1. **stranica:**

SPLITSKO-DALMATINSKA ŽUPANIJA

OŠ SPLIT 3

IME I PREZIME

SEMINARSKI RAD

NASLOV TEME

Mjesto, mjesec i godina

**2.stranica**

SPLITSKO-DALMATINSKA ŽUPANIJA

OŠ SPLIT 3

PREDMET: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

SEMINARSKI RAD

KANDIDAT: Ime i prezime

TEMA SEMINARSKOG RADA: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

MENTOR: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Mjesto, mjesec i godina

**3.stranica**

Sadržaj

**4. stranica**

UVOD

Testiranje programske podrške je proces traženja grešaka. U njemu su obuhvaćene sve aktivnosti potrebne za vrednovanje sposobnosti programa da izvrši pred njim postavljeni zadatak, kao i za tumačenje dobivenih rezultata. Za razliku od većine materijalnih (fizičkih) sustava kod kojih do kvara dolazi zbog grešaka nastalih u fazi proizvodnje ili fazi eksploatacije, većina neispravnosti koje se pojavljuju tijekom izvođenja programa su posljedica grešaka nastalih u fazi dizajna. Programske pogreške gotovo će uvijek egzistirati u bilo kojem programskom modulu umjerene veličine, ne zbog nepažljivosti ili neodgovornosti programera, nego zato što je programska podrška općenito veoma kompleksna. Za kompleksne sustave može se sa sigurnošću reći da greške nastale prilikom dizajna, najvjerojatnije neće nikada biti sve u potpunosti uklonjene.

Bez obzira na ograničenja, testiranje je integralni dio razvoja programske podrške. Ono se primjenjuje u svim fazama razvoja programske podrške. Obično se više od 50% vremena potroši na testiranje. Testiranje programske podrške najčešće se izvodi zbog: poboljšanja kvalitete, provjeravanja i potvrde ispravnosti, procjene pouzdanosti.

TESTIRANJE

**Što je to testiranje?**

Testiranje programske podrške tj. softwarea je formalni proces koji se izvodi sa specifičnim timom za testiranje, kojim se ispituju jedinice softwarea ili cjelokupni softwareski paketi izvršavanjem programa na računalu, sa svrhom otkrivanja grešaka i nepravilnosti u software-u, evaluaciji njegove kvalitete i na kraju poboljšanja software-a.

Testiranje software-a je proces izvršavanja programa s ciljem pronalaženja grešaka. Ono se sastoji od dinamičke verifikacije ponašanja programa na konačnom skupu test slučajeva, prikladno izabranih iz obično beskonačne domene izvršavanja, prema očekivanom ponašanju.

Tehnike testiranja uključuju proces izvršavanja aplikacije u cilju pronalaženja s softwareskih grešaka, ali nisu ograničene samo na to.

Testiranje softwarea se također može navesti kao proces provjere i verificiranja programa odnosno proizvoda koji:

Ispunjava poslovne i tehničke zahtjeve kojima je napravljen njegov dizajn i razvoj,

Radi kako treba,

Može biti implementiran sa tim istim karakteristikama.

U razvojnom ciklusu softvera sve je značajniji zadatak *Testiranje softwarea* (TS) ili *Verifikacija i Validacija* (V&V) koji treba osigurati zahtijevani nivo povjerenja u ispravnost softwarea.

**Zašto testiramo?**

Zbog današnje raširenosti i sveprisutnosti informatike, tehnologije kao takve i samog načina proizvodnje softwarea, iznimno je važno testirati takve proizvode jer greške u samom proizvodu mogu prouzročiti niz problema za tvrtku koja se bavi njihovom proizvodnjom i klijenata koji koriste taj softver.

Neki od tih problema zadiru u financijski spektar jer potencijalno mogu prouzročiti ogromne financijske gubitke. Zamislimo situaciju da smo razvili neki software i nakon izdavanja na tržište se pokaže da neki ključni dijelovi softwarea ne rade ispravno. Ta situacija bi prouzročila velike financijske gubitke, a da ne govorimo o još jednom vrlo važnom aspektu, kao što je gubitak ugleda tvrtke. U određenim slučajevima, kao što su na primjer softwarei za upravljanje zračnog ili svemirskog prometa, eventualne greške u sustavu mogu dovesti do gubitaka ljudskih života.

Iz tih razloga se zadnjih godina što je više moguće povećava ulaganje u procese testiranja sustava.

**Tko testira?**

Poseban tim ljudi unutar svake tvrtke koji se trudi predvidjeti sve moguće probleme u softwareu i testiranjem provjeriti dali se ti problemi zaista javljaju. Do 1980. su developeri ujedno i testirali, no nakon te godine se testiranje postepeno odvojilo kao zasebna profesija. Ovisno o periodu i različitim ciljevima testiranja su se uspostavile različite uloge: manager, test lead, test designer, tester, automation developer i test administrator. Poželjne vještine kod člana testerskog tima su npr: kreativno razmišljanje, radoznalost, dobro prosuđivanje, detaljnost kod pisanja dokumentacije, timski rad.

I sami korisnici su testeri.

**Ciljevi testiranja**

Dva su osnovna i najveća cilja testiranja: verifikacija i validacija (V&V). Verifikacijom tražimo odgovor na pitanje: Gradimo li proizvod DOBRO. Odgovor i potvrdu na to pitanje treba dati software svojom dokumentacijom. S druge strane, validacijom tražimo odgovor na pitanje: gradimo li DOBAR proizvod. Odgovor daje sam software, ako software zaista radi ono što korisnik želi.

Verifikacija je provjera odgovaraju li rezultati etapa razvoja očekivanim rezultatima pojedine etape. Validacija provjerava da je sustav dao očekivane rezultate kao funkciju ulaza.

Nadalje, unutar procesa V&V se primjenjuju dvije tehnike provjere i analize programskog sistema: kontrola softwarea i testiranje sustava.

Kontrola softwarea se odnosi na analizu statičkog prikaza sustava kako bi se otkrili problemi (statička verifikacija). Ono uključuje analizu i provjeru predstavljenog sustava kroz specifikaciju zahtjeva, dijagrame oblikovanja i izvorni kod programa.

Testiranje sustava se odnosi na ispitivanje i promatranje ponašanja softwareskog proizvoda (dinamička verifikacija). Ono uključuje implementaciju sustava sa test podacima, ispitivanje izlaznih rezultata i provjera da li se ponašanje sustava odvaja na način kako je to zahtijevano.

VRSTE I FAZE TESTIRANJA

Postoje razne vrste testiranja.

**Razvojno testiranje** provodi razvojni tim tijekom razvoja softwarea.

**Testiranje izdanja** obavlja posebni tim za testiranje unutar ssoftwareske kuće uoči isporuke novog izdanja softwarea.

**Korisničkim testiranjem** bave se sami korisnici tijekom isporuke softwarea.

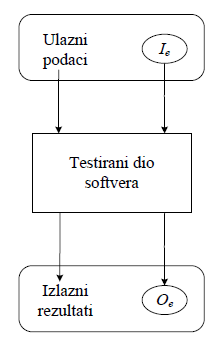
Svaka vrsta testiranja s prve razine podjele dalje se dijeli na podvrste.

**Testiranje dijelova softvera**

Da bi testiranje dijela softwarea bilo uspješno, važno je znati odabrati test podatke.

*Funkcionalno testiranje (black-box testing)*

Test-primjeri izvode se isključivo iz specifikacije dotičnog dijela softwarea. Ili, drukčije rečeno, software se promatra kao „crna kutija“ o kojoj jedino znamo da bi ona (prema specifikaciji) za zadane ulaze trebala proizvesti zadane izlaze. Zbog postojanja grešaka, ulazi iz određenog (nepoznatog) skupa Ie izazvat će neispravno ponašanje, koje ćemo prepoznati po izlazima iz skupa Oe (slika 2).



Funkcionalno testiranje

U postupku testiranja nastojimo izabrati ulaze za koje postoji velika vjerojatnost da pripadaju skupu Ie. Izbor takvih test primjera najčešće se zasniva na iskustvu softwareskog inženjera. Ipak, moguć je i sistematičniji pristup zasnovan na podjeli skupa svih mogućih ulaznih podataka (domene) na klase. Ovdje riječ „klasa“ nije upotrjebljena u smislu objektno – orijentiranog pristupa, nego u smislu matematičkog pojma „klasa ekvivalencije“. Očekujemo da će se program ponašati slično za sve podatke iz iste klase. Biramo barem po jedan test primjer iz svake klase. Također je dobro isprobati „rubove“ klasa.

*Strukturalno testiranje (white-box testing)*

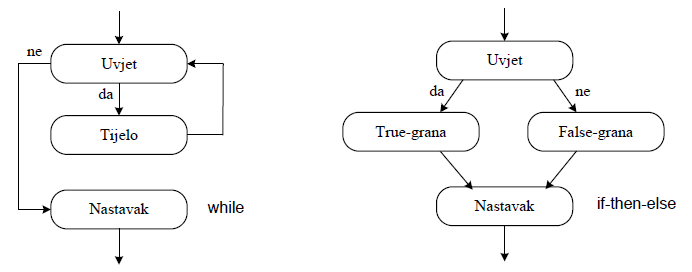
Dodatni test-primjeri izvode se na osnovu poznavanja interne strukture testiranog dijela softwarea i korištenih algoritama. Dakle služimo se analizom programskog koda dotičnog dijela softwarea da bi odabrali dodatne test primjere, to jest da bi bolje podijelili domenu na klase.

*Testiranje ekvivalentnim particijama*

Često ulazni podaci i izlani rezultati pripadaju različitim klasama gdje su svi članovi srodne klase (pozitivni brojevi, negativni brojevi...). Svaka od ovih klasa je jedna ekvivalentna particija gdje se program ponaša na jednak (ekvivalentni) način za svakog člana klase, stoga testni slučaj treba odabrati iz svake particije. Na primjer, imamo program koji za određen unos osobnih podataka daje određenu osobu. Osoba ima više različitih podataka i mi želimo za svaki od tih podataka dobiti isti rezultat. Pod ekvivalentne particije smatramo ime, prezime, OIB, adresu itd.

*Testiranje po putovima*

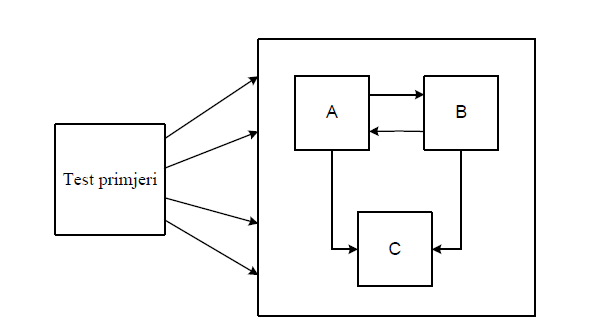
Riječ je o posebnoj vrsti strukturalnog testiranja, gdje se test primjeri biraju tako da se isproba svaki „nezavisni put“ kroz dijagram toka kontrole testiranog dijela softwarea. Dakle najprije se mora nacrtati dijagram toka kontrole dotičnog dijela softwarea. Dijagram se crta tako da se naredbe *while* i *if - then - else* prikažu obrascima (Slika 3), a ostale naredbe se prikažu svaka po jednim čvorom. Uočavaju se čvorovi koji predstavljaju „početak“ i „kraj“ toka kontrole. Testiranje po putovima zahtijeva da se konstruiraju takvi test primjeri koji će isprobati svaki od nezavisnih putova kroz dijagram. Nezavisni put je onaj koji ide od „početka“ do „kraja“ i prolazi bar jednim novim lukom



Obrasci za crtanje dijagrama toka kontrole

*Testiranje integracije*

Testiranje integracije provodi se nakon što se manji dijelovi softwarea udruže u veću cjelinu. Cilj testiranja je otkriti greške koje nastaju zato što dijelovi na pogrešan način koriste međusobna sučelja. Kao što je prikazano na Slici 4 test primjeri se primjenjuju na cjelinu, a ne na pojedine dijelove.



Primjena test primjera kod testiranja integracije

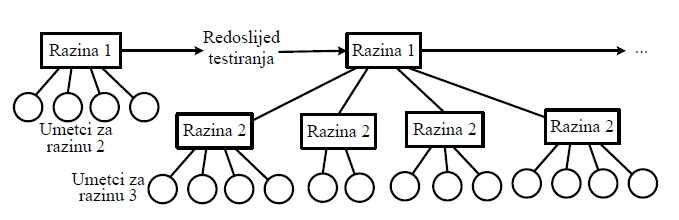
Ova vrsta testiranja osobito je važna za objektno-orijentirane sustave. Naime, objekti su definirani preko svojih sučelja, te se ponovo upotrebljavaju u kombinaciji s drugim objektima u drugim sustavima – tada nastaju greške čiji uzrok je interakcija objekata, a ne rad pojedinog objekta.

Testiranje integracije posebno je teško zato što se greške mogu pojaviti samo u iznimnim situacijama (veliko opterećenje sustava) ili pak na neočekivanim mjestima (u sasvim drugom dijelu softwarea).

*Top-down i bottom-up testiranje*

Softwareski sustavi obično sadrže nekoliko razina integracije. Na primjer: najmanje jedinke udružuju se u module, koji se spajaju u pod-sustave, koji čine sustav. Zbog više razina integracije, testiranje integracije mora se ponoviti više puta. Pritom redoslijed testiranja može biti jedan od sljedeća dva.

Top-down: najprije se testira integracija na najvišoj razini, zatim na nižoj razini, … , na kraju na najnižoj razini. Pritom se nepostojeći ili netestirani dijelovi s niže razine zamjenjuju s nadomjescima - takozvanim stub-ovima. Stub ima isto sučelje kao odgovarajući dio ali vrlo ograničenu funkcionalnost.



Testiranje integracije u redoslijedu top-down

Bottom-up: najprije se testira integracija na najnižoj razini, zatim na idućoj višoj razini, …, na kraju na najvišoj razini. Nisu potrebni stub-ovi jer za svaku iduću razinu integracije već imamo implementirane i testirane sve sastavne dijelove. Ipak, potrebni su test driver-i koji pozivaju manje dijelove softwarea i simuliraju njihovu okolinu.

Prednost top-down redoslijeda je da on ranije otkriva eventualne greške u arhitekturi. također, psihološka prednost top-down-a je da se relativno rano dobiva sustav koji radi (doduše s ograničenom funkcionalnošću) i koji se može pokazivati. Mana top-down-a je da se troši vrijeme na izradu stub-ova.

Prednost bottom-up redoslijeda je da se lakše uključuju gotove komponente. Također, prednost bottom-up-a je da arhitektura ne mora biti dovršena sve do zadnjeg časa. Mana bottom-up-a je da se troši vrijeme na izradu test driver-a.

ZAKLJUČAK

Testiranje programske podrške je svaka aktivnost usmjerena na procjenu osobina ili sposobnosti programa te određivanje onoga što vodi zahtijevanim rezultatima. Teškoće u razvoju programske podrške proizlaze iz složenosti programskih rješenja. Testiranje je više od običnog debugiranja. Cilj testiranja nije samo u provjeri, radi li program ispravno, nego i u procjeni kvalitete sigurnosti, procjeni pouzdanosti ili procjeni performansi gotovih programskih rješenja. Testiranje se izvodi u svim fazama životnog vijeka programa. Mogu se testirati gotova programska rješenja ili samo pojedini dijelovi.

Tehnike koje se najčešće koriste kod testiranje su testiranje crnom kutijom, testiranje bijelom kutijom, testiranje stresa, testiranje izdržljivosti, kao i razni programi za testiranje performansi. Kako se zbog složenosti gotovih programskih rješenja nikad sa sigurnošću ne može kazati da su otkrivene i otklonjene sve greške unutar programske podrške, a samo testiranje predstavlja trošak, pitanje je kad prestati s testiranjem. Zato se može zaključiti da je testiranje programske podrške trgovina između raspoloživih novčanih sredstava, vremena i željene kvalitete.

LITERATURA

http://www.croz.net/index.php/hrv/services/poslovna\_analiza/testing

http://www.scribd.com/doc/27426792/Testiranje-Softvera-Rad

http://www.compaid.com/caiinternet/ezine/cost\_of\_quality\_1.pdf

http://web.efzg.hr/dok//inf/pozgaj/pisani%20materijali/T09%20Testiranje.pdf

http://en.wikipedia.org/wiki/Software\_testing