Troškovi osiguranje kvaliteta softvera

Zenaida Šabotić, Ljubomir Lazić, Dženan Avdić

Sadržaj **– Ovim radom pokazujemo glavne troškove osiguranja kvaliteta softvera. Akcenat je na troškovima detekcije i uklanjanja grešaka. Postoje softveri koji se razvijaju godinama i u koje je uloženo mnogo novca i vremena, ali oni nikada ne dobiju tržišnu valorizaciju zbog neusaglašenosti sa standardima kvaliteta softvera. Mi u radu dajemo načine povećanja efikasnosti povratka investicija testiranja. Velike softverske kuće koje se bave proizvodnjom softvera koriste slične aktivnosti. U radu dajemo tehnike za povratak investicija uloženih u proces testiranja (ROTI), koji je neophodan u procesu poboljšanja softverskog procesa.**

Ključne reči **— povratak investicija uloženih u proces testiranja (ROTI); kvalitet softvera; osiguranje kvaliteta softver;, troškovi kvaliteta**

I. UVOD

Softver ima neke specifičnosti koje uzrokuju probleme. Softver je apstraktni rezultat mentalnog procesa, on je nevidljiv, samo su apstraktni opisi različitih aspekata softvera kao podaci, funkcije, kontrole dostupne u obliku dijagrama ili tekstova. Softver je nematerijalan i samim time teško ga je meriti. Teško je nadalje upravljati nečim što ne možemo izmeriti. Nedostatak pouzdanih merila proizvoda i procesa otežava planiranje i kontrolu.

Kvalitet isporučenog softvera se ne može garantovati: proizvodnja softvera nije kvantificirana i značajno je niska, dok su troškovi razvoja i procene isporuke često nerealni. Iskustvo pokazuje da se zahtevi softvera često menjaju, važne analize se sprovode površno što uzrokuje promene zahteva i specifikacija softvera, prema nadolazećim potrebama, međutim, softver je lako menjati ali ga je teško menjati korektno. Kvalitet softvera posebno je važna tamo gdje se softver koristi za upravljanje i nadzor kritičnih sistema, tj. tamo gdje greške softvera ili ispad softvera iz rada može uzrokovati gubitak ljudskih života ili uzrokovati velike materijalne gubitke (vojna industrija, avio-industrija, istraživanje svemira, nuklearne elektrane). Sa stanovišta menadžmenta tri su osnovna međuzavisna parametra koje je potrebno kontrolisati tokom razvoja softvera: troškove, redosled izvođenja radova i kvalitet. Kako bi proizveli softver sa što nižim troškovima, i kako bi bili sposobni upravljati razvojem, razvoj softvera mora biti utemeljen na valjanoj inženjerskoj praksi.

Šta je kvalitet softvera?

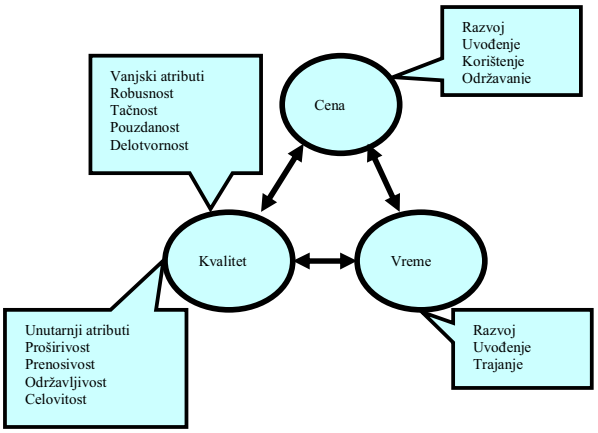
Ovaj rad delimično je finansiralo Ministarstvo nauke Republike Srbije, Projekat tehnološkog razvoja TR 13018.

Z. Šabotić, Državni Univerzitet u Novom Pazaru, Srbija; (e-mail: zsabotic@np.ac.rs).
Lj. Lazić, Državni Univerzitet u Novom Pazaru, Srbija; (e-mail: llazic@np.ac.rs).
Dž. Avdić, Državni Univerzitet u Novom Pazaru, Vuka Karadžića bb, 36300 Novi Pazar, Srbija (e-mail: davdic@np.ac.rs).

U našem radu pod kvalitetom softvera se smatra: "sposobnost za upotrebu ukupnog softverskog proizvoda".

II. MODEL TROŠKOVA KVALITETA SOFTVERA

U radu razmatramo TQM model (engl. Total Quality Management), što znači potpuno uključivanje svih funkcija i svih zaposlenih u preduzeću, dobavljača i kupaca na ostvarivanju visokog kvaliteta uz najniže troškove. U inženjerstvu kvalitet softvera, cena i vreme su tri međusobno zavisna faktora, kako je to prikazano na Sl. 1. Ako su dva od ovih faktora konstante, treći faktor nije moguće kontrolisati[1][2]. Ova tri elementa uzrokuju i najveće probleme, jer su troškovi i vreme isporuke softvera često iznad predviđenih veličina, a često softver ne zadovoljava zahteve korisnika. Međutim, zadovoljstvo korisnika je osnovni cilj osiguranja kvaliteta.



Sl. 1. Odnos kvaliteta, vremenena i cene

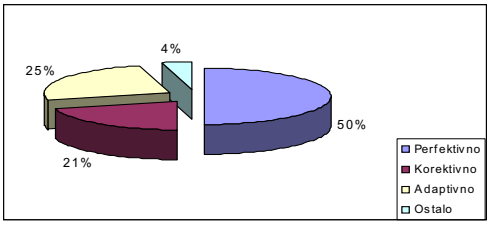
Održavanje softvera je najskuplja faza životnog ciklusa softvera, jer se tu prave i najveći troškovi koje treba smanjiti delujući po određenim modelima. Održavanje softvera definišemo sa četiri aktivnosti nakon što je program stavljen u upotrebu.

1) *Korektivno održavanje:*Nerezonski je pretpostaviti da se greške u velikim softverskim programima mogu otkriti u fazi testiranja.

2) *Adaptivno održavanje:*Događa se zbog čestih izmena koje se događaju u svakom aspektu kompjuterizacije.

3) *Perfektivno održavanje:*Održavanje softvera zadovoljava zahteve korisnika ali preporučuju se nove mogućnosti (modifikacije).

4) *Preventivno održavanje:*Kada se proces razvoja softvera menja kako bi se poboljšala mogućnost održavanja i pouzdanosti

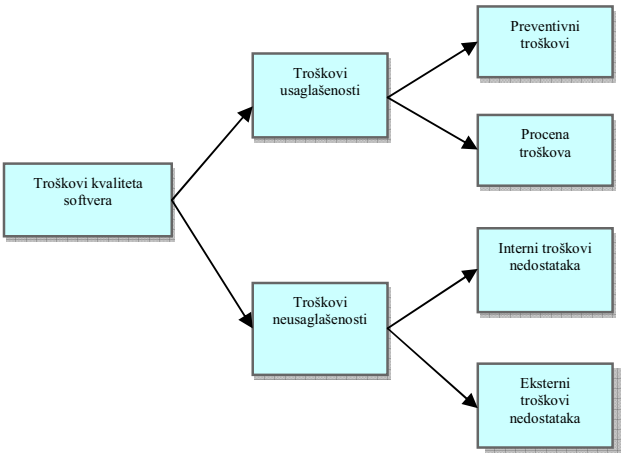


Mnogi modeli se bave problemom smanjenja troškova a da su konkurentni na tržištu i sa što boljim kvalitetom. Najveći uzroci nekvaliteta su greške koje nastaju u fazama zivotnog ciklusa proizvoda. Mi ćemo predstaviti i dati formulu troškova u fazi održavanja za tzv. Vodopadni model razvoja softvera:

- POD = Faza nastajanja Defekta (u fazi razvoja ili održavanja)
- PD = Prošli (prebegli) Defekti (iz bivše faze ili aktivnosti osiguranja kvaliteta)
- % FE = % Efikasnost filtriranja (u ovoj fazi efikasnost uklanjanja treba da je 100% jer samo na taj način imamo kvalitetan softver)
- RD = Uklonjeni Defekti (broj)
- CDR = Troškovi Uklanjanja jedne greške (u fazi održavanja cena uklanjanja jedne greške je 110 puta veća nego da je ukljonjena u fazi kad je greška nastala)
- TRC = Totali troškovi uklanjanja: TRC = RD × CDR.

U sledećem primeru, uz predpostavku da nije napravljena nijedna greška u tekućoj fazi PD=10, %FE=100%, RD=svi pridošli, ukupni troškovi uklanjanja iznose:
TRC= RD × CDR =10*110=1100 jedinica cene (€, Eur)
ili
TRC= RD × CDR × %FE=(10*110)*100%=1100 jed. cene.
Ovi troškovi održavanja se mogu izbeći ako se sprovedu odgovarajuće mere iz modela TQM u predhodnim fazama gde uklanjanje jedne greške mnogo manje košta, kao na primer u fazi specifikacije zahteva, gde uklanjanje jedne greške košta 1 jedinica cene, u fazi Dizajna 2.5, u fazi Jediničnog testiranja 6.5, Integracioni test 16, Sistem test 40 jedinica cene [3].
Efikasnost detekcije defekta (procentualno) ili DDE pokazuje kolika je uspešnost testiranja i detekcija greške, sto poboljšava kvalitet softvera [4]. Ova metrika se proračunava po formuli:
DDE=TDFT/(TDFT+TDFC)*100
gde je: TDFT -ukupno defekata nađeno testiranjem (rezultat test tima), a TDFC - ukupno defektata nađeno kod korisnika (mereno do neke tačke od puštanja softvera- 6 meseci)
Efikasnost uklanjanja defekta ili DRE pokazuje kolika je uspešnost aktivnosti Testiranja tj. uklanjanja detektovane greške, a izračunava se po formuli:
DRE=TDCT/TDFT*100
gde je:TDCT - ukupno zatvorenih defekata tokom testiranja, a TDFT- ukupno defekata nađenih tokom testiranja.
Model troškova kvaliteta, obezbeđuje metodologiju za klasifikaciju troškova vezanih za osiguranje kvaliteta proizvoda sa ekonomskog aspekta. Razvijen je tako da odgovara zahtevanom kvalitetu u proizvodnoj organizaciji te je model od velikog značaja u praksi [5].

Ovaj model svrstava troškove vezane za kvalitet proizvoda u dve opšte klase:
– **Troškovi kontrole:** obuhvataju troškove koji se troše za sprečavanje i otkrivanje grešaka softvera u cilju da ih svede na prihvatljiv nivo.
– **Neuspeli troškovi kontrole**: obuhvataju troškove koji nisu uspeli da otkriju ili spreče greške koje su se dogodile u softveru. Ovaj model troškove dalje razlaže na podklase.
Troškovi kontrole se mogu podeliti na interne i eksterne troškove nedostataka:
- Interni troškovi nedostataka uključuju troškove greške koje su otkrivene u dizajnu, testiranju softvera i testiranju prihvatljivosti (sprovedena od strane korisnika), a završen pre nego što je instaliran na opremi kod klijenta[5].
- Eksterni troškovi nedostataka uključuju sve troškove vezane za ispravljanje propusta otkrivenih od strane kupaca ili tima za održavanje nakon sto je softverski sistem instaliran.
Model troškova kvaliteta softvera je prikazan na Sl. 2.



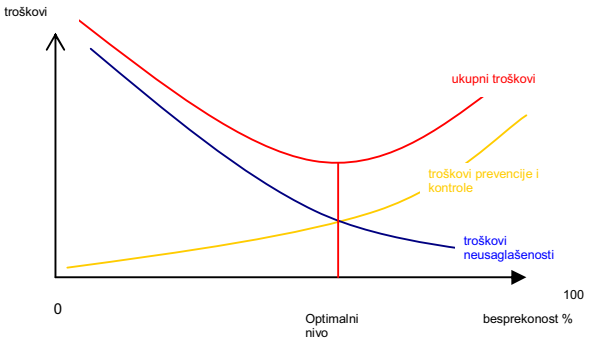
Sl. 2. Model troškova kvaliteta softvera

Sto je manje grešaka to su manji troškovi, a samim tim i kvalitet softvera. Troškovi kvaliteta softvera zavise direkno i od toga kada je primećena greška tj. u kojoj fazi životnog ciklusa proizvoda je uočena greška, jer se iz prakse zna da ako je greška uočena u fazi prodaje, a nastala na početku razvoja softvera, ona košta 1000 puta više. Najjednostavnije prestavljen trošak kvaliteta (Tk) je jednak vrednosti racionalno (planski) utvrđenih troškova, odnosno prihvatljivih troškova procesa (Tp) i vrednosti troškova koji su nastali zbog neusaglašenosti procesa sa zahtevima kvaliteta, odnosno neprihvatljivih troškova (Tn), tačnije: Tk= Tp + Tn. Ukupni troškovi direktno zavise od troškova prevencije i kontrole i troškova neusaglašenosti pa postoji tačka optimalnog upravljanja kvalitetom softvera, to se najbolje vidi na slici 3.

III. EKONOMIKA UPRAVLJANJA KVALITETOM SOFTVERA – HIPOTETIČKI SLUČAJ

A. Tehnike za analizu povraćaja investicije uloženih u proces testiranja softvera (ROTI)

ROTI model [3][4] upoređuje troškove razvoja konvencionalnog projekta, sa troškovima razvoja projekta koji koristi tzv. Test Driven Development model (TDD).



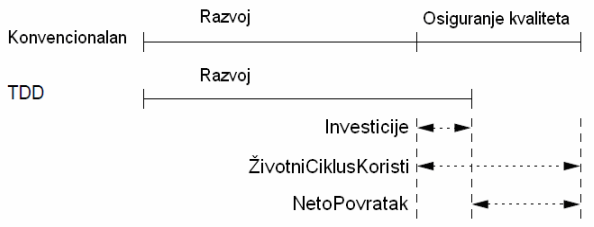
Sl. 3 Opmalni nivo ukupnih troškova kvaliteta

Troškovi investicija su početni korak neophodan za kompletiranje TDD projekta koji bi bio uporediv sa konvencionalnim. U životnom ciklusu proizvoda korist se meri razlikom u kvalitetu izraženim brojem grešaka otkrivenim i ispravljenim od strane TDD tima, dok takav slučaj nije u konvencionalnim (tradicionalnim) projektima. Ovaj nedostatak (razlika) se pretvara u novčanu vrednost koristeći dodatni napor programera u cilju pronalaženja i ispravljanja ovih nedostataka u konvencionalnom modelu. Koncept koristi (dobiti) u toku životnog cilkusa i neophodnih investicionih troškova su predstavljeni na Sl. 4. Horizontalnim linijama je predstavljen konvencionalni tok realizacije projekat sa fazom dodatnog obezbeđenja kvaliteta. Niže horizontalne linije predstavljaju TDD tok realizacije projekta. Kao rezultat, TDD model beleži povratak investicija (ROI) [3][4] TDD tima u procesu testiranja poboljšanja softvera (SPI).

B. Finansijski izražen ROI

Iz perspektive programera, postoje dva tipa koristi koje se mogu ostvariti implementacijom dobre prakse izrade kvalitetnog softvera i alata: novac i vreme. Finansijski ROI traga za uštedama u troškovima, vremenski ROI za uštedama u rasporedu (vremenu). Direktni finansijski ROI je izražen u vidu napora, obzirom da je ovo najveći trošak u projektu softvera. Postoji nekoliko različitih modela koji mogu biti upotrebljeni za ocenu finansijkog ROI za kvalitet softvera. Prvi je, najčešće korišćeni ROI model. Pokazaćemo da ovaj model nije adekvatan, zato što ne obračunava tačno koristi ulaganja u softverskim projektima. Ovo ne znači da je model nekoristan (na primer, računovođe sa kojima smo razgovarali preferiraju tradicionalni ROI model), samo što ga mi nećemo primeniti u našim kalkulacijama.

Metod povratka investicija (ROI) uključuje koristi, troškove, odnos koristi/troškovi, neto sadašnju vrednost i prelomnu tačku. To je predstavljeno na Sl. 5. ROI metode su, uopšteno posmatrano, sasvim lake, neophodne, snažno pojednostavljene i apsolutno neophodne u polju procesa softverskog poboljšanja (SPI). Ironija je da ROI metode nisu deo uobičajene prakse.



Sl.4 Koncept računanja odnosa koristi i troškova

U literaturi ne obiluju ROI metode za SPI. Veoma je teško pronaći literaturu vezanu za ROI. Pojavljuje se povremeno i često rezultati SPI koji umeju da budu zbunjujući.

Metrika	Definicija	Formula
Costs	Ukupan iznos potrošenog novca na novom i poboljšanom softverskom procesu	$\sum_{i=1}^{n} Cost_i$
Benefits	Ukupan iznos dobijenog novca od novog i poboljšanog softverskog procesa	$\sum_{i=1}^{n} Benefit_i$
B/CR	Odnost koristi i troškova	$\frac{Benefits}{Costs}$
ROI	Odnos prilagođenih koristi i troškova	$\frac{Benefits - Costs}{Costs} \times 100\%$
NPV	Diskontni protok novca	$\sum_{i=1}^{Years} \frac{Benefits_i}{(1+Discount\ Rate)^{Years}} - Costs_0$
BEP	Tačka kada upoznajemo dobiti ili prekoračujemo troškove	$\frac{Costs}{Old\ Costs / New\ Costs - 1}$

Sl.5 ROI metrika, prikaz jednostavnih formula povratka investicija i njihov redosled primene

Takođe tragamo za ROI na nivou projekta, posebno za povratkom investicija testitanja (ROTI) pre nego za preduzeće kao celinu. ROI na nivou preduzeća (ili za više projekata) zahteva nešto drugačiji pristup koji nećemo direktno razmatrato sada. Najčešći ROTI model, koji je često korišćen u softverskom inženjerstvu je sledeći:

$$ROTI_1 = \frac{Total \cdot CoQ \cdot Saved - Test \cdot Investment}{Test \cdot Investment}$$

Ovaj ROTI model nam pokazuje kolike su uštede u ukupnim troškovima kvaliteta (CoQ) dobijene iz projekta u odnosu na inicijale investicije. Da pogledamo nekoliko primera koji pokazuju kako ovaj model funkcioniše. Koristićemo hipotetičku studiju da ilustrujemo primenu ove tehnike troškova kvaliteta za analizu povraćaja testiranog ulaganja.

Prvi slučaj: Rezultati niskog kvalitea

Prvi slučaj pretpostavlja relativno mali sistem softverskog projekta sa 251 funkcionalnom tačkom. Broj defekata se procenjuje stepenovanjem broja funkcionalnih tačaka na 1.25, što rezultira u ukupno 1.000 nedostataka, ili 4 nedostatka (greške) po funkcionalnoj tački [4]. Efikasnost uklanjanja nedotsataka se pretpostavlja da je 75%. Pretpostavlja se da je razvojni tim ispod nivoa 1 na CMM skali u Procesu razvoja softvera (SDP), koji je nepredvidiv i slabo kontrolisan tj. Ad-hoc nivo. Kao što je prikazano u koloni ''Prvi slučaj testiranja'' u tabeli 1, nači troškovi kvaliteta su ¾ miliona dolara. Nije ni da nam ovih 750.000 utrošenih dolara nije donelo ništa profita, ali, s obzirom da se 750 grešaka našlo u upotrebi od starne korisnika, to je siguran znak da su korisnici poludeli od njih!

Drugi slučaj: Rezultati dobrog kvaliteta

Slučaj 2 podrazumeva softver iste veličine i klase kao u Prvom slučaju. Menadžment projekta je odlučio da poboljša proces testiranja softvera (STP) i da investira u osoblje 60.000€, i infrastrukturu 10.000€ kao što je prikazano u koloni ''Slučaj 2-testiranje'' u Tabeli 1. Pretpostavlja se da je razvojni tim Nivo 1 CMM skale. Efikasnost uklanjanja nedostataka se pretpostavlja da je 85%. Operacije uklanjanja nedostataka se sastoje od testova: 1) testa jedinice, 2) testiranja novih funkcija, 3) testa regresije, 4) testa integracije, 5) sistemskog testa i 6) eksternog Beta testa.

Treći slučaj: Rezultati visokog kvaliteta

I u trećem slučaju je softver iste veličine i klase kao u prvom. Pretpostavlja se da je razvojni tim na nivou većem od trećeg na CMM skali. To podrazumeva efikasniju prevenciju nedostataka kao što je: Funkcija razvoja kvaliteta (QFD) i primena Six-sigma strategije pa su potencijalni nedostaci niži. Efikasnost uklanjanja neodstataka se pretpostavlja da iznosi 95%. Operacije uklanjanja nedostataka se sastoje od 9 faza: 1) kontrola dizajna, 2) nadzor programskog koda, 3) testa jedinice, 4) testiranja novih funkcija, 5) testa regresije, 6) testa integracije, 7) testa performansi, 8) sistemskog testa, 9) Eksternog Beta testa.

TABELA 1: TROŠKOVI KVALITETA ANALIZA DVA NACINA RAČUNANJA POVRATKA INVESTICIJA

Testiranje resursa	Slučaj 1 CMM<1 Nivo	Slučaj 2 CMM 1 Nivo	Slučaj 3 CMM>3 Nivo
Osoblje	€ 0	€ 60.000	€ 60.000
Infrastruktura	€ 0	€ 10.000	€ 10.000
Alati	€ 0	€ 0	€ 12.500
Ukupna Investicija Testiranja	€ 0	€ 70.000	€ 82.500
Razvoj (Zahtevi, Dizajn, Kodiranje)			
Obavezno-popraviti nađene Bagove (greške)	250	250	350
Troškovi popravke-€10 po bagu (interne greške)	€ 2.500	€ 2.500	€ 3.500
Testiranje			
Obavezno-popraviti nađene Bagove (greške)	0	600	600
Troškovi popravke-€100 po bagu (interne greške)	€ 0	€ 60.000	€ 60.000
Podrška korisniku			
Obavezno-popraviti nađene Bagove (greške)	750	150	50
Troškovi popravke-€1000 po bagu (interne greške)	€ 750.000	€ 150.000	€ 50.000
Troškovi Kvaliteta (CoQ)			
Troškovi Usaglašenosti sa zahtevima	€ 0	€ 70.000	€ 82.500
Troškovi Ne-Usaglašenosti sa zahtevima	€ 752.500	€ 212.500	€ 113.500
Ukupno CoQ	€ 752.500	€ 282.500	€ 196.000
Return on Investment 1	#N/A	571%	575%
Return on Investment 2	#N/A	62%	74%

Pretpostavimo da smo izračunali da će greške evidentirane od strane programera koštati 100€. Ovo je jedna desetina troška koji bi nastao kada bi se greška pojavila kod našeg korisnika. Dakle, mi smo investirali 70.000€ kvartalno u Slučaju 2. Kolona ''Slučaj 2-testiranje'' nam pokazuje koliko su profitabilne naše investicije. Programeri su otkrili 600 grešaka pre puštanja, što smanjuje skoro za 80% broj grešaka pronađen od strane korisnika. Ovo će bez sumnje usrećiti korisnike. Ukupni troškovi kvaliteta su pali na oko pola miliona dolara, i takođe uživamo u lepom prihodu od 571% povraćaja na 70.000€ investicija.

U nekim slučajevima, možemo čak i bolje. Na primer, pretpostavimo da smo uložili 12.500€ u sredstva automatizacije i nadzorne aktivnosti. Pretpostavimo da nameravamo da povratimo uložena sredstva tokom sledećih 12 kvartalnih puštanja. Da li ćemo biti srećni ako nam ove investicije u automatizaciju omoguće otkrivanje za oko 67% više grešaka?

Otkrivanje 350 grešaka u fazi razvoja, i 600 u procesu testiranja će smanjiti ukupne greške pronađene od strane korisnika računato po svakom puštanju (kvartalno) na 50. Razvojem i primenom još formalnijih i rigoroznijih tehnika testiranja se uklanja čak 950 od 1000 grešaka, tj.ukupno 95% DRE. Svakako, korisnici će biti mnogo zadovoljniji ako imaju sistem koji je mnogo više testiran (više puta prošao kroz proces testiranja). Takođe, troškovi kvaliteta će pasti nešto niže od 200.000€ a povratak investicija (ROI) iznositi 575%.

IV. ZAKLJUČAK

Svaka softverska firma treba da ima osiguranje kvaliteta softvera iz prostih razloga da bi bila konkurentna na tržištu. Samo softver sa visokim nivoem kvalitata je prihvatljiv za korišćenje, zato se moraju koristiti aktivnosti koje će osigurati kavalitet softvera. Što veća ulaganja u neke faze životnog ciklusa proizvoda to ćemo smanjiti ukupne troškove. Troškovi su neophodni, ali ih treba upotrebiti u pravo vreme i sprečiti neželjene troškove koji su mnogo veći i nepredvidljivi, u nekim slučajevima dovode i do prekida proizvodnje ili korišćenja softvera.

LITERATURA

[1] S. H. Kan. *Metrics and Models in Software Quality Engineering*, Second Edition, Addison-Wesley, 2003.

[2] K Schulmeyer and J. McManus, "Total Quality Management for Software", International Thompson Computer Press, 1996.

[3] Lj. Lazić, N. Mastorakis. "Cost Effective Software Test Metrics", WSEAS TRANSACTIONS on COMPUTERS , Issue 6, Volume 7, June 2008, p599-619.

[4] Lj. Lazić, "An Economic Model of Software Quality Costs", in the WSEAS Book " RECENT ADVANCES IN COMPUTER ENGINEERING", Bucharest, Romania, November2008, p125-129

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

[6] G. Ben-Yaacov, P. Suratkar, M. Holliday & K. Bartleson, "Continuous Quality Improvements at a Silicon Valley High-Tech Software Company", Software Quality Professional (SQP), American Society for Quality, Vol 5, No 2, March 2003.

[7] R.A. Radice, "ISO 9001 Interpreted for Software Organiations", Paradoxicon Publishing, 1995.

ABSTRACT

Our paper describes successful deployment of software quality assessment program at a leading high-tech software company. In this paper we propose a model which to minimize the cost. Costs controlled by the organization and expended to prevent and detect failures so as to reduce total failures to an acceptable level, costs of failures, regarded as consequences, caused by failure to prevent and detect software errors.

COST SOFTWARE QUALITY ASSURANCE

Zenaida Šabotić, Ljubomir Lazić, Dženan Avdić