PRIMENA „CAUSE-EFFECT“ GRAFOVA I KOMBINATORNOG TESTIRANJA U TESTIRANJU SOFTVERA
APPLYING “CAUSE-EFFECT“ GRAPHS AND COMBINATORIAL TESTING TO SOFTWARE TESTING

Branislav Milojković, *Računarski Fakultet, Beograd*

Sadržaj – *Testiranje softvera je već više godina, pa i decenija, ustanovljeno kao naučna, ali i inženjerska, disciplina. Radi kontrole učinka u procesu testiranja softvera, uz obraćanje pažnje na uložen trud i resurse, neophodno je primeniti adekvatne tehnike za izbor elemenata softverskog proizvoda koji će se testirati. U ovom radu se daje opis primene kombinacije dve takve tehnike – „Cause-Effect“ analize i kombinatornog testiranja, i to na primeru konkretnog softverskog proizvoda – MS PAINT. „Cause-Effect“ analiza sistema koji se testira optimizuje broj testova prema međusobnoj sprezi „ulaza“ i „izlaza“, dok se kombinatorno testiranje fokusira na pojavu grešaka u softveru usled interakcije više faktora sistema i svojim različitim podtehnikama, takođe, daje optimalan izbor testova. U radu je ukazano na prednosti i nedostatke primene ovih tehnika, a takođe su i predloženi postupci za donošenje odluke o odabiru tehnike testiranja u opštijem slučaju.*
Abstract – *Software Testing is established as a scientific discipline, as well as one of engineering, for many years, and even decades. To control efficiency, keeping in mind the spent effort and resources for achieving this efficiency, it is necessary to apply quality techniques in chosing the elements of the product to be tested. This paper will show the application of a combination of two such techniques – “Cause-Effect” analysis and combinatorial testing; and the application of these techniques will be demonstrated on an example of a concrete software product – MS PAINT. “Cause-Effect” analysis of the System Under Test (SUT) optimizes the number of tests according to the mutual relationships of the “inputs” and “outputs” of the system, while, on the other hand, combinatorial testing focuses on the combinatorial nature of the system’s elements and uses it’s various subtechniques to make an optimal selection of test cases. Advantages and disadvantages of both techniques will be displayed, and, furthermore, a method of technique selection in a broader sense will be suggested.*

1. PROCES TESTIRANJA

Testiranje softvera, kao proces, ima tehničke, ali i ekonomske aspekte. Ekonomski aspekti su povezani sa činjenicom da su resursi i vreme na raspolaganju tima za testiranje ograničeni. Naime, iscrpno (potpuno) testiranje u mnogim slučajevima nije praktično izvodljivo zbog ekonomskih ograničenja. Organizacija koja se bavi razvojem softvera mora završiti projekat na vreme, u okvirima budžeta i zadovoljiti zahteve klijenata [1,2][1].

Tehnički aspekti testiranja se odnose na tehnike, metode, metriku i alate koji se koriste da bi se osiguralo da sistem koji se testira što manje bude izložen defektima, i što je moguće pouzdaniji, unutar uslova i ograničenja koje nameće okruženje u kojem će se sistem koristiti. Testiranje je proces, i kao takav, njime se mora upravljati. To, u najmanju ruku, znači da se na nivou organizacije mora osmisliti i dokumentovati strategija testiranja. Procedure i koraci procesa testiranja se moraju definisati. Proces mora da bude isplaniran, testeri obučeni, i proces mora imati merljive ciljeve koji se mogu pratiti [3,4]. Proces mora biti sposoban da evoluira i da uvek bude u stanju usavršavanja.

2. STUDIRANI SLUČAJ – SOFTVER KOJI SE TESTIRA

Tehnike i metodi prikazani u ovom radu će biti demonstrirani na primeru testiranja već završenog i dobro poznatog softverskog proizvoda – *MS PAINT*. Izbor je izvršen tako da nema nedoumica oko izvesnih tehničkih odluka, a da ipak sistem bude dovoljno kompleksan da se na njemu mogu pokazati prednosti, kao i nedostaci, korišćenih tehnika. Pre nego što zađemo dublje u same tehnike testiranja koje će se primeniti, moramo prvo izvršiti analizu studiranog softvera, kako bismo znali preciznije šta testiramo.

Funkcionalnosti koje se izdvajaju kao ključne za funkcionisanje *MS PAINT* softvera, pa samim tim i one na koje će se staviti akcenat u procesu testiranja, su:

- F001 – Crtanje prave linije
- F002 – Crtanje „slobodnom rukom“
- F003 – Crtanje krive linije
- F004 – Crtanje pravougaonika
- F005 – Crtanje elipse
- F006 – Crtanje poligona
- F007 – Odabir debljine linije
- F008 – Odabir boje

Treba primetiti i da su pomenute funkcionalnosti međusobno povezane, naime:

[1] Ovaj rad delimično je finansiralo Ministarstvo za Nauku i Tehnološki razvoj Republike Srbije u okviru projekta tehnološkog razvoja:"Integralni i optimizirani proces testiranja i održavanja softvera", TR 13018.

- F007 se može primeniti u sklopu F001, F002, F003, F004, F005, F006
- F008 se može primeniti u sklopu F001, F002, F003, F004, F005, F006

Eksplozija broja kombinacija funkcionalnosti koje bi se testirale u svakom pojedinačnom test slučaju, pa samim tim i broja test slučajeva, već je očigledna. Konkretno, u svakom test slučaju treba izvršiti izbor:

- Jednog od **6** oblika (F001-6)
- Jedne od **5** vrednosti za debljinu linije (F007)
- Jedne od **28** boja (F008)

Napomena: ovde smo ograničili testiranje samo na 28 boja ponuđenih na paleti. *MS PAINT* zapravo podržava znatno više boja, ali ovde smo se ograničili na ponuđen izbor radi jednostavnosti. Na kraju krajeva, logično je pretpostaviti da je mehanizam za postavku boje ispravan za koji god parametar, sve dok je drajver grafičke kartice ispravan. Da bismo izvršili iscrpno testiranje (sve kombinacije proverene) treba nam:

$6 \cdot 5 \cdot 28 = 840$ test slučajeva.

3. „CAUSE-EFFECT“ GRAFOVI

Cause-Effect (C-E) graf je, u suštini, tehnika koja se koristi za testiranje hardvera, i koja je prilagođena testiranju softvera, a onda tako i razvijana. Ovo je tehnika testiranja metodom "crne kutije", dakle, ona posmatra funkcionalno ponašanje sistema, bez potrebe da se analizira interna struktura dizajna sistema. Takođe, to je jedina tehnika testiranja dizajna crnom kutijom koja uzima u obzir kombinacije uzroka ponašanja sistema i koja je primenljiva i u fazi analize i izrade specifikacije dizajna.

Cause-Effect grafovi takođe opisuju model funkcionalne zavisnosti pojedinih komponenti softvera, te se koriste i pri dizajniranju softvera. C-E analiza se fokusira na prikazivanje veza zavisnosti između ulaza (uzroka) i izlaza (posledica) programa. Te veze se predstavljaju vizuelno pomoću C-E grafa. Graf je vizuelna predstava logičkog odnosa između ulaza i izlaza, koja se može predstaviti bulovom algebrom. C-E graf omogućava odabir raznih kombinacija ulaznih vrednosti kako bi se načinio test. Eksplozija broja test slučajeva se izbegava primenom heuristike i logičkih pravila prilikom pretrage grafa.

Kod C-E analize, na osnovu specifikacije i dostupne početne dokumentacije izrade softvera, prvo treba identifikovati:

- Uzroke
- Posledice
- Ograničenja

Uzrok je bilo koji uslov dat u zahtevima koji može da utiče na rezultat rada programa.

Posledica je reakcija programa na datu kombinaciju ulaza.

Ovde ćemo opisati generičku proceduru za generisanje testova pomoću C-E grafova kroz sledeće aktivnosti:

- Identifikovati uzroke i posledice iščitavanjem zahteva.
- Svakom uzroku i posledici dodeliti jedinstveni identifikator.
- Prikazati odnos uzroka i posledica pomoću „Cause-Effect“ grafa.
- Transformisati C-E graf u ograničenu tabelu odlučivanja.
- Generisati test slučajeve iz tabele odlučivanja.

Kao primer, biće uzeta funkcionalnost *MS PAINT* programa – crtanje linije zadatom bojom.

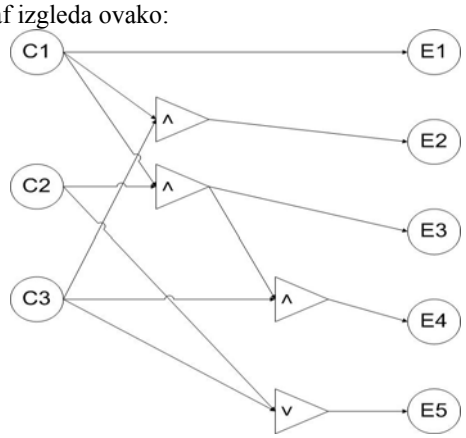
Uzroci su:

- C1: Odabir crtanja linije.
- C2: Odabir crvene boje.
- C3: Odabir debljine 2px.

Posledice su:

- E1 – Iscrtava se tanka linija crne boje.
- E2 – Iscrtava se debela linija crne boje.
- E3 – Iscrtava se tanka linija crvene boje.
- E4 – Iscrtava se debela linija crvene boje.
- E5 – Ne iscrtava se linija.

C-E graf izgleda ovako:



Slika 1. Primer C-E grafa

Naredni korak je izrada tabele odlučivanja iz samog grafa. Iz ove tabele, koja sledi, direktno će slediti test slučajevi. Ima više metoda za dobijanje tabele odlučivanja, i izbor metode izrade tabele odlučivanja igra ključnu ulogu u definisanju C-E procesa. Uzroke (C1 – izbor geometrijskog oblika, C2 – izbor boje, C3 – izbor debljine) možemo posmatrati kao bulove varijable (imaju vrednosti DA i NE, tj. 1 i 0). Trivijalan pristup bi bio napraviti po jedan test slučaj za svaku moguću kombinaciju vrednosti uzroka, što je opravdano pri testiranju *MS PAINT* na osnovu znanja da je za generisanje boje odgovoran program (drajver grafičke kartice).

	TS 1	**TS 2**	**TS 3**	**TS 4**	**TS 5**	**TS 6**	**TS 7**	**TS 8**
C1	**Da**	**Da**	**Da**	**Da**	*Ne*	*Ne*	*Ne*	*Ne*

C2	**Da**	**Da**	*Ne*	*Ne*	**Da**	**Da**	*Ne*	*Ne*
C3	**Da**	*Ne*	**Da**	*Ne*	**Da**	*Ne*	**Da**	*Ne*

Tabela 1. Trivijalno dobijena tabela odlučivanja

Po ovoj metodi dobijamo $2 \cdot 2 \cdot 2 = 2^3 = 8$ test slučajeva, ili opštije, za *n* uzroka, imaćemo 2^n test slučajeva, koji se u tehnici planiranog eksperimenta naziva još i puni faktorski plan. Takav eksponencijalan rast može da uzrokuje pojavu eksplozije broja test slučajeva, i da učini proces testiranja suviše skupim.

Prolazak unatrag kroz graf je tehnika kojom se odabir može optimizovati [1,5].

Postupak se sastoji u sledećem:

- Posmatramo graf kao skup stabala, od listova preko grana do korena, gde svako stablo počinje nekom posledicom (korenom), a završava se uzrocima (listovima).
- Ispratimo svako od tih stabala do listova (uzroka) na svaki mogući način.
- Kada dođemo do listova, one uzroke koje nismo mogli da dostignemo u tom prolasku postavimo na vrednost nedostupan (NE), a one do kojih smo došli postavimo na potvrdnu vrednost (DA).

Na našem primeru ovaj postupak daje skup od **pet** test slučajeva. Cause-Effect graf iz našeg primera bi postupkom prolaska unatrag dao narednu tabelu odlučivanja:

	TS 1	**TS 2**	**TS 3**	**TS 4**	**TS 5**
C1	**Da**	**Da**	**Da**	**Da**	*Ne*
C2	*Ne*	*Ne*	**Da**	**Da**	**Da**
C3	*Ne*	**Da**	*Ne*	**Da**	**Da**
E1	**Da**	Ne	*Ne*	*Ne*	*Ne*
E2	*Ne*	**Da**	*Ne*	*Ne*	*Ne*
E3	*Ne*	*Ne*	**Da**	*Ne*	*Ne*
E4	*Ne*	*Ne*	*Ne*	**Da**	Ne
E5	*Ne*	*Ne*	*Ne*	*Ne*	**Da**

Tabela 2. Tabela odlučivanja dobijena prolaskom unatrag kroz graf

Ako ovu tabelu prevedemo na test slučajeve, izgledaće ovako:

TS1. Korisnik crta oblik bez odabira boje i bez odabira debljine.

TS2. Korisnik crta oblik bez odabira boje i uz odabir debljine.

TS3. Korisnik crta oblik uz odabir boje i bez odabira debljine.

TS4. Korisnik crta oblik uz odabir boje i uz odabir debljine.

TS5. Korisnik ne crta oblik uz odabir boje i uz odabir debljine linije.

Ovih 5 testova takođe je opravdano kao i u prethodnom slučaju od 8 testova, zato što *MS PAINT* koristi pravilo postavljanja parametara (boja, oblik,debljina) na unapred zadatu vrednost (engl. Default). U nekim primenama je opravdano, umesto ovih minimalnih 5 testova, koji su takođe obuhvaćeni i prethodnom trivijalnom metodom od 8 testova, modifikovati sa 28 testova tako da za faktor (boja) koji ima najveći broj vrednosti (28) proširimoTS4 slučaj, a ostale faktore variramo na njihove vrednosti (oblik na 6 vrednosti, a debljinu na 5 vrednosti) ciklično.

4. KOMBINATORNO TESTIRANJE

Kada se krene u proces optimiziranja procesa testiranja *MS PAINT* softvera, prvo treba primetiti da u građenju kombinacija za izbor test slučajeva učestvuju tri različita faktora (oblik, boja i debljina). Ovo nam daje prostor za smanjenje broja test slučajeva, jer su statističke studije pokazale da se daleko veći procenat defekata javlja usled neke vrednosti jednog konkretnog faktora, izolovano, ili u interakciji dva različita faktora, a da verovatnoća javljanja defekta pri interakciji više faktora drastično opada [2,3]. Dakle, treba testirati samo sve parove ulaznih vrednosti koji učestvuju u realizaciji efekta u cilju detekcije defekata u softveru.

Prvi metod koji ćemo analizirati je **„All Pairs“ metod.** Ovo možemo postići primenom „All-Pairs“ (A-P) tehnike u okviru kombinatornog testiranja pri izradi sledeće tabele test slučajeva.

Odabir test slučajeva će ići po sledećem postupku:

1. Kolone za boju i oblik su popunjene tako da se pokriju sve kombinacije tih vrednosti. To su parovi koje nikako ne možemo da izbegnemo (izbor boje i i oblika imaju najviše različitih vrednosti).
2. Uštedu ćemo dobiti na trećoj koloni. Pravićemo kombinacije tako da na svakih 6 vrednosti iz kolone „Oblik“ bude (permutovanih) 5 vrednosti u koloni „Debljina“ i jedna nepopunjena vrednost (obeležili smo znakom ‘-’), radi lakšeg generisanja kombinacija. Nakon 6 krugova (tj. 36 testnih slučajeva) pokrićemo sve parove za ove dve kolone. Sada je bitno samo u ostatku tabele zadržati šablon da bi se uparili sve moguće debljine linije sa svakom mogućom bojom.
3. U prvu kolonu uneli smo kodne oznake boja, koje su preuzete iz zvaničnog opisa *MS PAINT* softvera.

R. Br.	**Boja**	**Oblik**	**Debljina**
1.	1 – Black	1 – Prava linija	1
2.	1 – Black	2 – Slobodna ruka	2
3.	1 – Black	3 – Kriva linija	3
4.	1 – Black	4 – Pravougaonik	4
5.	1 – Black	5 – Elipsa	5
6.	1 – Black	6 – Poligon	-
7.	2 – White	1 – Prava linija	2
8.	2 – White	2 – Slobodna ruka	1
9.	2 – White	3 – Kriva linija	4
10.	2 – White	4 – Pravougaonik	3

11.	2 – White	5 – Elipsa	-
12.	2 – White	6 – Poligon	5
...	...	...	...
163	28–Pump. Orange	1 – Prava linija	1
164	28 – Pump. Orange	2 – Slobodna ruka	2
165	28 – Pump. Orange	3 – Kriva linija	3
166	28 – Pump. Orange	4 – Pravougaonik	4
167	28 – Pump. Orange	5 – Elipsa	5
168	28 – Pump. Orange	6 – Poligon	-

Tabela 3. Test slučajeva po metodi A-P

Ukupno ima, kao što se iz postupka generisanja može videti: **28 · 6 = 168** test slučajeva.

Ortogonalni vektori (O-V) – ozbiljan nedostatak prethodno generisanog uzorka testova je taj što odabir vrednosti za faktore nije uniformno raspoređen kroz ceo skup test slučajeva. Pregledom Tabele 1, uviđa se da postoji izvesna redundantnost pri odabiru test slučajeva tj. kao da su neku faktori od većeg značaja, što u našem primeru nije slučaj. Broj test slučajeva bi se po ovom kriterijumu mogao još više smanjiti. Ortogonalni vektori (engl. Orthogonal Array – OA) obezbeđuju mehanizam za odabir testova kod kojih se problem neujednačenosti (favorizovanja) test faktora neće javiti.

Ortogonalni vektori su dvodimenzionalni vektori (matrice) brojeva koji poseduju svojstvo da izborom bilo koje dve kolone dobijamo podjednaku raspodelu za sve parove vrednosti test faktora u redovima prethodne tabele. Osnovni elementi (pojmovi) ortogonalnih vektora su:

- **Eksperimenti** – Redovi u matrici. Broj redova u matrici (tabeli), dakle, predstavljaju broj dobijenih test slučajeva, kombinacija, opita provere softvera.
- **Faktori** – Kolone u matrici. Kolone su faktori sa vrednostima koje se direktno prevode u broj promenljivih koje matrica O-V podržava.
- **Nivo variranja** – Najveći mogući broj vrednosti za neki od faktora.
- **Snaga** – Broj kolona (za O-V je dva) čije interakcije se sagledavaju tj. od 2 do broja faktora u kombinatornom testiranju.

Ortogonalni vektori (planovi) su unapred sračunati, korišćenjem gotovih, besplatnih ili komercijalnih softvera zadajući broj faktora i broj njihovih vrednosti [3]. Mi ćemo uzeti jedan takav O-V i prilagoditi ga našem problemu. Primenom O-V alata [3] se ispoljava inženjerski aspekt u softverskom testiranju.

Uzećemo gotov ortogonalni vektor OA(2^{28},28^{1},56), koji pokriva naš slučaj, a koji ima sledeće karakteristike: 56 eksperimenata, 28 faktora koji imaju po 2 vrednosti i jedan faktor koji se varira na 28 vrednosti (kod nas je to broj boja).

Originalnu tabelu tj. OA(2^{28},28^{1},56) plan treba prilagoditi za naš primer tako što ćemo faktor koji ima 28 vrednosti uzeti kako je dato, a vrednosti za druga dva faktora (oblik i debljinu) ćemo prilagoditi tako što ćemo popunjavati vrednosti za oblik, inkrementalno i ciklično (1,2,3,4,5,6,1,...), a za debljinu ciklično i inkrementalno (1,1,1,...,1,2 ... ... ,6,6,6,...,6) kako je dato u sledećoj tabeli. Veoma važna osobina kombinatornog testiranja po metodi O-V je ta da se 100% otkrivaju greške u softveru a) ako tu grešku izaziva samo jedan faktor i neka njegova vrednost, b) ako grešku izaziva interakcija 2 faktora sa svim mogućim njihovim vrednostima, i c) ako postoji određena verovatnoća detekcije grešaka u softveru a da u njenom aktiviranju učestvuje više faktora (u našem slučaju 3). To znači da je bilo dovoljno samo 56 eksperimenata, a ne 168 kao po metodi A-P tj. uzimanjem u obzir svih kombinacija parova faktora i njihovih vrednosti. Naravno, ovo smanjenje broja testova za 67% =(168-56)/168 ima za posledicu izvestan rizik da ne otkrijemo greške u softveru koje izaziva interakcija 3 i više faktora. O ovome treba voditi računa kada se Ortogonalni vektori (planovi) primenjuju u softverskim sistemima sa kritičnom misijom (npr. Vojne primene). Pošto je *MS PAINT* softver, čija primena nije kritična, primena O-V metode donosi velike uštede u njegovom testiranju. Nedostatak O-V metode, pored navedenog rizika, je i narušavanje uniformne raspodele test slučajeva (redundancija) za neke faktore i njihove vrednosti u slučajevima kada se faktori variraju na različiti broj vrednosti, što se pojavilo u poslednja dva eksperimenta za *MS PAINT* softver, gde u koloni „oblik“ - vrednosti 1 i 2 se ponavljaju više puta nego što bi bilo kada bi faktor „oblik“ varirali na 5 vrednosti, isto kao i faktor "debljina". Primenom O-V metode smo dobili dobro balansiran skup test slučajeva, koji ima 100% verovatnoću da pronađe otkaz ako ga izazivaju parovi vrednosti test faktora, uz smanjenje sa početnih 840 na 56 test slučajeva ili minimalnih 5, ili 8, odnosno 28 kombinacija test slučajeva primenom „Cause-Effect“ grafa.

Naravno, svaki od navedenih metoda izbora test slučajeva je opravdan, i treba ga koristiti u testiranju aplikacija kod kojih će to biti isplativo. Isplativost njihove primene je opravdana ako je izračunata ušteda značajna, usled izvršavanja manjeg broja eksperimenata u odnosu na trošak popravke neotkrivene greške, ako istu otkrije korisnik u operativnoj upotrebi softvera.

R. Br.	Boja	Oblik	Debljina
1.	1	1	1
2.	14	2	1
3.	10	3	1
4.	25	4	1
5.	20	5	1
6.	22	6	1
7.	27	1	1
8.	15	2	1
9.	17	3	1
10.	18	4	1
11.	16	5	1

12.	23	6	2
13.	21	1	2
14.	2	2	2
15.	8	3	2
16.	11	4	2
17.	6	5	2
18.	28	6	2
19.	19	1	2
20.	24	2	2
21.	3	3	2
22.	9	4	2
23.	12	5	3
24.	4	6	3
25.	13	1	3
26.	5	2	3
27.	6	3	3
28.	7	4	3
29.	7	5	3
30.	6	6	3
31.	5	1	3
32.	13	2	3
33.	4	3	3
34.	12	4	4
35.	9	5	4
36.	3	6	4
37.	24	1	4
38.	19	2	4
39.	28	3	4
40.	26	4	4
41.	11	5	4
42.	8	6	4
43.	2	1	4
44.	21	2	4
45.	23	3	5
46.	16	4	5
47.	18	5	5
48.	17	6	5
49.	15	1	5
50.	27	2	5
51.	22	3	5
52.	20	4	5
53.	25	5	5
54.	10	6	5
55.	14	1	5
56.	1	2	1

Tabela 4. Test slučajeva po metodi O-V

5. ZAKLJUČAK

Proces testiranja softvera može da bude veoma problematičan zbog nametnutih ograničenja u resursima i vremenu. Promišljeno planiranje strategije testiranja je ključno za kvalitetno upravljanje procesom razvoja i testiranja softvera. Moraju se uzeti u obzir kako ekonomski tako i tehnički aspekti, posebno rizici neotkrivanja grešaka.

Prednosti „Cause-Effect“ tehnike su njena podložnost automatizovanju i činjenica da se test slučajevi izvlače iz softverske specifikacije date prirodnim jezikom. Sa druge strane, ista tehnika je veoma problematična zbog relativno male snage pri smanjivanju broja test slučajeva.

Kombinatorno testiranje uspeva da iskoristi prirodne statističke pravilnosti pri odabiru test slučajeva tj. približno ravnomerno zastupljene kombinacije vrednosti faktora koji izazivaju defekt u softveru smanjenjem redundantnih test slučajeva i na taj način daje relativno mali broj eksperimenata koji uspešno pokrivaju veliki broj funkcionalnosti i otkrivaju veliki broj defekata.

Svaki od navedenih metoda izbora test slučajeva je opravdan, i treba ga koristiti u aplikacijama u čijem su testiranju nedostaci i rizici neotkrivanja grešaka u softveru primenom opisanih postupaka i metoda prihvatljivi na osnovu ušteda u troškovima izvršavanja manjeg broja eksperimenata, poredeći sa troškom u slučaju da grešku otkrije korisnik u operativnoj upotrebi softvera.

6. LITERATURA

[1] G. L. Myers, "*The Art of Software Testing*", Wiley-Interscience, New-York, 1979.

[2] Lj. Lazić, N. Mastorakis. ” OptimalSQM:Integrated and Optimized Software Quality Management”, WSEAS TRANSACTIONS on INFORMATION SCIENCE and APPLICATIONS, Issue 10, Volume 6, pp 1636-1664, ISSN: 1790-0832, October 2009.

[3] Lj. Lazić, S. Popovic, N. Mastorakis. ” A Simultaneous Application of Combinatorial Testing and Virtualization as a Method for Software Testing”, WSEAS TRANSACTIONS on INFORMATION SCIENCE and APPLICATIONS, Issue 11, Volume 6, p p 1802-1813, ISSN: 1790-0832, November 2009.

[4] Ilene Burnstein , “*Practical Software Testing*”, Springer-Verlag New York, Inc., 2003.

[5] Lee Copeland, "*A Practitioner's Guide to Software Test Design*", Artech House © 2004.