ISSN: 2089-1121

PENINGKATAN EFEKTIVITAS METODA UJI COBA PERANGKAT LUNAK DENGAN MENGGUNAKAN FSCS-ADAPTIVE RANDOM TESTING DAN RESTRICTED RANDOM TESTING

[1]Daniel Harsono Christanto, [2]Edwin Pramana
Teknik Informatika
Sekolah Tinggi Teknik Surabaya
[1]danielchristanto@gmail.com, [2]epramana@stts.edu

ABSTRAK

Adaptive Random Testing merupakan sebuah pengembangan dari Random Testing (RT). Baik ART maupun RT sama-sama mengusahakan menghasilkan kasus uji coba yang tersebar merata. Namun ketika dilakukan observasi lebih lanjut, ART kurang efektif bila diaplikasikan pada parameter input yang semakin banyak. Pada penelitian tesis ini, kami menganalisa Fixed-sized Candidate Set-Adaptive Random Testing (FSCS-ART) pada kasus parameter input yang semakin besar dan mempelajari pengembangan yang dapat dilakukan untuk mengatasi permasalahan tersebut. Kami mengusulkan untuk menambah kandidat kasus uji coba melalui Restricted Random Testing (RRT). Diharapkan penelitian kami akan meningkatkan improvisasi algoritma ART yang lain pada permasalahan parameter input yang semakin besar.

Kata kunci: Metoda Uji Coba Program, Adaptive Random Testing, ART, FSCS-ART, Restricted Random Testing, RRT.

ABSTRACT

Adaptive random testing (ART), an enhancement of random testing (RT), aims to both randomly select and evenly spread test cases. Recently, it has been observed that effectiveness of some ART algorithms may deteriorate as the number of program input parameters (dimensionality) increases. In this thesis, we analyze Fixed-sized candidate set (FSCS-ART) in the high dimensional input domain setting, and study how FSCS-ART can be further enhanced to address these problems. We propose to add more test cases candidate through executing Restricted Random Testing. Other ART algorithms may face similar problems as FSCS-ART. Our study also brings insight into the improvement of other ART algorithms in high dimensional space.

Keywords: Software Testing Method, Adaptive Random Testing, ART, FSCS-ART, Restricted Random Testing, RRT.

1. PENDAHULUAN

Dalam bidang software development, kesalahan ataupun kegagalan harus diminimalkan. Sebagian besar usaha dikeluarkan pada saat testing. Segala cara diupayakan agar nilai kesalahan atau kegagalan bisa sama dengan 0. Jika terdapat kesalahan atau kegagalan berapapun jumlahnya, hal itu akan memakan biaya lagi yang

ISSN: 2089-1121

tentunya akan membebani biaya software development itu sendiri. Perbaikan yang sangat lama dan memakan biaya yang besar dinilai kurang layak dikerjakan dan sebaiknya dihindari.

Pemilihan test case merupakan hal yang kritikal dalam bidang software testing. Tidak semua test case harus dilakukan sehingga menghemat waktu dan biaya pengujian. Telah banyak sekali metoda uji coba dikembangkan untuk membantu memilih test case yang benar dan memaksimalkan pengidentifikasian kesalahan atau kegagalan. Salah satu metoda dalam uji coba perangkat lunak yang hendak dibahas dalam tesis ini adalah Adaptive Random Testing (ART). ART merupakan pengembangan dari metode Random Testing (RT). Kelemahan ART terletak pada permasalahan dimensi input yang semakin banyak. Fixed-sized Candidate Set – Adaptive Random Testing (FSCS-ART) hadir sebagai wujud pengembangan dari ART. Namun jika semakin banyak lagi dimensi input, maka perlu adanya perbaikan di FSCS-ART. Penulis mengusulkan FSCS-ART digabung dengan Restricted Random Testing (RRT) untuk menambah perbendaharaan kandidat kasus uji coba.

2. FSCS-ART dan FSCS-ART RRT

Fixed-sized Candidate Set (FSCS-ART) melakukan penghitungan jarak maksimum dengan kasus uji coba yang telah dijalankan. Dengan demikian kasus uji coba yang memiliki kemiripan yang paling rendah yang akan diuji berikutnya.

Algoritma 1 : FSCS-ART

1: Set n = 0 and E = { }.
2: Randomly select a test case, t, from the input domain (according to uniform distribution).
3: Increment n by 1.
4: If t reveals a failure, go to Step 9; otherwise, store t in E
5: Randomly generate k inputs to construct C = {c1, c2,..., ck} (according to uniform distribution).
6: For each element ci in C, calculate the distance di between ci and its nearest neighbour in E.
7: In C, find which element, cj, has the longest distance to its nearest neighbour in E.
8: Let t = cj and go to Step 3.
9: Return n and t, and EXIT.

Restricted random testing (RRT) menetapkan daerah larangan (*exclusion zone*) di sekitar daerah kasus uji coba yang telah dieksekusi. Kasus uji coba yang akan diujikan harus berada di luar zona tersebut. Penetapan daerah larangan pada RRT dihitung dengan menggunakan rumus , dimana *R* adalah radius yang ditetapkan sebelumnya sebelum RRT dijalankan, | *I*| jumlah *input domain*, dan |*E*| merupakan jumlah kasus uji coba yang telah dieksekusi.

Algoritma 2 : RRT

1: Input maxTrial and *R* where both maxTrial and $R > 0$.
2: Set $n = 0$ and E = { }.
3: Randomly select a test case, t, from the input domain according to the uniform distribution.
4: Increment *n* by 1.
5: If t reveals a failure, go to Step 4; otherwise, store t in E.
6: Set noOfTrial = 0.
7: $\forall i\ e_i \square$ E, determine x_i.
8: Repeat Steps 9 – 13.
9: If noOfTrial = Maxtrial, then set $R = R - 0.1$ and re-determine all x_i.

ISSN: 2089-1121

Algoritma 2 : RRT (Lanjutan)

10: Randomly select a test case, t, from the input domain according to the uniform distribution.
11: If t is outside $\bigcup_{i=1}^{|E|} x_i$, go to Step 4.
12: Increment noOfTrial by 1.
13: Return *n* and t. EXIT.

Penelitian dilakukan pada tiga fungsi yang terdapat *software* pegawai, yaitu : fungsi perhitungan lembur (2 dimensi *input*), bonus tunjangan hari raya (1 dimensi *input*), dan *checkclock* (4 dimensi *input*). Masing-masing fungsi tersebut diujikan dengan menggunakan FSCS-ART dan FSCS-ART RRT.

FSCS-ART RRT memiliki parameter yang ditentukan dahulu sebelum RRT dijalankan. Parameter tersebut dapat dilihat pada tabel 1.

Tabel 1. Parameter RRT

Parameter	Keterangan
MaxTrial	Iterasi maksimum yang diperbolehkan untuk mencoba kasus uji coba sampai mendeteksi kasus uji coba tersebut berada di dalam zona terlarang.
InitialR	Nilai radius (R), mendefinisikan jari-jari atau radius zona terlarang.
Input Domain	Menyatakan jumlah input domain yang diambilkan dari sumbu input domain terpanjang.

Penghitungan lembur karyawan memanggil sebuah fungsi di dalam database yang bernama *GetOvertimeAmount*. Fungsi *GetOvertimeAmount* memiliki parameter seperti pada tabel 2 dan mengembalikan nilai rupiah lembur yang bertipe data *number*.

Tabel 2. Parameter GetOvertimeAmount

Nama Parameter	Tipe Data	Keterangan
Tlamalembur	Number	Lama lembur karyawan, dinyatakan dalam satuan jam.
TypeOfDay	String	Tipe hari lembur karyawan, bernilai 'BIASA' atau 'LIBUR'. 'BIASA' merupakan hari kerja non-hari libur. 'LIBUR' merupakan hari libur nasional, termasuk hari minggu.
<Return>	Number	Rupiah lembur karyawan berdasarkan tabel lembur yang berlaku.

ISSN: 2089-1121

Penghitungan bonus Tunjangan Hari Raya (THR) karyawan memanggil sebuah fungsi di dalam database yang bernama *GetBonusTHR*. Fungsi *GetBonusTHR* memiliki *parameter* seperti pada tabel 3 dan mengembalikan nilai rupiah bonus THR yang bertipe data *number*.

Tabel 3. Parameter GetCheckin

Nama Parameter	Tipe Data	Keterangan
TmasaKerja	Number	Masa kerja karyawan yang dinyatakan dalam satuan bulan.
<Return>	Number	Rupiah bonus Tunjangan Hari Raya

Checkclock karyawan memanggil sebuah fungsi di dalam database yang bernama *GetCheckin*. Fungsi *GetCheckin* memiliki *parameter* seperti pada Tabel 4 dan mengembalikan nilai keterangan *checkclock* yang bertipe data *string* atau *varchar2*.

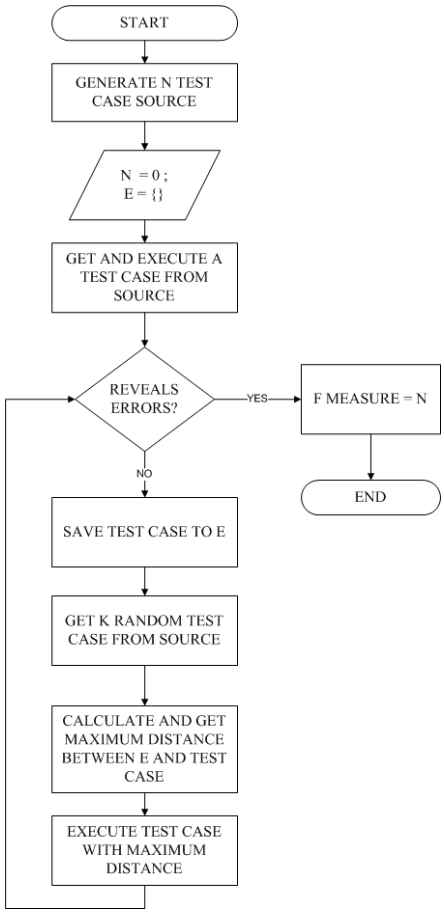
Tabel 4. Parameter GetCheckin

Nama Parameter	Tipe Data	Keterangan
KaryawanID	Varchar2(11)	Nomor identitas unik karyawan.
LastCheckClock	DateTime	*Checkclock* terakhir.
CurrentCheckClock	DateTime	*Checkclock* saat ini.
DayOfWeek	Integer	Nomor hari dalam satu minggu. Hari Minggu bernilai 1 sampai hari Sabtu bernilai 7.
<Return>	Number	Rupiah bonus Tunjangan Hari Raya.

Pada mulanya *test case* dihasilkan pertama kali untuk dipakai dalam pengujian menggunakan FSCS-ART da FSCS-ART RRT sebagai *test case source*. Tujuannya adalah memastikan bahwa *test case* yang dipakai memiliki nilai dan urutan yang sama sehingga hasil pengujian bisa akurat. Kemudian nilai N, mengidentifikasikan jumlah iterasi, diinisialisasi dengan nilai nol. Demikian juga *array* E diinisialisasi dengan nilai nol. Variabel *array* E untuk menampung *test case* yang telah tereksekusi dengan sukses tanpa *error*. Kemudian *test case* diambil dari *test case source* dan diuji apakah dengan menggunakan *test case* tersebut menimbulkan *error* ataukah tidak. Jika tidak menimbulkan *error*, maka iterasi mulai dihitung dan menjalankan algoritma FSCS-ART dan FSCS-ART RRT. Jarak antara *test case* dengan *executed test case* dihitung dan dipilih jarak maksimum untuk dipakai sebagai *next test case* dan kemudian diuji dengan menggunakan *test case* tersebut. Iterasi akan terus dijalankan dan dihitung sampai menemukan *error* pertama kali dan menghasilkan *F-Measure*. Nilai *F-Measure* ini akan dipakai sebagai tolak ukur keefektifan antara algoritma FSCS-ART dengan FSCS-ART RRT. Nilai F-Measure semakin kecil mengindikasikan keefektifan makin tinggi.

ISSN: 2089-1121

Sedangkan nilai F-Measure semakin tinggil mengindikasikan keefektifan makin rendah. Proses FSCS-ART dapat dilihat pada gambar 1.

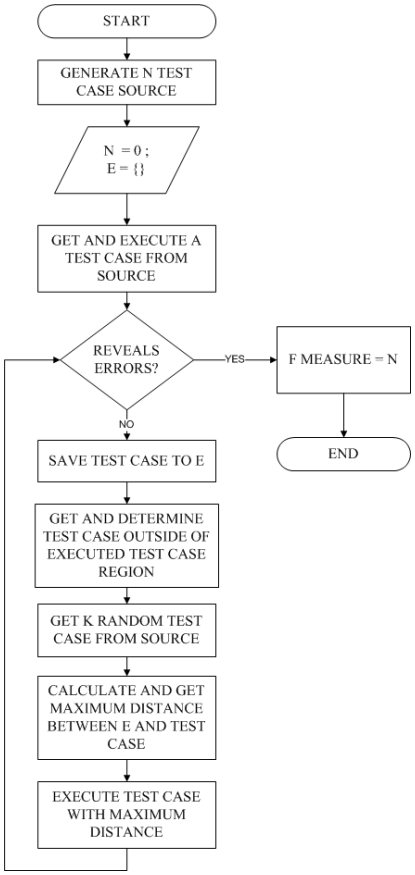


Gambar 1. Flowchart FSCS-ART

Pada FSCS-ART RRT tidak jauh berbeda dengan FSCS-ART. FSCS-ART RRT didasarkan pada FSCS-ART sehingga perbedaannya hanya penambahan *Restricted Random Testing* (RRT) saja. Ide dasar RRT adalah penetapan daerah larangan (*exclusion zone*) yang dimana *test case* tidak diizinkan untuk berada di dalam daerah larangan tersebut. Daerah larangan memiliki nilai *R*, yaitu besar radius. Semakin banyak *executed test case* pada himpunan *input domain* maka luas daerah larangan semakin sempit. Dengan demikian *test case* yang dipilih memiliki kemiripan yang kecil bila dibandingkan dengan *executed test case*. Penambahan fitur RRT diletakkan setelah menambahkan *k-random*. Setelah didapatkan satu buah *test case* dari penghitungan RRT, *test case* dari RRT tersebut juga dimasukkan ke dalam sebuah variabel penampung untuk dihitung masing-masing jaraknya terhadap *executed test case*. *Test case* yang memiliki jarak terjauh yang dipakai sebagai *next test case*. Proses FSCS-ART RRT dapat dilihat pada Gambar 2. Jarak dapat dihitung dengan persamaan matematika berikut ini :

$$dist(a,b) = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2} \quad \text{.....Persamaan 1.}$$

ISSN: 2089-1121



Gambar 2. Flowchart FSCS-ART RRT

3. HASIL PENELITIAN

Hasil penelitian yang dilakukan dengan menjalankan fungsi *GetBonusTHR* dan nilai R = 0,25 adalah sebagai berikut :

Tabel 5. Hasil Penelitian R=0,25 GetBonusTHR

No	FSCS-ART	FSCS-ART RRT	F Ratio
1	24	15	0,625
2	5	5	1
3	1	1	1
4	11	9	0,81818
5	3	1	0,33333
6	5	3	0,6
7	0	0	0
8	12	7	0,58333
9	8	4	0,5
10	7	4	0,57143
11	13	11	0,84615
12	1	1	1

ISSN: 2089-1121

Tabel 5. Hasil Penelitian R=0,25 GetBonusTHR (Lanjutan)

No	FSCS-ART	FSCS-ART RRT	F Ratio
13	7	5	0,71429
14	8	4	0,5
15	6	6	1
16	4	2	0,5
17	10	10	1
18	21	13	0,61905
19	12	9	0,75
20	7	5	0,71429
Rata-rata	8,25	5,75	0,68375

Penelitian pada fungsi *GetBonusTHR* menunjukkan bahwa FSCS-ART RRT lebih efektif 2,5 poin dibandingkan dengan FSCS-ART pada kondisi R = 0,25. Penambahan fitur RRT memberikan hasil positif dengan rata-rata F-Measure 0,68375 kepada pengujian terhadap fungsi *GetBonusTHR*.

Hasil penelitian yang dilakukan dengan menjalankan fungsi *GetBonusTHR* dan nilai R = 0,5 adalah sebagai berikut:

Tabel 6. Hasil Penelitian R=0,5 GetBonusTHR

No	FSCS-ART	FSCS-ART RRT	F Ratio
1	22	15	0,68182
2	7	5	0,71429
3	1	1	1
4	9	7	0,77778
5	1	1	1
6	5	3	0,6
7	0	0	0
8	13	7	0,53846
9	8	4	0,5
10	4	2	0,5
11	14	11	0,78571
12	1	1	1
13	7	5	0,71429
14	9	4	0,44444
15	6	4	0,66667
16	4	2	0,5
17	10	8	0,8
18	24	13	0,54167

ISSN: 2089-1121

Tabel 6. Hasil Penelitian R=0,5 GetBonusTHR (Lanjutan)

No	FSCS-ART	FSCS-ART RRT	F Ratio
19	11	5	0,45455
20	5	5	1

Penelitian pada fungsi *GetBonusTHR* menunjukkan bahwa FSCS-ART RRT lebih efektif 2,9 poin dibandingkan dengan FSCS-ART pada kondisi R = 0,5. Penambahan fitur RRT memberikan hasil positif dengan rata-rata F-Measure 0,66098 kepada pengujian terhadap fungsi *GetBonusTHR*.

Hasil penelitian yang dilakukan dengan fungsi *GetBonusTHR* dan nilai R = 0,75 adalah sebagai berikut :

Tabel 7. Hasil Penelitian R=0,75 GetBonusTHR

No	FSCS-ART	FSCS-ART RRT	F Ratio
1	19	17	0,894737
2	5	5	1
3	1	1	1
4	11	7	0,636364
5	3	1	0,333333
6	5	3	0,6
7	0	0	0
8	12	7	0,583333
9	10	6	0,6
10	6	4	0,666667
11	13	11	0,846154
12	1	1	1
13	8	7	0,875
14	9	4	0,444444
15	7	6	0,857143
16	4	2	0,5
17	10	8	0,8
18	21	11	0,52381
19	11	5	0,454545
20	5	5	1
Rata-rata	8,05	5,55	0,680776

Penelitian pada fungsi *GetBonusTHR* menunjukkan bahwa FSCS-ART RRT lebih efektif 2,5 poin dibandingkan dengan FSCS-ART pada kondisi R = 0,75.

ISSN: 2089-1121

Penambahan fitur RRT memberikan hasil positif dengan rata-rata F-Measure 0,680776 kepada pengujian terhadap fungsi *GetBonusTHR*.

Hasil penelitian yang dilakukan dengan fungsi *GetCheckin* dan nilai R = 0,25 adalah sebagai berikut :

Tabel 8. Hasil Penelitian R=0,25 GetCheckin

No	FSCS-ART	FSCS-ART RRT	F Ratio
1	11	7	0,63636
2	4	3	0,75
3	29	25	0,86207
4	10	17	1,7
5	2	2	1
6	6	3	0,5
7	26	16	0,61538
8	5	3	0,6
9	6	4	0,66667
10	13	4	0,30769
11	11	7	0,63636
12	22	13	0,59091
13	1	1	1
14	9	10	1,11111
15	17	35	2,05882
16	24	30	1,25
17	23	13	0,56522
18	11	8	0,72727
19	9	7	0,77778
20	4	2	0,5
Rata-rata	12,15	10,5	0,84278

Penelitian pada fungsi *GetCheckin* menunjukkan bahwa FSCS-ART RRT lebih efektif 1,65 poin dibandingkan dengan FSCS-ART pada kondisi R = 0,25. Penambahan fitur RRT memberikan hasil positif dengan rata-rata F-Measure 0,84278 kepada pengujian terhadap fungsi *GetCheckin*.

Hasil penelitian yang dilakukan dengan fungsi *GetCheckin* dan nilai R = 0,5 adalah sebagai berikut :

Tabel 9. Hasil Penelitian R=0,5 GetCheckin

No	FSCS-ART	FSCS-ART RRT	F Ratio
1	13	5	0,38462
2	1	3	3

ISSN: 2089-1121

Tabel 9. Hasil Penelitian R=0,5 GetCheckin (Lanjutan)

No	FSCS-ART	FSCS-ART RRT	F Ratio
3	27	29	1,07407
4	9	19	2,11111
5	2	1	0,5
6	6	3	0,5
7	27	16	0,59259
8	5	3	0,6
9	6	4	0,66667
10	10	4	0,4
11	13	13	1
12	19	10	0,52632
13	3	1	0,33333
14	5	10	2
15	15	25	1,66667
16	36	30	0,83333
17	21	13	0,61905
18	12	6	0,5
19	11	9	0,81818
20	4	2	0,5
Rata-rata	12,25	10,3	0,9313

Penelitian pada fungsi *GetCheckin* menunjukkan bahwa FSCS-ART RRT lebih efektif 1,95 poin dibandingkan dengan FSCS-ART pada kondisi R = 0,5. Penambahan fitur RRT memberikan hasil positif dengan rata-rata F-Measure 0,9313 kepada pengujian terhadap fungsi *GetCheckin*.

Hasil penelitian yang dilakukan dengan fungsi *GetCheckin* dan nilai R = 0,75 adalah sebagai berikut :

Tabel 10. Hasil Penelitian R=0,75 GetCheckin

No	FSCS-ART	FSCS-ART RRT	F Ratio
1	12	3	0,25
2	1	3	3
3	29	31	1,06897
4	6	17	2,83333
5	1	1	1
6	4	2	0,5
7	23	16	0,69565

ISSN: 2089-1121

Tabel 10. Hasil Penelitian R=0,75 GetCheckin (Lanjutan)

No	FSCS-ART	FSCS-ART RRT	F Ratio
8	5	5	1
9	6	4	0,66667
10	9	4	0,44444
11	14	5	0,35714
12	10	6	0,6
13	3	1	0,33333
14	9	12	1,33333
15	17	7	0,41176
16	24	28	1,16667
17	29	5	0,17241
18	10	2	0,2
19	11	9	0,81818
20	4	2	0,5
Rata-rata	11,35	8,15	0,86759

Penelitian pada fungsi *GetCheckin* menunjukkan bahwa FSCS-ART RRT lebih efektif 3,2 poin dibandingkan dengan FSCS-ART pada kondisi R = 0,75. Penambahan fitur RRT memberikan hasil positif dengan rata-rata F-Measure 0,86759 kepada pengujian terhadap fungsi *GetCheckin*.

Seluruh hasil penelitian yang dilakukan diringkas ke dalam satu tabel berikut di bawah ini:

Tabel 11. Ringkasan Hasil Penelitian

No	Function	R	Average F-Meassure	
			FSCS-ART	FSCS-ART RRT
1	Bonus THR	0.25	8,25	5,75
		0.5	8,05	5,15
		0.75	8,05	5,55
2	Lembur	0.25	6,2	5,3
		0.5	5,6	2,4
		0.75	5,8	1,6
3	Checkclock	0.25	12,15	10,5
		0.5	12,25	10,3
		0.75	11,35	8,15

Tabel 10 mencerminkan F-Meassure rata-rata dari metoda FSCS-ART dan FSCS-ART RRT untuk setiap nilai R. Nilai F-Meassure pada fungsi *getCheckin* bernilai

ISSN: 2089-1121

paling tinggi dari hasil F-Meassure fungsi *getBonusTHR* dan fungsi *getOvertimeAmount* dikarenakan letak *error* pada fungsi *getCheckin* berada di percabangan yang jarang dilalui. Semakin besar nilai R maka akan mempercepat menemukan kandidat uji coba yang optimum.

4. KESIMPULAN

RRT memberikan dampak peningkatan pada FSCS-ART signifikan minimum 13% bila dibandingkan dengan nilai F-Meassure FSCS-ART tanpa RRT.
Nilai R yang semakin besar bisa memberikan peningkatan performa F-Meassure berkisar antara 10%.

FSCS-ART dan FSCS-ART-RRT sangat bergantung pada hasil pengacakan kasus uji coba, sehingga F-Meassure akan tidak selalu sama setiap kali pengujian.

Besarnya *input domain* merupakan parameter yang dimasukkan secara manual. Besar *input domain* yang tidak tepat akan membuat nilai *F-Meassure* semakin tidak efektif.

Letak *error* pada *source code* semakin tersembunyi dan jarang dilalui, maka semakin besar nilai F-Meassure.

5. DAFTAR PUSTAKA

Bentley, John E. *Software Testing Fundamentals—Concepts, Roles, and Terminology*. http://www2.sas.com/proceedings/sugi30/141-30.pdf. Mei 2011.

Chen, Tsong Yueh, Fei-Ching Kuo, Robert G. Merkel, T.H. Tse. Adaptive Random Testing: The ART of Test Case Diversity. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.150.4003.pdf. Juli 2011.

Chen, T. Y., F. C. Kuo, R. G. Merkel, S. P. Ng. Mirror Adaptive Random Testing. Texas : Third International Conference On Quality Software. 2003.

Khan, Mohd. Ehmer. Different Forms of Software Testing Techniques for Finding Errors. http://www.ijcsi.org/papers/7-3-1-11-16.pdf. Mei 2011.

Kuo, F. C., T. Y. Chen, H. Liu, W. K. Chan. Enhancing Adaptive Random Testing in High Dimensional Input Domains. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.108.1763.pdf. Mei 2011.

Liu, H., Xiaodong Xie, Jing Yang, Yansheng Lu, and Tsong Yueh. Adaptive Random Testing by Exclusion through Test Profile. http://www.computer.org/portal/web/csdl/doi/10.1109/QSIC.2010.61. Juli 2011.

Wikipedia bahasa Indonesia. Pengujian Perangkat Lunak. http://id.wikipedia.org/wiki/Pengujian_perangkat_lunak. Mei 2011.