumevaju ne samo ulazni podaci u uobičajenom smislu nego akcije nad programom u najširoj interpretaciji (dakle, i pritisci na tastere ili dejstva mišem), kao što se pod izlaznim podacima podrazumevaju ne samo podaci nego i bilo kakva druga reakcija programa na ulaz.

Važni termini

- U toku procesa realizacije programa prave se ***greške*** (engl. error).
- Rezultat greške pojavljuje se u programskom kodu kao ***defekt*** (fault).
- U praksi se termini "greška" i "defekt" najčešće smatraju sinonimima, iako strogo gledano oni to nisu.
- Kada se u toku izvršenja programa naiđe na defekt dolazi do ***otkaza*** (failure) koji može, ali ne mora biti momentalno primećen.
- Eksplicitno manifestovanje otkaza zove se ***incident***.

Važni termini

- Otkrivanje defekata u programu najčešće se obavlja na bazi nekih ulaznih vrednosti, konkretnih ili uopštenih.
- Primerak skupa ulaznih vrednosti kojim se program može pokrenuti i koji je uvršten u test nosi nazi **testna stavka** (test case).
- Skup testnih stavki zove se **test skup** ili test set.

Pristupi testiranju

- Destruktivni (dokazati da program ima grešaka)
- Konstruktivni (dokazati da program nema grešaka)

- Prethodna dva pristupa su međusobno suprotstavljena

Preporuke

- Test mora da obuhvati kako validne i očekivane uslove tako i invalidne i neočekivane
- Test treba da obezbedi odgovor na dva pitanja:
  - (a) da li program radi ono za šta je namenjen i
  - (b) da li program možda radi i nešto što je nepredviđeno (i nepoželjno)
- Izbegavati ad hoc testove ("Ubaci par ulaznih podataka da vidimo šta će da se desi"), posebno nedokumentovani, jer se ne mogu ponoviti niti daju bilo kakvu garanciju da će program u najvećem broju slučajeva davati korektan izlaz, odn. da će biti kvalitetan
- Nikako ne vršiti testiranje pod psihoioškom pretpostavkom da nema defekata.
- Praksa pokazuje da je potencijalna gustina grešaka najveća u segmentima u kojima su već locirane.

Destruktivni pristup testiranju

- Destruktivnim testovima ne može se dokazati da je program korektan, nego samo to da ima defekata (kako se među programerima govori, "svaki program ima grešaka dok se ne dokaže suprotno, što je inače nemoguće").
- Glavni cilj destruktivnog testiranja: otkriti što više defekata uz najmanje troškove, tj. u najkraćem vremenu.

Strategije destruktivnog pristupa

1. Analiza programa
1. Analiza programskih segmenata (Code Inspection)
2. Simulacija izvršavanja programa (Walkthroughs)

2. Strategija bele kutije (White Box Strategy) nazvana još i strukturno testiranje
3. Strategija crne kutije (Black Box Strategy) ili funkcionalno testiranje

Strategija bele kutije

- Prekrivanje naredbi (Statement Coverage)
- Prekrivanje odluka (Decision Coverage)
- Prekrivanje uslova (Condition Coverage)
- Prekrivanje odluka-uslova (Decision-Condition Coverage)
- Prekrivanje višestrukih uslova (Multiple Condition Coverage).

Prekrivanje naredbi

- Metoda prekrivanja naredbi predviđa definisanje takvih ulaznih podataka koji obezbeđuju da se svaka funkcionalna naredba programa izvrši bar jedan put.
- Loše osobine:
  - Neki putevi (moguće i velik broj) neće biti testirani.
  - Programske odluke (testovi) će biti provereni slabo.

Prekrivanje odluka

- Metoda prekrivania odluka predviđa definisanje takvih ulaznih podataka da za svaki algoritamski korak odluke (testa) bar jedanput budu realizovana oba ishoda ("DA" i "NE").
- Pri tome, ne uzima se u obzir nikakva uzročno-posledična veza između pojedinih odluka (posmatraju se izolovano jedna u odnosu na drugu) niti se ulazi u struktura predikata koji je u osnovi koraka odluke.
- Rezultati koji se ostvaraju metodom prekrivanja odluka nešto su - ali ne mnogo - bolji, samim tim što prekrivanje svih odluka za posledicu ima prekrivanje većeg dela naredbi.

Prekrivanie uslova

- Altemativna tehnika za prekrivanje odluka je tzv. prekrivanie uslova gde se pod uslovom podrazumeva svaki potpredikat u odlukama (u odluci 01 to bi bili potpredikati (A>1) i (B=0). Pri tome ne vodi se računa o tome da i svaka odluka kao celina ima oba ishoda.
- Suprotno prvom utisku, metoda prekrivanja uslova je otprilike iste efikasnosti kao i metoda prekrivanja odluka.

Prekrivanje odluka-uslova

- Test koji je u opštem slučaju precizniji od prethodnih, ali zahteva veći broj ulaza te je i kompleksniji i dugotrajniji je metoda prekrivanja odluka-uslova.
- Njena karakterisitika je da se ulazni podaci podešavaju tako da:
  - svaka odluka ima oba ishoda i
  - svaki pojedinačni uslov u okvirn odluke ima oba ishoda.

Prekrivanje odluka-uslova

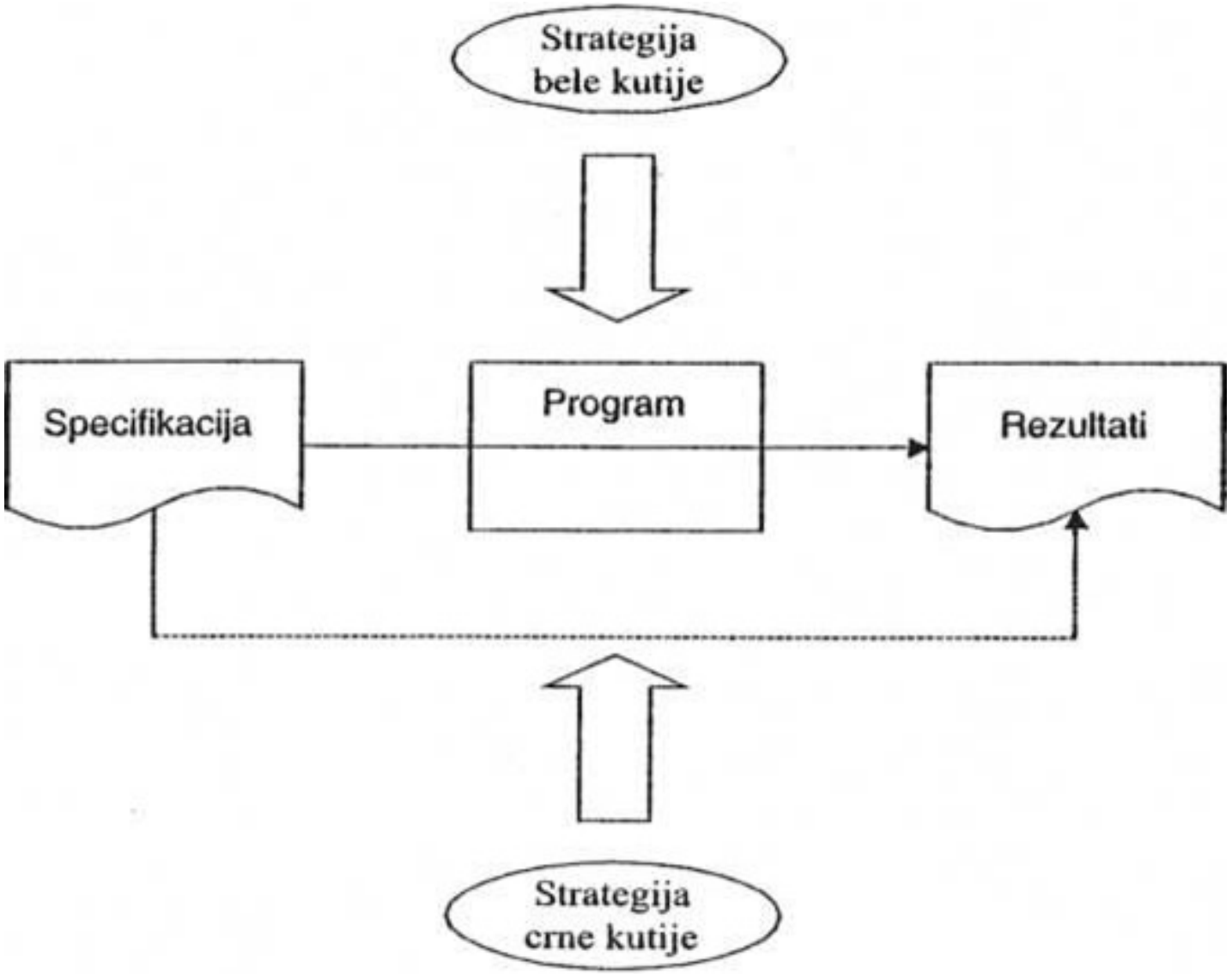
- Prekrivanje odluka - uslova je svakako precizniji test od prekrivanja odluka odnosno uslova posebno ali
- Prvo, zahteva veći broj ulaza
- drugo, određivanje tačnih vrednosti tih test ulaza nije ni približno tako jednostavno kao u prethodna dva slučaja.

Prekrivanje višestrukih uslova

- Za prekrivanje predikata najpreciznji, ali istovremeno i najsloženiji, postupak je metoda prekrivanja višestrukih uslova.
- Ona predviđa da se ulaznim podacima izazove prekrivanje svih kombinacija potpredikata u svim odlukama ponaosob.
- Ovo se postiže kreiranjem istinitosnih tablica za svaku odluku i izborom takvih ulaza da sve moguće kombinacije iz tablica budu ostvarene.

Strategija crne kutije

- Strategija crne kutije razlikuje se od prethodne po tome što se prilikom projektovanja ulaza ***ne konsultuje izvomi program već se test podaci kreiraju na osnovu specifikacije programa***
- Na taj način, program se posmatra kao "crna kutija" čiji detaljan sadržaj nije od interesa za testiranje.


Strategija
bele kutije
Specifikacija
Program
Rezultati
Strategija
crne kutije

Strategija crne kutije

- Prilikom definisanja ulaza od važnosti je da se proveravaju samo osobine predviđene specifikacijom.
- Specifikacija se tretira kao svojevrstan ugovor, a test treba da pokaže da li program odgovara tom "ugovoru".
- Zato specifikacija programa mora ispunjavati tri uslova, mora biti:
  - tačna,
  - razumljiva
  - potpuna

Strategija crne kutije (funkcionalno testiranje)

1. Metoda graničnih vrednosti
    1. Test graničnih vrednosti
        1. Totalni test graničnih vrednosti
        2. Parcijalni test graničnih vrednosti
    2. Test robusnosti

2. Metoda klasa ekvivalencije

Metoda graničnih vrednosti

- Metoda graničnih vrednosti bazira se na sledećim opštim postavkama:
  1. Ulazni podaci pripadaju nekom opsegu ograničenom minimalnom i maksimalnom vrednošću.
  2. Ulazni podaci su međusobno nezavisni.
  3. Otkazi ne nastaju simultanim uticajem više defekata. Ovaj uslov zove se engleski single fault assumption.

Metoda graničnih vrednosti

- Metoda je izgrađena na empirijski utvrđenoj činjenici da je verovatnoća greške najveća na graničnim vrednostima ulaznih podataka, odnosno u njihovoj okolini.
- Postoje četiri osnovne varijante:
  - Parcijalni test graničih vrednosti (Boundary Value Analysis)
  - Totalni test graničnih vrednosti (Worst Case Analysis)
  - Parcijalni test robustnosti (Robustness Testing)
  - Totalni test robustnosti (Worst Case Robustness Testing).

Testiranje složenih programa

- Pojedinačne funkcije mogu raditi ispravno same za sebe, ali treba videti kako radi kompozicija
- Sistemski test – testiranje programa u celosti

Problemi

- Osnovni problem kod testiranja složenih programskih sistema je *redosled* testiranja pojedinih modula.
- Pri tom, za testiranje svakog modula potrebno je imati na raspolaganju:
  - Modul (module) iz kojih se poziva testirani modul
  - Modul (module) koje testirani modul poziva

Problemi

- Najgori način za testiranje složenog sistema je njegovo testiranje kao celine.
  - Veoma je teško pratiti tokove podataka između modula ako ih oni ne razmenjuju neposredno i
  - još teže ustanovljenu grešku locirati, odnosno odrediti modul u kojem je ona nastala.
- Module, dakle, treba testirati pojedinačno.
- S druge strane, testiranje svakog modula zahteva postojanje svih njemu nadređenih (onih koji ga pozivaju) i svih njemu podređenih modula (onih koje on poziva).
- Ova dva zahteva su u koliziji.

Inkrementalno testiranje

- Naziv "inkrementalno" za ovaj pristup testiranju potiče upravo od činjenice da se već provereni moduli koriste za testiranje dragih te, tako, broj provereni' modula sistematski raste.
- Postoje dve metode za ovu vrstu testiranja i to:
  - Inkrementalno testiranje ***s vrha ka dnu*** (engl. top-down), gde testiranje počinje od glavnog modula i izvodi se u redosledu od nadređenih modula ka podređenim i
  - Inkrementalno testiranje ***sa dna ka vrhu*** (engl. bottom-up) kod kojeg se polazi od modula najnižeg nivoa i sprovodi u redosledu od podređenih ka nadređenim modulima.