Orodja za podporo testiranja parov

Tomaž Dogša, Mirjam Bonačić
Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko, Smetanova ulica 17, 2000 Maribor
tdogsa@uni-mb.si; mirjam.bonacic@uni-mb.si

Izvleček

V prispevku je na kratko opisana strategija testiranja vseh parov. Strategija testiranja parov je sistematičen postopek izbora vhodnih podatkov, ki predpostavlja, da so v programu takšne napake, katerih navzočnost se bo pokazala pri ustrezni kombinaciji samo dveh vhodnih vrednosti. Zelo pogosto jo uporabljamo pri testiranju raznih računalniških konfiguracij (npr. različni operacijski sistemi, tiskalniki, grafične kartice) in opcij. Sam postopek je že dlje časa znan in zahteva določen napor, če testne primere tvorimo ročno s pomočjo tabel. Ker obstaja na tržišču večje število orodij, s katerimi si lahko pomagamo pri tej strategiji, smo raziskali učinkovitost in dostopnost izbranih brezplačnih orodij.

Ključne besede: strategija testiranja vseh parov, kombinatorno testiranje, ortogonalna polja, tvorjenje testnih vzorcev, računalniška orodja.

Abstract

Pairwise Testing Tools

In the paper pairwise testing strategy is briefly described. The pairwise testing strategy is a systematic test case design based on the assumption, that presence of a fault will be discovered when particular pairs of input values are used. The procedure has been well known for a long time and demands a particular effort if test cases are designed manually. As there are many commercial tools on the market that support this strategy, we analyzed and compared the efficiency and availability of selected free pairwise testing tools.

Key words: pairwise testing strategy, combinatorial testing, orthogonal arrays, test case generation, test case generation tools.

1 UVOD

Če program testiramo z vsemi mogočimi vhodnimi podatki, smo izvedli totalno testiranje in s tem odkrili vse nepravilnosti. Tako lahko tudi dokažemo popolno korektnost programa. Ker je to v večini primerov praktično neizvedljivo oz. predrago, je naloga preverjevalca, da na podlagi raznih testnih strategij tvori manjše število učinkovitih testnih primerov. Manjšanje števila testnih primerov veča nevarnost, da bodo kljub testiranju ostale še neodkrite nepravilnosti. Na ta kompromis pa vpliva tudi pričakovana kakovost programa. Če je zahtevana kakovost velika, moramo ustrezno povečati zahteve glede temeljitosti testiranja. Sistematično načrtovanje testnih primerov[1] je pomembna disciplina, ki omogoča pregled nad temeljitostjo, stroški in učinkovitostjo.

Najbolj pomembni elementi testnega primera so začetno stanje, vhodni podatki in pričakovano obnašanje. Vhodni podatek je vsak dogodek, vrednost ali stanje zunanjih entitet, ki vpliva na delovanje programa oz. naprave. Glede na število mogočih vrednosti, ki ji lahko zavzame vhodni podatek, jih lahko razdelimo v te skupine: binarni podatek (npr. spol – moški/ženska, teža – maksimalna/minimalna), majhno število diskretnih vrednosti (npr. operacijski sistem – Windows XP, Windows 7, Linux) in zelo veliko število diskretnih vrednosti (npr. 8-bitno število).

Strategija, ki jo bomo obravnavali podrobneje, je uporabna le za vhodne podatke, stanja ali vrste komponent, ki jih lahko razvrstimo v prvi dve skupini. Hkrati bomo preizkusili dve orodji, ki tvorita testne vzorce po strategiji testiranja vseh parov.

Prispevek je razdeljen na pet razdelkov. Ker strategijo testiranja vseh parov uvrščamo v večjo skupino, ki ji pravimo kombinatorno testiranje, je le-to na kratko predstavljeno v drugem razdelku. Tretji razdelek obravnava temeljne lastnosti in probleme testiranja vseh parov. Iz nabora brezplačnih orodij smo izbrali dva primerka in napravili kratko primerjalno analizo o njihovi uporabljivosti. Rezultati primerjave so v četrtem razdelku.

2 KOMBINATORNO TESTIRANJE

Kombinatorno testiranje (angl. combinatorial testing) je testiranje raznih kombinacij vhodnih podatkov. Vsaka kombinacija predstavlja en testni vzorec. Zelo pogosto ga uporabljamo pri testiranju raznih ra-

[1] Več o testiranju programske opreme je v Beizer (1990), Jorgensen (2001), Dogša (1994) ter Bath in McKay (2008).

čunalniških konfiguracij (npr. različni operacijski sistemi, tiskalniki, grafične kartice) in opcij. V nadaljevanju bomo uporabljali samo izraz vhodni podatek,[2] čeprav bodo vse ugotovitve veljale tudi za stanja in vrste komponent.

Tabela 1 prikazuje enostaven program, ki prebere tri vhodne podatke X, Y in Z. Ker lahko vsak od njih zavzame dve vrednosti, je mogočih kombinacij oz. testnih vzorcev $2 \cdot 2 \cdot 2 = 2^3 = 8$. Tipičen program ni tako preprost, saj zahteva večje število vhodnih podatkov, kar pomeni, da obstaja zelo veliko mogočih kombinacij.

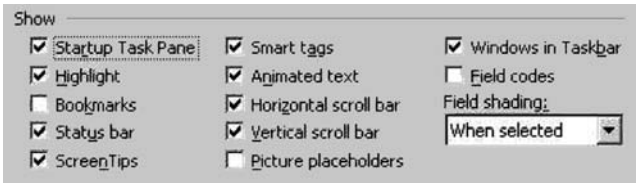
Tabela 1: **Nabor testnih vzorcev za program, ki prebere tri vhodne podatke, od katerih lahko zavzame vsak po dve različni vrednosti**

Testni vzorec	Vhod X = {1, 2}	Vhod Y = {A, B}	Vhod Z = {10, 20}
1	1	A	10
2	1	A	20
3	1	B	10
4	1	B	20
5	2	A	10
6	2	A	20
7	2	B	10
8	2	B	20

Najprej bomo izračunali število vseh mogočih kombinacij za poljubno število vhodnih podatkov. Vse vhodne podatke pregledamo in jih razvrstimo glede na število mogočih vrednosti. Naj bo m največje število mogočih vrednosti nekega vhodnega podatka in $k(i)$ število vhodnih podatkov, ki lahko zavzamejo i mogočih vrednosti. Število vseh mogočih kombinacij vrednosti vhodnih podatkov N izračunamo s pomočjo preproste enačbe:

$$N = \prod_{i=1}^{m} t^{k(i)}$$

V primeru, ki je prikazan na sliki 1, imamo 12 parametrov, ki lahko zavzamejo dve vrednosti (*'da'* ali *'ne'*) in enega, ki zavzame tri (*never, always, when selected*). Če hočemo program preveriti z vsemi mogočimi kombinacijami, moramo tvoriti $N = 2^{12} \cdot 3^1 = 12.228$ testnih primerov. Iz enačbe je razvidno, da je število kombinacij najbolj občutljivo na spremembo števila mogočih vrednosti. Če bi v prejšnjem zgledu 12 parametrov zavzelo namesto dveh tri mogoče vrednosti, bi se število kombinacij povečalo 130-krat. Če bi dodali samo en parameter, pa samo dvakrat. Ker število kombinacij skokovito narašča s številom mogočih vrednosti, je kombinatorno testiranje primerno za tiste sisteme, pri katerih lahko vsak podatek zavzame majhno število mogočih vrednosti. Za druge sisteme ima preverjevalec na voljo niz strategij, kot so mejne vrednosti, prepovedane vrednosti, ekvivalentni razredi itn.



Slika 1: **Dialog opcijskih nastavitev za MS Word (Bach in Schroeder, 2004)**

Ukaz DIR (slika 2) pozna 18 različnih vhodnih podatkov (parametrov).[3] Največ vrednosti[4] lahko zavzame stikalo /A. Ker gre za majhno število mogočih vrednosti, ki jih lahko zavzame posamezen parameter, je ta ukaz primeren kandidat za testiranje vseh parov. Število vseh mogočih kombinacij oz. testnih vzorcev je 21,626.880.

```
DIR [drive:] [path] [filename]
[/A[[:]attributes]] [/B] [/C] [/D] [/L] [/N]
[/O[[:]sortorder]] [/P] [/Q] [/R] [/S]
[/T[[:]timefield]] [/W] [/X] [/4]

/A : D,H,S,L,R,A,I,-D,-H,-S,-L,-R,-A,-I
/C : /-C
/O : N, E,G,S,D, -N, -E,-G,-S,-D
/T : C,A,W
```

Slika 2: **Vhodni podatki za ukaz DIR**

Ker je v večini primerov testiranje programa z vsemi mogočimi kombinacijami vhodnih podatkov praktično neizvedljivo oz. predrago, je naloga preverjevalca, da tvori manjše število učinkovitih testnih primerov. Ena izmed možnosti je, da program, ki ima w vhodnih podatkov, preverimo z vsemi kombinacijami n-teric, za katere velja, da je $n < w$. Npr. program,

[2] Pogosto se uporabljajo naslednji sinonimi: vhodni podatek = faktor, vrednost = opcija.

[3] Windows 7.

[4] 14 vrednosti + parametra ne navedemo = 15 mogočih vrednosti.

ki ima 8 vhodnih podatkov, preverimo z vsemi kombinacijami trojic. Število najdenih nepravilnosti z večanjem n narašča, vendar zelo počasi (Kuhn, Wallace in Gallo, 2004). Raziskave so pokazale (Black, 2007; Stobie, 2005), da če namesto vseh mogočih kombinacij uporabimo samo vse mogoče pare (n = 2), učinkovitost testnih primerov bistveno ne pade, število testnih primerov pa se izrazito zmanjša. Tako se v praksi najpogosteje odločamo za testiranje parov, saj je dovolj učinkovito, hkrati pa zanj uporabimo veliko manj testnih vzorcev kot za testiranje n-teric z višjim številom n. Pri tvorjenju testnih vzorcev si pomagamo s tabelami (ortogonalnimi polji[5]) ali pa s posebnimi računalniškimi programi.

3 STRATEGIJA TESTIRANJA VSEH PAROV

Vsaka strategija temelji na določeni predpostavki[6] (angl. bug assumption), ki se nanaša na morebitno nepravilnost. Strategija testiranja parov (angl. pairwise testing, testing all-pairs, 2-way testing) je sistematičen postopek izbora vhodnih podatkov, ki predpostavlja, da so v programu takšne napake, katerih navzočnost se bo pokazala pri ustrezni kombinaciji samo dveh vhodnih vrednosti, neodvisno od ostalih vrednosti. Ta predpostavka o nepravilnosti temelji na dveh možnostih: določena kombinacija ne sme povzročiti nepravilnega delovanja (npr. menjava vrste tiskalnika) ali pa mora vplivati na izhodno vrednost (npr. vrsta bencina na stroške prevoza). Ker ne vemo, pri katerem paru se bo to zgodilo, program preizkusimo z vsemi pari.

Za primer, ki je prikazan v tabeli 1, tvorimo pare med dvema vhodnima podatkoma: (X, Y), (X, Z) in (Y, Z). Za vhoda X in Y obstajajo natanko štirje pari: (1, A), (1, B), (2, A), (2, B). Če ustrezno izbiramo še ostale pare, potrebujemo za naš primer namesto 8 le 4 testne primere (v tabeli 1 so označeni s sivo). S tem smo zmanjšali število testnih primerov za 50 odstotkov. Kot je razvidno iz tabele 1, je vsak par vhodnih podatkov zastopan v vsaj enem od testnih vzorcev. Večje kot je število vhodnih podatkov in njihovih mogočih vrednosti, večja je razlika med številom vseh mogočih kombinacij vhodnih vrednosti ter številom vseh parov. Če imamo npr. štiri vhodne podatke, od katerih lahko zavzame vsak tri vrednosti, obstaja $3^4 = 81$ kombinacij oz. potrebujemo 81 testnih primerov. Če se odločimo za pare, lahko testiranje izvedemo samo z devetimi testnimi primeri. Za primere, pri katerih obstaja majhno število vseh mogočih kombinacij, lahko vse pare poiščemo ročno, pri večjem številu kombinacij pa je treba uporabiti posebne generatorje testnih primerov.

3.1 Učinkovitost testiranja vseh parov

Kljub relativno majhnemu številu testnih primerov raziskave kažejo, da lahko z njimi odkrijemo večino nepravilnosti na testiranem sistemu, česar ne dosežemo z ročnimi testi (Stobie, 2005; Tai in Lei, 2002). Metoda testiranja vseh parov je zaradi tega v zadnjih letih postala zelo popularna. Kljub tej popularnosti pa nekateri strokovnjaki opozarjajo na prevelika pričakovanja (Bach in Schroeder, 2004).

Učinkovitost metode testiranja parov je v svoji raziskavi potrdil Justin Hunter (Kuhn, Kracker, Lei in Hunter, 2009). V raziskavo je vključil deset različnih projektov, na katerih je primerjal uspešnost testiranja vseh parov z uspešnostjo testiranja z ročnimi metodami. Projekte je vodilo šest družb, ki so testirale aplikacije v razvoju. V vsakem projektu sta sodelovali dve skupini testerjev, ki sta istočasno testirali isto aplikacijo, vendar vsaka z drugimi testirnimi metodami. Prva je uporabljala običajne testirne strategije (npr. specifikacijsko testiranje), druga skupina pa testiranje vseh parov, pri kateri so si pomagali z generatorji testnih primerov.

Rezultati raziskave so pokazali, da je bila učinkovitost testiranja z vsemi pari večja kot pri testiranju z drugimi metodami. Vse nepravilnosti, ki so jih odkrili z ročnim izbiranjem testnih primerov, so bile najdene tudi v skupinah, ki so projekte testirale s testiranjem vseh parov. V petih projektih so te skupine odkrile še dodatne nepravilnosti, ki jih nasprotne skupine niso. Tudi primerjava časa, ki so ga posamezne skupine porabile, gre v korist strategije testiranja vseh parov.

3.2 Pogostost kombinacij

Učinkovitost testiranja lahko povečamo, če poznamo profil uporabe, ki pa le redko ustreza enakomerni porazdelitvi. Kadar so vhodni parametri določene nastavitve sistema, izkušnje kažejo, da uporabniki v večini primerov uporabljajo privzete nastavitve in jih le redko spremenijo. Če spremenijo katero izmed nastavitev, spremenijo najpogosteje le en parameter ali dva. Za določene kombinacije nastavitev obstaja

[5] Angl. orthogonal array testing.
[6] Več o tem v Beizer (1990) ali Dogša (2001).

večja verjetnost, da se bodo izvedle, kot za druge (Bryce in Colbourn, 2006). Če ta podatek poznamo, lahko določimo prioriteto izvrševanja testnih primerov. S testiranjem začnemo s kombinacijami, ki se v praksi najpogosteje uporabljajo. Kadar imamo zelo veliko število testnih primerov, lahko izpustimo tiste kombinacije, za katere velja zelo majhna verjetnost uporabe.

3.3 Pomen medsebojne povezave vhodnih podatkov

Če je n = 1, imamo najbolj preprosto strategijo, kar pomeni, da npr. že samo nastavitev enega parametra povzroči odpoved. V raziskavi (Kuhn, Wallace in Gallo, 2004) so odkrili, da je en parameter povzročil 30 do 60 odstotkov nepravilnosti. Pri n = 2 (testiranje vseh parov), pa ta učinkovitost naraste na 70 do 90 odstotkov (slika 3).

Pri odločitvi glede n je pomembno, da vemo, kateri vhodni podatki vplivajo na določeni izhod. Kadar ocenimo, da je to število majhno (npr. 2), je testiranje vseh parov dobra izbira testirne metode. Kadar pa je to število večje (8 in več), testiranje vseh parov zagotovo ni več dovolj učinkovito (Bach in Schroeder, 2004). V tem primeru je treba povečati n in si pri načrtovanju testnih primerov pomagati z ortogonalnimi polji.

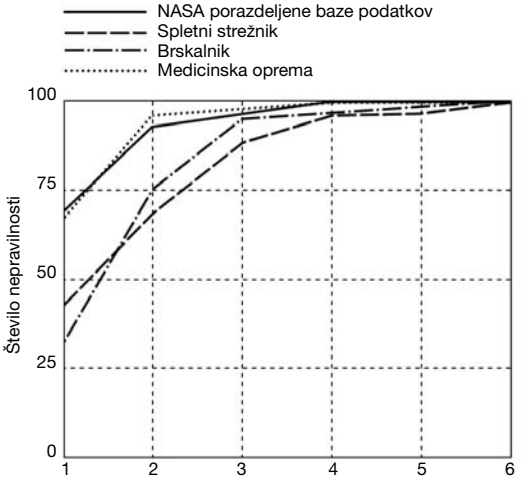
3.4 Problem oraklja

Načrtovanje testnih podatkov (vhodne vrednosti) je mogoče avtomatizirati pri katerem koli kombinacijskem testiranju, kar pa ne velja za pričakovano obnašanje. To je treba določiti s pomočjo oraklja drugače (Kuhn, Lei in Kacker, 2008). S pomočjo tabel ali orodij lahko hitro določimo testne primere, dosti več napora pa moramo vložiti v določitev pričakovanega obnašanja. Pogosto se to določevanje ne izvaja, ker preverjevalci predpostavljajo, da bo nepravilnost tako očitna, da jo bodo zlahka opazili.

4 GENERATORJI PAROV

V generatorjih, ki na podlagi strategije vseh parov tvorijo testne vzorce, so implementirani različni algoritmi. Tabela 2 prikazuje učinkovitost najpomembnejših orodij, ki je bila izmerjena na istih primerih. Takoj je opazno izrazito znižanje števila potrebnih testnih primerov, npr. s 1020 na 180. V povprečju je dalo najboljše rezultate orodje TestCover, ki ga prodaja podjetje Testcover.com.

Med prostodostopnimi orodji so npr. Jenny (podjetje Jenkins), Allpairs (podjetje Satisfice), Allpairs (podjetje McDowell), Allpairs (podjetje MetaCommunications), PICT (podjetje Microsoft) in Pairwise TestCase Generator (podjetje TestersDesk).



Slika 3: **Število nepravilnosti v odvisnosti od števila povezav med vhodnimi podatki (Kuhn, Wallace in Gallo, 2004)**

Tabela 2: **Primerjava učinkovitosti orodij za generiranje vseh parov (Czerwonka, 2009) za šest zgledov (prva kolona). V kolonah 3 do 11 je število testnih vzorcev, ki jih je tvorilo določeno orodje.**

Zgled – število vhodnih podatkov in njihovih vrednosti	Število vseh mogočih kombinacij	AETG	IPO	Tconfig	CTS	Jenny	TestCover	DDA	AllPairs [McDowell]	PICT	EXACT	IPO-s
3^4	81	9	9	9	9	11	9	?	9	9	9	9
3^{13}	$1{,}6{\cdot}10^6$	15	17	15	15	18	15	18	17	18	15	17
$4^{15}\ 3^{17}\ 2^{29}$	$7{,}4{\cdot}10^{25}$	41	34	40	39	38	29	35	34	37	?	32
$4^1\ 3^{39}\ 2^{35}$	$5{,}5{\cdot}10^{29}$	28	26	30	29	28	21	27	26	27	21	23
2^{100}	$1{,}2{\cdot}10^{30}$	10	15	14	10	16	10	15	14	15	10	10
10^{20}	10^{20}	180	212	231	210	193	181	201	197	210	?	220

V naslednjih razdelkih bomo natančneje primerjali dve prostodostopni orodji: Allpairs (McDowell) (»All-pairs Testing« n. d.) in Pairwise TestCase Generator (TestersDesk.com, n. d.), katerih učinkovitost smo preverili na dveh primerih. Za prvi primer smo vzeli ukaz DIR (slika 2), ki ima 18 vhodnih podatkov. V drugem primeru smo testirali sistem, ki vsebuje dvajset vhodnih podatkov, od katerih lahko vsak zasede 10 mogočih vrednosti.

Tabela 3: **Tabela preslikav za primer ukaza DIR**

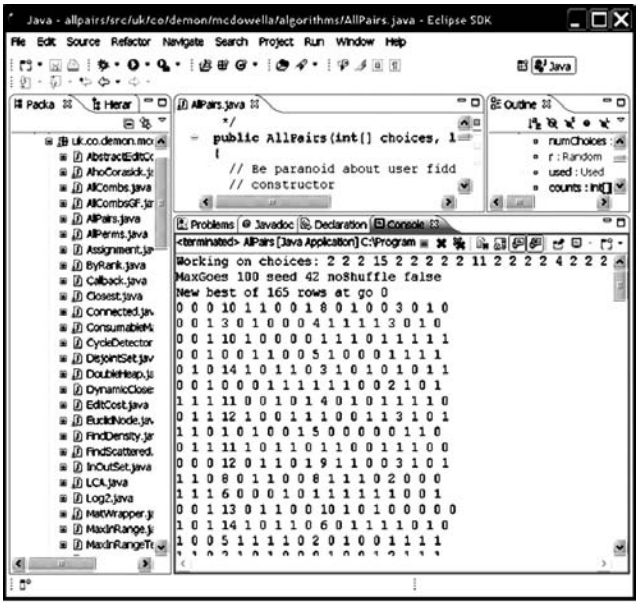
Parameter / Preslikava		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Indeks	**Ime**															
1.	[drive:]	Stikalo ni uporabljeno.	Stikalo je uporabljeno.													
2.	[path]	Stikalo ni uporabljeno.	Stikalo je uporabljeno.													
3.	[filename]	Stikalo ni uporabljeno.	Stikalo je uporabljeno.													
4.	[/A[[:]attributes]]	Stikalo ni uporabljeno.	D	H	S	L	R	A	I	-D	-H	-S	-L	-R	-A	-I
5.	[/B]	Stikalo ni uporabljeno.	Stikalo je uporabljeno.													
6.	[/C]	Stikalo ni uporabljeno.	/-C													
...	...	Stikalo ni uporabljeno.	Stikalo je uporabljeno.													
10.	[/O[[:]sortorder]]	Stikalo ni uporabljeno.	N	E	G	S	D	-N	-E	-G	-S	-D				
...	...	Stikalo ni uporabljeno.	Stikalo je uporabljeno.													
15.	[/T[[:]timefield]]	Stikalo ni uporabljeno.	C	A	W											
...	...	Stikalo ni uporabljeno.	Stikalo je uporabljeno.													
18.	[/4]	Stikalo ni uporabljeno.	Stikalo je uporabljeno.													

4.1 AllPairs (MC Dowell)

AllPairs je orodje, ki tvori kombinacije vseh parov. Orodje je izdelano v programskem jeziku java in ne vsebuje grafičnega vmesnika. Zagon orodja AllPairs smo izvedli v orodju Eclipse.

Vhodni podatek aplikacije Allpairs je niz pozitivnih števil, ki predstavljajo preprost opis vhodnih podatkov. Ker ne ponuja možnosti vnosa dejanskih vrednosti vhodnih podatkov, moramo izdelati tabelo preslikav, pri čemer je vsaki vrednosti vhodnega podatka prirejeno celo pozitivno število (glej tabelo 3).

Poglejmo prvi primer, ki ga prikazuje slika 2 in ima 18 vhodnih podatkov. S pomočjo tabele 3 tvorimo niz celih števil, s katerimi zaženemo AllPairs: 2 2 2 15 2 2 2 2 2 11 2 2 2 2 4 2 2 2. Ta niz je v bistvu opis vhodne domene. Prvi trije vhodni podatki lahko zavzamejo vrednosti 0 ali 1, četrti 0, 1, 2 … 14, peti zopet 0 ali 1 itn. Program vrne 165 kombinacij oz. testnih vzorcev, ki pokrivajo vse pare vhodnih podatkov testiranega sistema (slika 5).



Slika 5: **Rezultat tvorjenja vseh parov s pomočjo orodja AllPairs**

Rezultat moramo nato prevesti s pomočjo tabele 3 v konkretne vrednosti vhodnih podatkov. Ta pomanjkljivost je moteča, predvsem kadar je število vhodnih podatkov in njihovih vrednosti veliko.

Z orodjem AllPairs smo tvorili tudi testne vzorce za drugi primer. Nastalo je 198 testnih vzorcev, kar je veliko manj od števila vseh mogočih kombinacij, ki je 10^{20}. Primer nam pokaže, da metoda testiranja vseh parov učinkovito zmanjša število testnih vzorcev.

4.2 Pairwise TestCase Generator

Drugo orodje za testiranje vseh parov, ki smo ga preizkusili, je Pairwise TestCase Generator (PTCG). Do orodja dostopamo prek grafičnega spletnega vmesnika (http://www.testersdesk.com/). Za brezplačno uporabo tega orodja je potrebna registracija v sistem. Kot registrirani uporabnik lahko izbiramo med šestimi različnimi programi za generiranje testnih vzorcev, ki uporabljajo različne metode: Pairwise TestCase Generator za testiranje po metodi vseh parov, T-way TestCase Generator, N-wat/Random TestCase Generator, Subset TestCase Generator, Boundary TestCase Generator (BVA++) ter TestCase Permutations Generator.

PTCG omogoča vnos vseh mogočih vhodnih podatkov ter njihovih vrednosti, ki jih vpišemo v navadno besedilno datoteko, tako da ne potrebujemo tabele preslikav kot pri uporabi orodja AllPairs. Zapis vhodnih podatkov mora biti oblikovan tako, kot prikazuje slika 6. Prednost tega orodja v primerjavi z orodjem AllPairs je v tem, da so lahko vhodni podatki s pripadajočimi vrednostmi zapisani v besedilni obliki. To omogoča lažjo povezljivost z dejanskimi vhodnimi podatki testiranega sistema.

Po vnosu vhodnih podatkov in njihovih pripadajočih vrednosti v PTCG lahko izbiramo med izpisom rezultata na zaslon ter izpisom v formatu CSV.[7] Slika 6 prikazuje prvi način izpisa. Rezultat generiranja testnih vzorcev za primer ukaza DIR je 165.

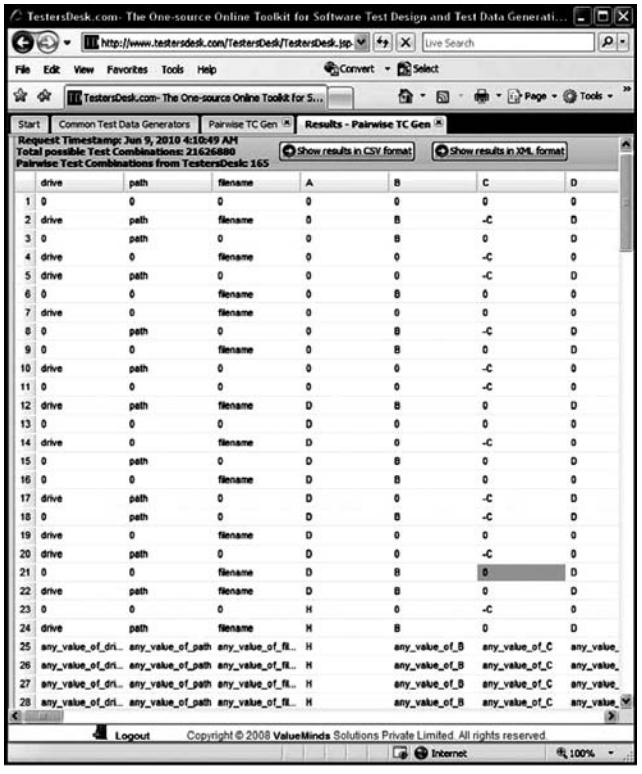
```
drive:0,drive
path:0,path
filename:0,filename
A:0,D,H,S,L,R,A,I,-D,-H,-S,-L,-R,-A,-I
B: 0,B
C:0,-C
D:0,D
L:0,L
N:0,N
O:0,N,E,G,S,D,-N,-E,-G,-S,-D
P:0,P
Q:0,Q
R:0,R
S:0,S
T:0,C,A,W
W:0,W
X:0,X
4:0,4
```

Slika 6: **Oblika zapisa vhodnih podatkov ukaza DIR, pripravljenih za uporabo v programu PTCG**

Orodje PTCG je za omenjeni primer generiralo isto število testnih vzorcev kot orodje AllPairs. Ob podrobnem pregledu rezultatov obeh orodij pa ugotovimo, da se testni vzorci, ki sta jih generirali orodji, razlikujejo med seboj.

Iz slike 7 je razvidno, da je orodje PTCG v 25. do 28. testnem vzorcu za nekatere vhodne vrednosti zapisalo flany_value_of_X« (pri čemer je X ime vhodnega podatka). To pomeni, da lahko uporabnik sam izbere vrednost vhodnega podatka X, saj bo v vsakem primeru zagotovljena popolna pokritost vseh parov. Aplikacija izračuna tudi število vseh mogočih kombinacij vhodnih podatkov.

[7] Angl. comma-separated values – preprost besedilni format, v katerem so podatki ločeni z decimalno vejico.



Slika 7: **Testni vzorci za primer vhodnih podatkov s slike 6, ki jih je tvorilo orodje Pairwise TestCase Generator**

Z orodjem PTCG smo generirali tudi testne vzorce za drugi primer z dvajsetimi vhodnimi podatki. Kot rezultat smo dobili 215 kombinacij testnih vzorcev, kar je 17 več kot smo jih dobili s pomočjo orodja AllPairs. V tem primeru lahko sklepamo, da je algoritem orodja AllPairs učinkovitejši, vendar pa sta obe orodji nalogo opravili zelo uspešno, če rezultat primerjamo s številom vseh mogočih kombinacij testnega primera, ki je 10^{20}.

	AllPairs (McDowell)	**PTCG**
Grafični uporabniški vmesnik	Ne	Da
Vrsta dostopa	Namizna aplikacija	Spletna aplikacija
Možnost vnosa dejanskih vrednosti vhodnih podatkov	Ne (potrebna je izdelava tabele preslikav)	Da
Možnost vnosa vhodnih podatkov iz besedilne datoteke	Ne	Da
Možnost izvoza testnih vzorcev v formatu CSV	Ne	Da
Izračun vseh mogočih kombinacij vhodnih podatkov	Ne	Da
Število generiranih testnih vzorcev za primer ukaza DIR	165	165
Število generiranih testnih vzorcev za primer dvajsetih vhodnih podatkov z desetimi vrednostmi	198	215

Slika 8: **Primerjava orodij AllPairs in PTCG**

Primerjava obeh orodij (slika 8) pokaže, da je orodje AllPairs bolj učinkovito, kadar tvorimo testne vzorce za večje število vhodnih podatkov. Na drugi strani je PTCG uporabniku bolj prijazno orodje, saj ima preprost grafični uporabniški vmesnik ter možnost branja vhodnih podatkov iz besedilne datoteke, v katero lahko zapišemo dejanske vrednosti vhodnih podatkov, torej ne potrebujemo tabele preslikav. PTCG je dostopna prek spleta, medtem ko je AllPairs namizna aplikacija, za katero potrebujemo okolje java.

5 SKLEP

Učinkovito testiranje zahteva, da ocenimo kakovost sistema v čim krajšem času ter s čim nižjimi stroški. Ker imajo različni sistemi svoje specifične značilnosti, ne obstaja univerzalna metoda testiranja, ki bi bila najboljša rešitev za vse sisteme. Metoda testiranja vseh parov je kljub sistematičnosti preprosta in v mnogih primerih zelo učinkovita rešitev za testiranje sistemov z več diskretnimi vhodnimi podatki, ki so medsebojno le rahlo povezani. Da bi znali metodo testiranja vseh parov uporabiti pravilno in učinkovito, moramo poznati predpostavke o napakah oz. nepravilnostih, katerih navzočnost lahko odkrijemo s to strategijo. Na tržišču obstaja niz brezplačnih orodij, ki podpirajo načrtovanje parov oziroma n-teric. Njihova uporabljivost je na takšni ravni, da predstavljajo učinkovit pripomoček za načrtovanje testnih vzorcev.

6 LITERATURA

[1] *All-Pairs Testing.* http://www.mcdowella.demon.co.uk/allPairs.html, dostopno 26. 8. 2010.

[2] Bach, J., Schroeder, P. (2004, oktober). *Pairwise testing: A best practice that isn't.* 22nd Pacific Northwest Software Quality Conference, Portland, 175–193.

[3] Bath, G., McKay, J. (2008). *The Software Test Engineer's Handbook: A Study Guide for the ISTQB Test Analyst and Technical Analyst Advanced Level Certificates,* Rocky Nook Inc.

[4] Beizer, B. (1990). *Software Testing Techniques,* Van Nostrand Reinhold.

[5] Black, R. (2007). *Pragmatic Software Testing: Becoming an Effective and Efficient Test Professional.* New York: Wiley.

[6] Bryce, R. C., Colbourn, C. J. (2006, oktober). *Prioritized interaction testing for pair-wise coverage with seeding and constraints.* Information and Software Technology, 48(10), 960–970.

[7] Cohen, D. M., Dalal, S. R., Fredman, M. L., Patton, G. C. (1997). *The AETG system: An approach to testing based on combinatorial design.* IEEE Transactions On Software Engineering, 23(7).

[8] Colbourn, C. J., Cohen, M. B., Turban, R. C. (februar, 2004). *A Deterministic Density Algorithm for Pairwise Interaction Coverage*. Proc. IASTED International Conf. Software Engineering, Innsbruck Austria, 245–252.

[9] Czerwonka, J. (2009). *Pairwise Testing: Combinatorial Test Case Generation*. http://www.pairwise.org/, dostopno 26. 8. 2010.

[10] Czerwonka, J. (2006, oktober). *Pairwise testing in real world*. In PacificNorthwest Software Quality Conference, 419–430.

[11] Dogša, T. (2001). *Splošen koncept načrtovanja testnih primerov*, Uporabna informatika, štev. 2, letnik IX, 2001, 75–79.

[12] Dogša, T. (1994). *Verifikacija in validacija programske opreme,* Tehniška fakulteta Maribor.

[13] Jorgensen, Paul C. (2001). *Software testing – A Craftsman's Approach,* CRC Press LLC.

[14] Kuhn, R., Kacker, R., Lei, Y., Hunter, J. (2009, avgust). *Combinatorial Software Testing.* Computer*,* 42(8), 94–96.

[15] Kuhn, R., Lei, Y., Kacker, R. (2008, maj/junij). *Practical Combinatorial Testing: Beyond Pairwise.* IT Professional Magazine. Washington, 10(3), 19.

[16] Kuhn, D. R., Wallace, D. R., Gallo, A. J., Jr. (2004, julij). *Software Fault Interactions and Implications for Software Testing*. IEEE Transactions on Software Engineering, 30(6).

[17] *Pairwise TestCase Generator from TestersDesk.com.* http://www.testersdesk.com/pairwse_testersdesk.html, dostopno 26. 8. 2010.

[18] Stobie, K. (2005, februar). *Too darned BIG to test*. New York: ACM. Queue, 3(1), 30–37.

[19] Tai, K. C., Lei, Y. (2002, januar). *A test generation strategy for pairwise testing.* IEEE Transactions on Software Engineering. New York, 28(1), 109.

▪

Tomaž Dogša je izredni profesor na Fakulteti za elektrotehniko, računalništvo in informatiko Univerze v Mariboru, kjer predava na dodiplomski in podiplomski stopnji in vodi Center za verifikacijo in validacijo sistemov. Na raziskovalnem področju se ukvarja predvsem s tehnologijo V&V oz. testirnimi orodji.

▪

Mirjam Bonačić je asistentka za področje informatike na Fakulteti za elektrotehniko, računalništvo in informatiko Univerze v Mariboru, kjer sodeluje pri predmetih s področja podatkovnih baz. Hkrati je študentka magistrskega študija na omenjeni fakulteti. Aktivno se udeležuje konferenc in je avtorica in soavtorica številnih člankov v domačih in tujih revijah.