SQA Model – efikasnost i troškovi uklanjanja grešaka u softveru

Dženan Avdić, Ljubomir Lazić, Amel Kolašinac

Sadržaj — Da bi softverski proizvid zadovoljio potrebe korisnika on treba da zodovolji opšte kvalitete softverskog proizvoda. Organizacije koje razvijaju softver žele da zadovolje potrebe svojih korisnika i na taj način ostanu profatibilne. Da bi softverski proizvod zadovoljio standarde kvaliteta on mora biti detaljno testiran. U radu se daje preporuka kako da organizacije uspešno i na vreme isporuče softver, uz minimalne troškove. Dat je SQA (Software Quality Assurance) model koji objašnjava kako greške otklonjene u ranoj fazi manje koštaju od onih koje se otkriju kasnije.

***Ključne reči* — kvalitet softvera, SQA, troškovi uklanjanja greške, upravljane testiranja, zahtevi korisnika.**

Svrha ovog rada je da ukaže da otkrivanje grešaka u ranim fazama životnog ciklusa softverskog proizvoda dosta manje koštaju od onih koje se otkriju u kasnijim fazama [1,4,5]. Da bi softverski proizvod zadovoljio potrebe korisnika on treba da zodovolji opšte standarde kvaliteta softverskog proizvoda. Organizacije koje razvijaju softver žele da zadovolje potrebe svojih korisnika i na taj način ostanu profatibilne. Da bi softverski proizvod zadovoljio standarde kvaliteta on mora biti detaljno testiran. U radu se daje preporuka kako da organizacije koje se bave razvojom softvera uspešno i na vreme isporuče softver uz minimalne troškove. Pojam testiranja (pronalaženje i uklanje grešaka) danas se razlikuje od vremena sedamdestih godina kada je to značilo isključivo traženje grešaka. Sada ono obuhvata niz aktivnosti od planiranja, dizajniranja, izgradnje, održavanja, pa sve do sprovođenja testova. Potrebno je definisati načine merenja pojedinačnih faktora kvaliteta i definisati kriterijume njihovog zadovoljenja. Pronalaženje i uklananje grešaka u softveru je složen posao koji obuhvata testiranje jedinca, modula, podsistema, sistema i prijemno testiranje. Ciljevi testiranja su različiti. Ogledaju se kroz demonstraciju-proveru korektnosti dizajna i implementacije softvera, destrukciju-detekciju grešaka na nivou programiranja, evaluaciju-otkrivanju grešaka u specifikaciji zahteva, dizajnu i implementaciji i prevenciju-sprečavanju grešaka kroz faze razvoja softvera.

Ovaj rad delimično je finansiralo Ministarstvo nauke Republike Srbije, Projekat tehnološkog razvoja TR 13018.

Dž. Avdić, Državni Univerzitet u Novom Pazaru, Vuka Karadžića bb, 36300 Novi Pazar, Srbija (telefon: 381-65-2626347; e-mail: davdic@np.ac.rs).

Lj. Lazić, Državni Univerzitet u Novom Pazaru, Srbija; (e-mail: llazic@np.ac.rs).

A. Kolašinac, Državni Univerzitet u Novom Pazaru, Srbija; (e-mail: ifetahovic@np.ac.rs).

II. VERIFIKACIJA, VALIDACIJA I KVALIFIKACIJA

Tri aspekta osiguranja kvaliteta softverskog proizvoda spadaju pod:verifikacija, validacija i kvalifikacija. softverskih propusta. Iako su sprovedene različite studije kako bi se istražile tehnike statičkih ispitivanja za dizajn baze podataka, relativno malo pažnje je dato dinamičkom ispitivanju database aplikacija. Studija je, prema tome, pokrenuta da ispita efikasnost primene nekoliko popularnih tehnika testiranja softverskih aplikacija baze podataka. Stoga, predlažemo pristup testiranja koji preobražava ugrađenu SQL naredbu u aplikaciji baza podataka u proceduru programskog jezika opšte namene (GPL). Cilj preobražaja je uključiti semantičke SQL izjave tako da više test slučajevia može biti generisano kako bi se otkrile mane na koje se odnosi promena internih stanja baze podataka. Pristup omogućava test slučajeve koji će se brinuti o semantici i GPL naredbi i SQL naredbi koje će se generisati korišćenjem konvencionalnih "bela kutija" tehnika testiranja a time pomoći otkrivanju više grešaka koje se nalaze u aplikaciji baze podataka. Posebno se otkrivaju greške koje su se dogodile u nekim specifičnim internim stanjima baze podataka. Međunarodni standard STD 610.12-1990 IEEE (IEEE 1990) definiše ova tri aspekata osiguranja kvaliteta:

- Verifikacija - Proces evaluacije sistema ili komponenti kako bi se utvrdilo da li proizvodi u određenoj fazi razvoja zadovoljavaju nametnute uslove na početku faze.
- Validacija - Proces evaluacije sistema ili komponenti za vrijeme ili na kraju razvojnog procesa, kako bi se utvrdilo da li zadovoljava navedene zahteve iz ugla korisnika.
- Kvalifikacija - Proces koji se koristi kako bi se utvrdilo da li je komponenta sistema pogodna za operativnu upotrebu.

Prema IEEE definiciji, *verifikacijom* se pregledava dosljednost proizvoda koji je razvijen iz proizvoda razvijenog u prethodnim fazama. Kada se radi tako, ispitivač slijedi razvoj procesa i pretpostavlja da su sve prethodne faze razvoja završene ispravno, kako je prvobitno bilo planirano ili nakon uklanjanja svih otkrivenih nedostataka.

Provjera *validacijm* predstavlja interese kupca ispitivanjem stepena usklađenosti sa njihovim izvornim zahtevima. Sveobuhvatna provjera valjanosti sistema recenziranjem ima za cilj povećanje zadovoljstva kupaca tog sistema.

Kvalifikacija se fokusira na operativne aspekte, gdje je održavanje glavno pitanje. Softverske komponente koje su razvijene i dokumentirane u skladu s profesionalnim standardima, očekuje se da će biti mnogo lakše održavane nego one koje su nastale kroz improvizovano kodiranje. Iskusni planeri su potrebni da bi se utvrdilo koji od ovih

aspekata treba ispitati u svakoj aktivnosti osiguranja kvaliteta.

III. MODEL SQA KOJI EFIKASNO OTKLANJA GREŠKE UZ MINIMALNE TROŠKOVE

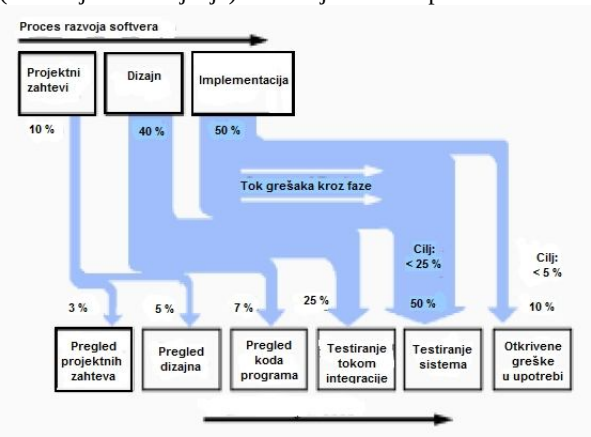
Model se bavi sa dva kvantitativna aspekta planiranja SQA preko detekcije otkaza kroz aktivnosti:

(1) Planiranja ukupne efikasnosti u procesu uklanjanja grešaka.
(2) Smanjenja ukupnih troškova u procesu uklanjanja grešaka.

Planiranje je potrebno da bi se u okviru projekta razvio efikasan proces testiranja softvera. Primjena ovog modela temelji se na tri vrste podataka, opisanog pod naslovima koji slede.

A. Porijeklo grešaka i njihova distribucija

Porijeklo grešaka (u fazi u kojoj su nastale) su distribuirane tokom razvojnog procesa, od početka projekta do njegovog kraja. Istraživanja koja su sprovedena od strane glavnih razvijatelja softvera, kao što su IBM i TRW, sažeta u radovima Boehm [6] i Jones [3], otkrivaju slične obrasce porijekla grešaka, njihovu distribuciju tokom životnog ciklusa kao i proces filtriranja (detekcije i otklanjanja) kao što je na Sl. 1 prikazano.



Sl. 1. Proces filtriranja (detekcije i otklanjanja) grešaka

Razvoj softvera pokazuje da se taj uzorak nije bitno promijenio u zadnje dve decenije. Karakteristična distribucija porijekla grešaka softvera na temelju pomenutih radova, prikazana je u tabeli 1.

B. Efikasnost uklanjanja grešaka

Pretpostavka je da bilo koja aktivnost osiguranja kvaliteta filtrira (engl. screens) određeni postotak postojećih nedostataka. Treba napomenuti da je u većini slučajeva, postotak uklonjenih grešaka nešto manji od postotka otkrivenih (detektovanih) nedostataka primenom neke test metode (oko 10% u skladu sa Jones [3]) tj. neefikasan je ili je neodgovarajući postupak “dibagiranja”. Preostale greške, neotkrivene ili nekorektno ispravljene su prošle u naredne faze razvoja. Aktivnosti osiguranja kvaliteta, koje slede, imaju zadatak da se izbore sa ovim kombinovanim greškama: onim koje su preostale nakon sprovedenih prethodnih aktivnosti osiguranja kvaliteta, zajedno sa "novim" napravljenim greškama u tekućoj fazi razvoja. Pretpostavlja se da efikasnost filtriranja grešaka svake aktivnosti osiguranja kvaliteta nije manja od 40% (tj. aktivnost uklanjanja grešaka treba da je najmanje 40% od prispelih defekata). Tipična prosečna efikasnost filtriranja grešaka različitih aktivnosti osiguranja kvaliteta, po fazama razvoja, utemeljene na radovima Boehm [6] i Jones [3], navedeni su u Tabeli 2.

Tabela 1 Karakteristična distribucija porijekla defekata softvera

Br.	Faza razvoja softvera	Srednji procenat porekla grešaka u razvojnoj fazi
1.	Specifikacija zahteva	15%
2.	Dizajn	35%
3.	Kodiranje (kod 30% i integracija 10%)	40%
4.	Dokumentovanje	10%

C. Troškovi uklanjanja nedostataka u softveru

Podaci o troškovima projekata, prikupljeni na više razvojnih projekata, pokazuju da trošak otkrivenja i uklanjanja nedostataka varira kroz faze razvoja, dok je porast tih troškova znatan (eksponencijalan) kako se razvojni proces odvija. Na primjer, uklanjanje kvara u design fazi može zahtijevati ulaganje od 2,5 radna dana; uklanjanje istog kvara može zahtijevati 40 radnih dana u toku testiranja prilikom isporuke softvera kupcu. Nekoliko anketa sprovedene su od strane IBM-a, TRW, GTE, Boehm [6] i drugih, za procjenu relativnih troškova za ispravljanje pogrešaka u svakoj fazi razvoja. Procjene efikasnosti softverskih alata osiguranja kvaliteta i relativnih troškova uklanjanja daje i Jones [3]. Iako su podaci o statistici i troškovima uklanjanja grešaka prilično rijetki, stručnjaci se slažu da je proporcionalno povećanje troškova uklanjanja grešaka ostala konstantna na bazi sprovedenih anketa u 1970-tim i 1980-tim. Primer tih prosečnih stopa efikasnosti filtriranja grešaka po fazama i aktivnostima osiguranja kvaliteta prikazane su u Tabeli 3.

Tabela 2: Prosek efikasnosti filtriranja (uklanjanja grešaka) u aktivnosti osiguranja kvaliteta

Br.	Aktivnosti osiguranja kvaliteta	Prosečna stopa efikasnosti filtriranja grešaka
1.	Pregled specifikacije zahteva	50%
2.	Ispekcija dizajna	60%
3.	Pregled dizajna	50%
4.	Ispekcija koda	65%
5.	Jedinični test	50%
6.	Jedinični test poslije ispekcije koda	30%
7.	Integracioni test	50%
8.	Sistem test/ test prihvatljivosi	50%
9.	Pregled dokumentacije	50%

Model je zasnovan na sljedećim pretpostavkama:

- Razvoj procesa je linearan i sekvencijalan, prati poznati vodopad model razvoja softvera.
- Broj "novih" grešaka se uvodi (pravi) u svakoj fazi razvoja. Za njihovu distribuciju, vidi tabelu 1.
- Pregled i testiranje programa, kao aktivnosti osiguranja kvaliteta, služe kao filteri uklanjajući određeni postotak ulaznih defekata i ostavljajući deo neotkrivenih da budu otkriveni u sledećoj razvojnoj fazi. Na primjer, ako je broj dolaznih grešaka 30, a efikasnost filtriranja je 60%, tada će biti 18 grešaka uklonjeno, a 12 grešaka će ostati i proći da bude otkriven u sljedećoj aktivnosti osiguranja

kvaliteta. Tipične stope uspešnosti filtriranja, za različite aktivnosti osiguranja kvaliteta, prikazani su u Tabeli 2.

■ U svakoj fazi, sabiraju se dolazni broj grešaka koji nije uklonjen u prethodnoj aktivnosti osiguranja kvaliteta zajedno s "novim" greškama uvedenim (stvorenim) u tekućoj fazi razvoja.

■ Troškovi uklanjanja defekata se izračunavaju za svaku aktivnost osiguranja kvaliteta množenjem broja uklonjenih nedostatka sa prosečnom cenom (troškom) uklanjanja jednog defekta u toj fazi (vidi tabelu 3).

■ Preostale defekte, nažalost, koji su dospeli do kupca, otkriće korisnik po znatno vrećoj jediničnoj ceni.

Tabela 3: Pprosečni relativni troškovi uklanjanja greška

Br.	Faze razvoja softvera	Prosečna relativna cena otklanjanja greške (jedinica cene - **cu**)
1.	Specifikacija zahteva	1
2.	Dizajn	2.5
3.	Jedinični test	6.5
4.	Integracioni test	16
5.	Sistem test/ test prihvatljivosti/ pregled dokumetacije	40
6.	Operativna upotreba (nakon isporuke kupcu)	110

U modelu, svaka od aktivnosti osiguranja kvaliteta predstavlja jedan stepen filtera, kao što je prikazano na slici 2.

Model predstavljaju sljedeće količine:

■ POD = Faza nastajanja Defekta (iz Tabele 1)

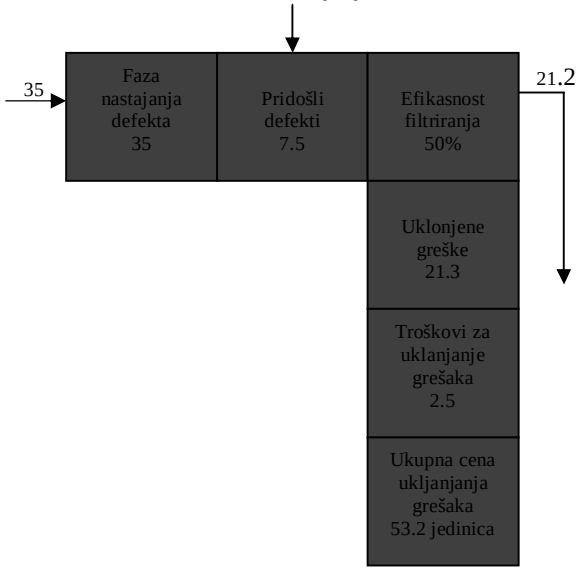
■ PD =Prošli (prebegli) Defekti (iz bivše faze ili aktivnosti osiguranja kvaliteta)

■ % FE =% Efikasnosti Filtriranja (također nazvan % „screening“ efikasnost) iz Tabele 2)

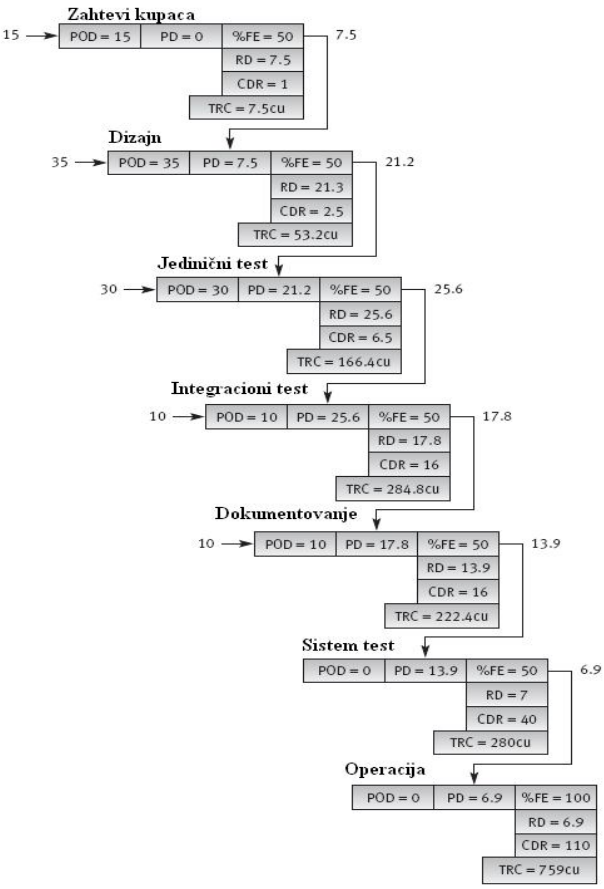
■ RD = Uklanjani Defekti

■ CDR = Troškovi Uklanjanja jedne greške (iz Tabele 3)

■ TRC = Totali troškovi uklanjanja: TRC = RD × CDR.



Sl. 2. Filter grešaka u fazi – efikasnost uklanjanja: primer



Sl. 3. Standardni model - plan procesa uklanjanje 100 grešaka

Sveobuhvatan plan osiguranja kvaliteta, za razliku od standardnog (sl. 3) ("sveobuhvatno filtriranje grešaka sistema") postiže slijedeće:

(1) dodju se dve aktivnosti osiguranja kvaliteta, tako da se oba izvode u dizajn fazi kao i u fazi kodiranja.

(2) poboljšava efikasnost "filtriranja" drugih aktivnosti osiguranja kvaliteta.

Slika 4. ilustruje kvantitativno sveobuhvatni model - plan procesa uklanjanje 100 grešaka. Nakon toga je izvršeno upoređivanje rezultata standardnog procesa obezbeđenja kvaliteta softvera naspram sveobuhvatnog plana otkrivanja grešaka u softveru.

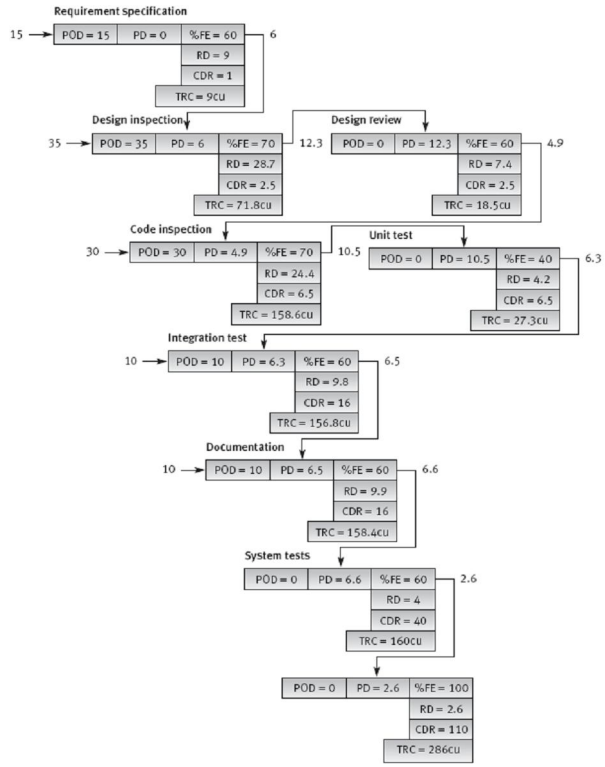
Najvažniji zaključci izvučeni iz upoređivanja su:

(1) standardni plan uspješno uklanja samo 57,6% (28,8 grešaka iz 50) od grešaka koji je nastao u fazi zahtjeva i dizajna, dok je za sveobuhvatni plan efikasnost 90,2%, prije početka kodiranja. Ovo se očekuje kao direktna posljedica intenzivnijeg postupka uklanjanje greška – povećani su napori testiranja koji sveobuhvatni plan uključuje.

(2) sveobuhvatni plan, u cjelini, je mnogo ekonomičniji od standardnog plana jer štedi 41% od ukupnih sredstava uloženih u uklanjanje greške u poređenju sa standardnim planom.

(3) U odnosu na standardni plan, sveobuhvatni plan doprinosi većem zadovoljstvu kupaca uz drastično

esmanjuje stop otkrivenih grešaka tokom redovne operativne upotrebe softvera (od 6,9% na 2,6%).



Sl. 4. Sveobuhvatni model - plan procesa uklanjanje 100 grešaka

IV. ZAKLJUČAK

Testiranje je jedana od najvažnijih aktivnosti u razvoju softvera i troši između 30 i 50% od ukupnih troškova razvoja softvera u skladu s mnogim studijama. Da bi omogućili dizajneru softvera da postigne veći kvalitet dizajna, bolji uvid u proces obezbeđenja kvaliteta kroz poboljšanje planova testiranja, ponuđen je odgovarajući SQA (Software Quality Assurance) model. U ovom radu opisan je predloženi SQA model koji omogućuje smanjivanje troškova u planu testiranja, kada trenutni plan (odabir tehnika verifikacije, validacije i kvalifikacije) ne može ispuniti zadati kvalitet softvera, u zadatom vremenu za testiranje pri čemu se troškovi uklanjanja grešaka svode na minimum.

Najvažniji zaključci izvučeni iz upoređivanja su:

(1) standardni plan uspješno uklanja samo 57,6% (28,8 grešaka iz 50) od grešaka koji je nastao u fazi zahtjeva i dizajna, dok je za sveobuhvatni plan efikasnost 90,2%, prije početka kodiranja. Ovo se očekuje kao direktna posljedica intenzivnijeg postupka uklanjanje greška – povećani su napori testiranja koji sveobuhvatni plan uključuje.

(2) sveobuhvatni plan, u cjelini, je mnogo ekonomičniji od standardnog plana jer štedi 41% od ukupnih sredstava uloženih u uklanjanje greške u poređenju sa standardnim planom.

(3) U odnosu na standardni plan, sveobuhvatni plan doprinosi većem zadovoljstvu kupaca uz drastično esmanjuje stop otkrivenih grešaka tokom redovne operativne upotrebe softvera (od 6,9% na 2,6%).

LITERATURA

[1] D. Galin, *Software Quality Assurance:From theory to implementation*, Pearson Education Limited, ISBN 0201 70945 7, 2004.

[2] A. Frost and M. Campo, " Advancing Defect Containment to Quantitative Defect Man", CrossTalk, December 2007.

[3] C. Jones, *Estimating Software Costs*. 2nd edition. McGraw-Hill, New York: 2007.

[4] Lj. Lazic, N. Mastorakis, " Optimizing Test Process Action Plans by Simulated Defect Removal Cost Savings", in the WSEAS Book "CONTROL, MODELLING and SIMULATION", Included in ISI/SCI Web of Science and Web of Knowledge, Istanbul, Turkey, 2009, pp. 280-287.

[5] Lars-Ola Damm, "Early and Cost-Effective Software Fault Detection", PhD Thesis, Blekinge Institute of Technology, SWEDEN, 2007.

[6] B. Boehm, *Software Engineering Economics*, Prentice Hall, Englewood Cliffs, NJ; 1981.

ABSTRACT

To enable software designers to achieve a higher quality for their design, a better insight into quality predictions for their design choices, test plans improvement using SQA (Software Quality Assurance) model is offered. In this paper we propose a model which enables to minimize the cost of switching between test plan alternatives, when the current choice cannot fulfill the quality constraints i.e. the time when defect repair costs are at their minimum.

SQA MODEL - DEFECT REMOVAL EFFECTIVENESS AND COST OF DEFECT REMOVAL

Dženan Avdić, Ljubomir Lazić, Amel Kolašinac