Osnovne aktivnosti za proveru pouzdanosti softverskog koda

U radu je dat istraživački i inženjerski pristup tehnologiji ostvarivanja zahtevane pouzdanosti softvera, u fazi provere ili dinamičkog testa pouzdanosti programa, a razvrstan je na bazi predloga kompletne strukture sa osnovnim aktivnostima ispitivanja. Pomoćno softversko okruženje ispitnih aktivnosti predstavljeno je idejno. Analizirani su uslovi i mogućnosti tehničke pripreme i provođenja ispitivanja pouzdanosti softvera.

Uvod

Softverski algoritam, ili, u tradicionalnom izražavanju, računarski program, proizvod je iz određene faze životnog veka softvera, koji se, ponekad, može prihvatiti kao prototip datog softverskog sredstva. Kraće, najčešće se govori o softverskom kodu (kao u naslovu) ili programu kao softverskom proizvodu. Formiran softverski kod, u izvornom ili mašinskom kodu (programskom jeziku), već može da bude podvrgnut eksperimentalnom ispitivanju dinamičkog karaktera u pogledu provere njegove pouzdanosti. Ako je ovo ispitivanje deo konstruktorskih ili prijemnih ispitivanja TMS, može se smatrati dinamičkim testom. Analogno poznatim i standardizovanim postupcima provere pouzdanosti uređaja [1], mogu da se definišu osnovne aktivnosti za dinamičku proveru pouzdanosti softverskog koda ili programa. Pouzdanost je, uostalom, dinamičko svojstvo svakog TMS, jer se definiše u vremenskoj bazi za određene uslove eksploatacije, pa je i njena provera takvog karaktera.

Zahtevi u vezi pouzdanosti softvera mogu da potiču iz oblasti elemenata kvaliteta ili ITOB-a softvera [2, 3], odnosno TMS. Kompromis su želja (potreba) i raspoloživih podataka (mogućnosti) pri ugradnji pouzdanosti softvera i sistema. Zavisno od namene softvera, mogu da budu iskazani kao parametri

srednjeg vremena do (između) otkaza ili učestanosti otkaza, ili preko uspešnosti funkcionisanja (funkcije pouzdanosti). Inače, pouzdanost softvera ima istu teorijsku osnovu kao i pouzdanost drugih TMS. Razlika je najčešće u mehanizmu nastajanja otkaza, gde se nedostatak, koji može da bude uzrok slučajnog otkaza softvera, pripisuje čoveku i raspoloživoj tehnologiji i tehnici izrade. Osnovne aktivnosti za proveru pouzdanosti softverskog koda, koje se dalje razmatraju, predmet su originalnog Plana i programa provere pouzdanosti softvera.

Softverski otkazi

Registrovanje fenomena otkaza tokom dinamičkog testa omogućava dobijanje podataka za donošenje odluke: da li produžiti eksperimente, da li odbaciti ili da li prihvatiti ispitivani softverski proizvod. Prema terminologiji poznatih standarda iz ove oblasti (MIL, IEC), otkazi koji se uzimaju u obzir za utvrđivanje postignute pouzdanosti nazivaju se relevantnim otkazima. Tehnologija izrade softvera jedinstvena je, susedne faze životnog veka nalaze se u iterativnom odnosu, aktivnosti pronalaženja otkaza (nedostataka) i njihovog otklanjanja (održavanja) stalno se uzajamno smenjuju. Treba reći, inženjerski pristup projektovanja softvera sadrži

stalnu analizu mogućih uzroka otkaza pri ugradnji pouzdanosti. U softverskom inženjerstvu nisu potrebne laboratorije koje bi se bavile kvalitativnom analizom pouzdanosti softverskih modula kao što je to slučaj za materijalne tehničke uređaje, već su i kvalitativni i kvantitativni pristupi analiza pri ugradnji pouzdanosti softvera stalno »u rukama« istog inženjera. Nedostatak — greška — otkaz čine, u izvesnom smislu, uslovljeni niz fenomena koji mora da se ima u vidu pri ugradnji i ispitivanju pouzdanosti softvera.

Matematičkim modelima pouzdanosti softvera ili, kraće, modelima pouzdanosti softvera [4], razvijanim od kraja šezdesetih godina ovog stoleća pa do današnjih dana, pretpostavljena je priroda nastajanja softverskih otkaza, tako da bi moglo da se sagleda buduće ponašanje softvera u eksploataciji. Cilj je najčešće bio omogućavanje prognoze pouzdanosti softvera, analogno prognozi pouzdanosti elektronskih uređaja, ali neki tipovi od njih omogućavaju pristup statističkom ispitivanju pouzdanosti softvera. Na primer, incidentni proces mogućih otkaza, za tzv. modele pouzdanosti definisane na ulaznom domenu, može da se pretpostavi na sledeći način: neka »n« ulaznih parametara (uzoraka) i »m« izlaznih parametara (odziva) može da se posmatra u funkciji $F: E \rightarrow Z$, gde je jedna ulazna količina za F vektor e u n -dimenzionalnom skupu E , a jedna izlazna vrednost $F(e)$ neka je vektor z u m -dimenzionalnom skupu Z . Preslikavanje funkcije F neka je definisano na skupu particija preslikavanja $F = \{F_1, F_2, \dots, F_L\}$. Tada svako parcijalno preslikavanje F_s , definisano na nekom podskupu X ulaznog domena, neka je domen definicije F_s , odnosno referiše se kao funkcija domena. Korespondentno, neko ukupno vreme ispitivanja T datog softvera »diskretizuje se«, pa se može posmatrati pod particijom, respektivno particiji F , kao skup $T = \{T_1, T_2, \dots, T_L\}$. U praktičnom izvođenju eksperimenata uslovi eksploatacije definišu se ulaznim do-

menom (ULAZNI PROFIL), koji je, u principu, statistički uzorak. Takođe, konačnom vremenu rada softvera korespondentan je diskretizovan skup T .

Razmatranje fenomena softverskih otkaza, njihovih priroda, mehanizma nastajanja i načina sprečavanja (otklanjanja), uključujući tu modelovanje pouzdanosti, zatim postavljanje zahteva u vezi pouzdanosti i realizovanje dinamičkog testa (provere) pouzdanosti, najviše imaju smisla za dve vrste softvera:

— softver kritične misije ili funkcije (upravljanje tehnološkim procesom, upravljanje vatrom oruđa, upravljanje letelicama, i sl.), i

— softver srazmerno velikog obima, na primer, izraženog u stotinama, ili hiljadama linija koda. Ovakva vrsta softvera obično ima značajnu prirodu namene (rad u realnom vremenu, visoku cenu transakcija, i dr.).

Definisanje skupa relevantnih otkaza u pripremi dinamičkog testa za ovakve vrste softvera, kako se da naslutiti, vrlo je značajna aktivnost povezana sa ukupnom metodologijom ispitivanja i zaključivanja. Posebno treba da se imaju u vidu dve neophodne grupe postupaka koje bi trebalo ugraditi u tokove eksperimenata provere pouzdanosti:

— manipulisanje sa INFORMACIONOM BAZOM EKSPERIMENTATA, i

— zaključivanje na osnovu OCENE POSTIGNUTE POUZDANOSTI.

Uz već pomenuti ULAZNI PROFIL, ovi postupci kao podrška aktivnostima ispitivanja mogu da budu automatizovani, odnosno da predstavljaju softversko okruženje eksperimenata [5]. Grafički predstavljen redosled osnovnih aktivnosti za proveru pouzdanosti softverskog koda (na sl. 1 i 2 u odeljku 3), sadrži, takođe, kontrole ograničavajućih parametara broja otkaza i vremena ispitivanja čije broječne vrednosti proizilaze iz pomenutih postupaka.

Određivanje relevantnih tipova otkaza ispitivanog softvera spada već među prve osnovne aktivnosti pripreme eksperimenata. Tome prethodi dobro upoznavanje i analiza ispitivanog softverskog koda (ISPITNI OBJEKT SOFTVERA). Tokom ove analize, koja je, u stvari, kvalitativna analiza pouzdanosti softvera, potrebno je razmotriti sledeće moguće tipove otkaza:

- logički;
- brojeani (numerički, podatkovni);
- vremenski, u smislu nesinhronizovanosti vremenskih intervala transakcija obrade;
- sigurnosni, koji može da proizađe od ugrađenih softverskih rešenja sigurnosti;
- izazvane od okruženja hardverskog i softverskog, i
- mešoviti.

Njihovi »pojavnici« oblici, pri tome, mogli bi da budu:

- pojedinačni;
- prolazni;
- učestali;
- vidljivi (stalni);
- skriveni, ili
- lažni (nešto što bi se moglo da izgleda kao otkaz, a, u stvari, nije otkaz. Na primer: lažni alarm).

Naravno, sa gledišta teorije pouzdanosti, samo tzv. primarni (neuslovljeni) otkaz može da bude deklarisan kao relevantni otkaz. Čin provere pouzdanosti softvera jeste namenjen tom cilju, ali tokom ispitivanja mogu da se otkriju i otklone i druge vrste nedostataka. To je tipično za tzv. klinrum (cleanroom) tehnologiju u softverskom inženjerstvu [6]. Očigledno, u analizi pouzdanosti softvera stalno treba da bude prisutno pitanje šta od eventualno otkrivenih nedostataka može u neodređenom (slučajnom) vremenskom trenutku, u budućnosti, da se manifestuje kao otkaz. Analiziranjem činjenica na ovaj način, uz zahteve da dati softverski proizvod treba pouzdano i bezbed-

no da obavi zadatu misiju, došlo se do postavljanja problema (ne)pouzdanosti softvera i osmišljavanja puteva njegovog rešavanja. Ispitivanje pouzdanosti softvera, statičko i dinamičko, otuda, pri projektovanju i izradi softvera, zauzima bar 50% od ukupnog radnog angažovanja stručnjaka [6].

Pregled osnovnih aktivnosti ispitivanja

Prema nacrtu životnog veka softvera [3], konstrukcijski ili prijemni tip ispitivanja, definisan kao dinamički test ili provera pouzdanosti, može se u okruženju računarskog sistema pojaviti u:

- fazi ispitivanja softverskog koda na laboratorijskom sistemu, i/ili
- fazi ispitivanja softverskog koda na ciljnom sistemu, gde inače treba da se ugradi ispitni objekt softvera.

Zavisno od količine raspoložive računarske opreme, eksperimenti, sa ciljem skraćivanja ukupnog vremena ispitivanja, mogu da budu izvođeni i na više istovetnih računarskih sistema. Dve grane u dijagramu aktivnosti DASP (na sl. 1) ukazuju na ovakve mogućnosti organizacije ispitnih aktivnosti. Dijagrami na sl. 1 i 2 prikazuju, inače, sve osnovne aktivnosti provere pouzdanosti softverskog koda po njihovom redosledu.

Ulazni statistički uzorak, pod simboličnim nazivom ULAZNI PROFIL, u aktivnostima pripreme navedenim u DASP-u i izvođenja eksperimenata navedenim na sl. 2 u dijagramu PSAK, može da bude ostvaren kao niz datoteka ranije pripremljenih podataka ili generisanjem, na primer, poznatom tehnikom simulacije Monte-Karlo. U slučaju izvođenja eksperimenata na laboratorijskom računarskom sistemu ulaznim profilom u potpunosti se definišu uslovi eksploatacije u smislu pouzdanosti. Kada se eksperimenti organizuju na ciljnom računarskom sistemu, ULAZNI PROFIL može da bude proširen nekim incidentnim slučajnim događa-

Sl. 1 — Dijagram aktivnosti eksperimentalne provere pouzdanosti softvera ((simboli po JU: A.FO.004)

Sl. 2 — Dijagram aktivnosti kontrole rada IOS-a

jima koji potiču od prirodnog okruženja ispitnog objekta softvera. Tipiski, eksperimenti sa laboratorijskim sistemom bliski su tzv. »eksperimentima tipa crne kutije«, a sa ciljnim sistemom mogu biti bliski tzv. »eksperimentima tipa bele kutije«.

Upoznavanje rezultata iz dokumentacije o već ranije izvedenim ispitivanjima na softveru koji se ispituje, simbolički nazvanim ISPITNI OBJEKT SOFTVERA, spada u prve aktivnosti ispitivanja po dijagramu DASP. Među prve aktivnosti ispitivanja trebalo bi da spada i priprema ULAZNOG PROFILA, a bolje je ako je on pripremljen tokom projektovanja i izrade softvera koji treba da se proverava i u vezi zahteva pouzdanosti. Ako je Monte-Karlo simulacija pri tome izabrana kao radna tehnika, onda jedan od pristupa može da bude sledeći [5]: neka je zadatak da se generiše niz pseudoslučajnih brojeva sa izabranom funkcijom gustine raspodele $f(x)$, definisane u nekom intervalu (A,B) koji odgovara ulaznom profilu. Prvo, pomoću niza pseudoslučajnih brojeva ξ_i, η_i , ravnomerno raspoređenih u intervalu $(0,1)$, treba da se transformišu razmere kontinualne krive $f(x)$ na jedinični kvadrat, kao na sl. 3. Transformacija razmera dobija se po relacijama, za nove koordinatne ose (u, v) ,

$$u = \frac{x-A}{B-A};$$

$$x = A + (B - A) \cdot \xi_i;$$

$$v = \frac{f(x)}{f(x)_{\max}} = \frac{f[A + (B - A) \cdot \xi_i]}{f(x)_{\max}}.$$

gde je $f(x)_{\max}$ najveća vrednost funkcije $f(x)$ u intervalu (A,B) . Onda, ako je među izračunatim vrednostima uspostavljen odnos $\eta_i < v$; tada generisana vrednost $A + (B - A) \cdot \xi_i$ pripada traženom nizu pseudoslučajnih brojeva. Od velike pomoći za ove pristupe simulaciji mogu da budu postojeće, matematičke i statističke biblioteke savremenih digitalnih računara.

Sl. 3 — Transformacija funkcije $f(x)$ u jediničnom kvadratu

Aktivnosti registrovanja podataka o eksperimentima pripadaju generisanju dokumenata koji se obično naziva Dnevnikom eksperimenata. Tu mogu da budu zabeleženi podaci, kako o trajanju i datumu ispitivanja, tako i o broju otkaza i trenucima otkaza. Obrada ovih »sirovih« podataka omogućava dobijanje podloga za zaključivanje. U automatizovanom vidu, slogovi podataka o eksperimentima i njihova obrada mogu da budu realizovani kao INFORMACIONA BAZA EKSPERIMENATA.

Tok eksperimenata, prikazan na sl. 2, pomoću dijagrama PSAK radi uprošćavanja ukupnog grafičkog prikaza aktivnosti ispitivanja, kao glavne aktivnosti, sadrži pobuđivanje (rad) ISPITNOG OBJEKTA SOFTVERA, stavljanjem u rad ULAZNOG PROFILA. Dijagram na sl. 2, u stvari, prikazuje ciklus ovih aktivnosti, kontrolisan graničnim vrednostima broja otkaza n_g i vremena ispitivanja t_g . Ovom kontrolom obezbeđen je, takođe, i prelaz na osnovnu aktivnost zaključivanja OCENA POSTIGNUTE POUZDANOSTI, u dijagramu na sl. 1. Izlaz iz ove osnovne

aktivnosti ili je nastavak eksperimenata (PSAK) ili je završetak ispitivanja (prihvatanje ili odbacivanje). Treba imati u vidu da aktivnosti koje pokriva simbolični naziv INFORMACIONA BAZA EKSPERIMENATA mogu da imaju širu ulogu: može da bude organizovana »istorijska« baza podataka za kasnije statističke analize.

Oцена postignute pouzdanosti

Cilj aktivnosti ocene postignute pouzdanosti je izvođenje određenih zaključaka o prihvatanju ili odbacivanju ispitivanog softverskog koda u pogledu pouzdanosti. S obzirom na podatke koji mogu da budu registrovani u dnevniku eksperimenata, ocene su tipično statističke prirode. Već razvijeni međunarodni standardi MIL i IEC, koji se odnose na elektronske uređaje, mogu i ovde da budu primenjeni [5, 7]. To su, pre svega, dokumenti MIL 781C, IEC 605-4, 5, 6 i 7, gde su standardizovani postupci, kako za sekvencijalnu proveru zahteva po pouzdanosti, tako i za tačkastu i intervalnu ocenu parametara pouzdanosti [1]. Matematički postupci su u navedenim dokumentima detaljno opisani sa primerima i planovima za testove.

U inženjerskoj praksi, a na osnovu pomenutih i drugih standarda, zahtevi u vezi pouzdanosti za elektronske uređaje i sisteme postavljaju se pod uslovom konstantnih parametara pouzdanosti (intenzitet otkaza λ , srednje vreme do/između otkaza m), tj. važenje eksponencijalne raspodele vremena rada do/između otkaza. To podrazumeva da je incidentni ulazni proces otkaza stacionarni Puasonov. Na osnovu toga, može se pretpostaviti da isti uslovi, u prvom stepenu približnosti, mogu da važe i za softverski kod koji je funkcionalni i strukturni deo elektronskih uređaja i sistema [7]. Pored toga, standardizovani postupci ocene postignute pouzdanosti (IEC 605-6) omogućavaju, takođe, utvrđivanje pretpostavke o konstantnom parametru λ .

Veličina statističkog uzorka, odnosno realizovani broj (relevantnih) otkaza u sekvencijalnoj statistici, predstavljaju parametre ocene na osnovu kojih se izvode zaključci u aktivnostima provere pouzdanosti. Zanimljivo je, međutim, da ocena parametra λ može da se dobije i kada u toku ispitivanja nije došlo do realizovanja otkaza. Za ovaj slučaj standard IEC 605-4 daje relaciju za tačkastu ocenu

$$\hat{\lambda} = \frac{1}{3T^*},$$

gde je T^* ukupno vreme trajanja eksperimenata. Uzima se da je softver visokopouzdanosti tehnički proizvod, pa mogu da budu ostvareni i ovakvi uslovi pri dinamičkom testu pouzdanosti.

Imajući u vidu osobine softvera u poređenju sa, na primer, elektronskim uređajima, postoji više povoljnosti u pogledu troškova, ukupno vremena ispitivanja i organizovanja statističkog uzorka određene veličine za korektnu statističku ocenu pouzdanosti softverskog koda. Prirodno je da:

- softverski kod je, praktično, neuništiv, jer se može obnoviti ako se raspolaze kvalitetnom i potpunom softverskom dokumentacijom;

- softverski kod ne stari tokom trajanja ispitivanja niti tokom stajanja, pa su mogućnosti organizovanja vremena ispitivanja vrlo elastične. To je jedan od uslova koje traže pomenuti standardi;

- vreme ispitivanja ne mora da bude neprekidno, tj. eksperimenti mogu da budu izvođeni s prekidima;

- veličina statističkog uzorka može da se definiše po potrebi, tj. može da se načini proizvoljan broj istovetnih kopija softverskog koda. Zna se, prema pomenutim standardima, da se na osnovu veće veličine uzorka skraćuje vreme sekvencijalne provere;

Dakle, aktivnosti ocene postignute pouzdanosti mogu da budu sprovedene i samo pomoću pomenutih postojećih standarda. Takođe, ove aktivnosti mogu

da budu uklopljene u Plan i program ispitivanja pouzdanosti elektronskih uređaja i sistema koji sadrže softverski kod kao funkcionalni podsistem. Pomenute povoljnosti u pogledu softvera ovakvu mogućnost samo potvrđuju.

Zaključak

Postavljene i analizirane osnovne aktivnosti i uslovi za proveru pouzdanosti softverskog koda, kao deo njegovog dinamičkog ispitivanja, ukazuju na mogućnosti za organizovanje ovakvog tipa ispitivanja softvera. Nema prepreka da ovakve provere budu u kompleksu ukupnih ispitivanja uređaja i sistema koji u strukturi imaju softverski kod. Standardizovana podloga postupaka za statističku ocenu i proveru pouzdanosti softverskog koda, takođe, postoji, a može se i proširivati po potrebi. Vodeći računa o predloženim aktivnostima u grafičkom pregledu okruženja

eksperimenata mogu da se preporuča sledeći naslovi sadržaja Plana i programa provere pouzdanosti softvera:

1. Predmet ispitivanja
2. Pripadnost sistemu
3. Zahtevi
4. Razvojna faza i okruženje eksperimenata
5. Prethodna ispitivanja
6. Metodološki pristup
7. Plan eksperimenata
8. Dnevnik eksperimenata
9. Ostali podaci o ispitivanju
10. Kadrovsko obezbeđenje
11. Izrada izveštaja.

Treba imati u vidu da postoje povoljni uslovi za organizovanje eksperimenata provere pouzdanosti softverskog koda, kako u pogledu troškova, tako i u pogledu organizovanja statističkog uzorka i rasporeda vremena ispitivanja. Takođe, izneti su predlozi o strukturi softverskog okruženja eksperimenata provere pouzdanosti softverskog koda.

Literatura:

- [1] Brkić D.: Ispitivanje pouzdanosti tehničkih sredstava NVO. Kvalitet i pouzdanost, 1988, 16.
- [2] Đokić D.: Osnovni elementi ITOB-a pri razvoju softvera automatizovanih sredstava NVO. Vojnotehnički glasnik, 1983, 2.
- [3] Đokić D.: Višekriterijumsko vrednovanje softvera po kvalitetu. Naučno-tehnički pregled, 1991, 10.
- [4] Đokić D.: Modeli pouzdanosti softvera. Kratka tehnička informacija 18, Beograd, 1983.
- [5] Đokić D.: Idejni projekt softverskog okruženja dinamičkog testa pouzdanosti softverskih proizvoda. Elaborat VTI, 1992.
- [6] Selb W. R., ...: Cleandrom Software Development: An Empirical Evaluation. IEEE Trans. on Software Engineering, 1987, Vol. 9-13.
- [7] Sandah H.: Reliability Demonstration Testing for Software. IEEE Trans. on Reliability, 1991, Vol. 40.