PEMANFAATAN MIRROR ADAPTIVE RANDOM TESTING UNTUK UJI COBA SOFTWARE

Sunu Jatmika[1] dan Edwin Pramana[2]
[1]Sistem Komputer Sekolah Tinggi Manajemen Informatika & Komputer ASIA Malang
[2]Teknik Informatika Sekolah Tinggi Teknik Surabaya
sunu.srg@gmail.com dan epramana@stts.edu

ABSTRAK

Dalam bidang pengembangan perangkat lunak, *error* harus diminimalkan, sebab hal ini akan sangat merugikan bagi perusahaan pengembang perangkat lunak. Maka dikembangkan beberapa teori untuk pengujian. Salah satunya adalah Adaptive Random Testing (ART) dimana metode ini mampu mengelompokkan input yang menyebabkan kesalahan sehingga efektifitas dari random testing bisa ditingkatkan. Sifat dari ART adalah mampu menyesuaikan diri dengan keadaan dari software yang diuji. Pemilihan test case dilakukan dengan cara beradaptasi sesuai dengan kondisi tujuan yang ingin dicapai. Mirror Adaptive Random Testing (MART) adalah salah satu metode terkini yang merupakan pengembangan dari ART.

Dalam pengujian dengan menggunakan metode MART, domain input yang akan diuji dibagi menjadi subdomain. Salah satu subdomain akan ditunjuk sebagai subdomain asli dan dapat diselesaikan dengan algoritma ART. Jika test case yang dijalankan pada subdomain asli dan tidak diketemukan error maka test case diteruskan dengan cara di mirror (dicerminkan) pada subdomain lainnya. Dalam pengujiannya metode ini diterapkan pada software penggajian untuk modul penghitungan pajak karyawan dengan 4 domain input yaitu gaji per bulan, status, jumlah tanggungan dan penghasilan netto. Sedangkan untuk pengujian dengan 2 input domain diujikan pada modul penghitungan upah lembur karyawan dengan input domain yang digunakan yaitu lama lembur dan jenis hari lembur.

Dari program penggajian yang diuji dengan menggunakan metode MART mampu meningkatkan performance ujicobasecara significant dalam menemukan F-measure dibandingkan dengan metode sebelumnya yaitu ART.

Kata kunci: *Software Development, Adaptive Random Testing, Mirror adaptive Random Testing, Mirror, F-measure*

ABSTRACT

In the field of software development, errors should be minimized, because it will be very costly for the software developers. Several testing theories are invented. One of them is the Adaptive Random Testing.This method is able to classify the input that caused the error so that the effectiveness of random testing can be improved. The nature of ART is able to adjust to the situation of the software being tested. Test cases are chosen by adapting in accordance with the conditions of the objectives to be achieved. Mirror Adaptive Random Testing (MART) is one of the current development from ART.

In testing using domain MART input method to be tested is divided into subdomains. One subdomain will be designated as the original subdomain and can be solved by the algorithm ART. If the test cases run on the original subdomain and cannot find any errors, then the test case is ismirroredto the other subdomain. The payroll software for employee tax calculation module is tested with 4 input domains: salary per month, status, number of dependents and net income. Furthermore, the module is tested in calculating the employee salary with two input domains: overtime duration and overtime day of week.

Of the payroll program that tested using MART method can significantly improve the performance in finding the F-measure compared to the previous method, namely ART.

Keywords: Software Development, Adaptive Random Testing, Mirror Adaptive Random Testing, Mirror, F-measure

I. PENDAHULUAN

Dalam bidang *software development* (pengembang perangkat lunak), resiko *bug* (kesalahan) harus diminimalkan baik kesalahan yang disebabkan oleh fungsi logik ataupun kesalahan pada output yang salah. Sebab hal ini akan sangat merugikan bagi perusahaan pengembang software maupun pengguna software. Bagi pengembang perusahaan software kebanyakan dalam pengujian software dengan menggunakan pengujian *overtesting* (yang terlalu banyak) yang akan memiliki resiko dari penggunaan sumber daya, waktu dan biaya. Di satu sisi pengujian *undertesting* (yang terlalu sedikit) dimana resiko yang akan diakibatkan berakibat langsung pada saat software diimplementasikan pada pengguna.

Dalam penelitian ini penulis menggunakan pendekatan pengujian dengan menggunakan metode *black-box testing* guna mengetahui fungsi-fungsi software yang yang akan diuji. Adapun software yang akan diuji adalah sebagian dari produk-produk yang telah dihasilkan oleh CV. Mitra Informatika Solusindo yang merupakan perusahaan yang bergerak dalam bidang *software house.*

Uji coba software sangat penting untuk diterapkan dalam dalam pembentukan suatu program. Dengan adanya uji coba software dapat menjaga mutu dan kualitas program yang dibuat. Dalam penelitian ini akan dibandingkan performance metode ART dan MART untuk pengujian software yang telah disediakan dengan menggunakan 1 domain input, 2 domain input dan 4 domain input.

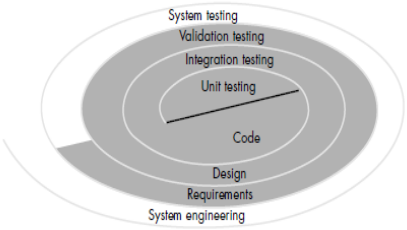
II. TINJAUAN PUSTAKA

Salah satu fase yang penting dalam pembuatan dan pengembangan *software* (perangkat lunak) adalah uji coba software (perangkat lunak). Tiga konsep yang harus diperhatikan dalam uji coba software (perangkat lunak) yaitu :

a. Demonstrasi validitas software (perangkat lunak) pada setiap tahapan pembangunan sistem
b. Penentuan validitas sistem akhir terhadap pemakai dan kebutuhan

c. Pemeriksaan implementasi sistem dengan menjalankan sistem pada suatu contoh data uji.

Pressman menggambarkan alur daripada proses *software engineering* dalam bentuk spiral, alur ini dapat dilihat seperti pada gambar 1.



Gambar 1. Strategi Testing

Maka ada dua bentuk pengujian perangkat lunak yaitu *white box testing* dan *black box testing*. *White box testing*, sering disebut sebagai *glass box testing*, merupakan pengujian yang didasarkan pada pengecekan secara detail terhadap perancangan program, sruktur kontrol dari desain program secara prosedural untuk membagi pengujian ke dalam beberapa kasus pengujian. *Black box testing* atau juga disebut *behavioral testing* adalah pengujian yang dilakukan hanya mengamati hasil eksekusi melalui data uji dan memeriksa fungsional dari perangkat lunak yang diuji, yang bisa dianalogikan seperti melihat kotak hitam yang hanya bisa melihat penampilan luarnya saja tanpa mengetahui apa isi dalamnya

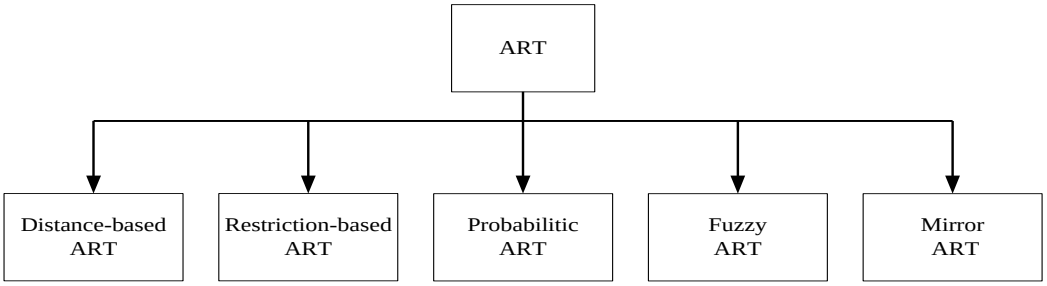
Pengujian dengan ***exhaustive testing*** sangat tidak disarankan untuk dilakukan karena terlalu banyak uji coba yang akan dijalankan dan waktu yang dibutuhkan sangat besar.

Karena pengujian dengan *exhaustive testing* tidak memungkinkan, maka mulai dikembangkan metode ART (adaptive random testing) dimana ART memilih test case dengan mengadaptasi sesuai kondisi tujuan yang ingin dicapai. Berikut adalah algoritma dari FSCS-ART.

```
1:   initial test data := randomly generate a test data from the
                          input domain;
2:   selected set := { initial test data };
3:   counter := 1;
4:   total number of candidates := 10;
5:   use initial test data to test the program;
6:   if (program output is incorrect) then
7:         reveal failure := true;
8:   Else
9:         reveal failure := false;
10:  end if
11:  while (not reveal failure) do
12:        candidate set := {};
13:        test data := Select The Best Test Data(selected set,
                        candidate set, total number of candidates);
14:        use test data to test the program;
15:        if (program output is incorrect) then
16:            reveal failure := true;
17:        Else
18:            selected set := selected set + { test data };
```

```
19:        end if
20:  end while
21:  output counter;
```

Sedangkan perkembangan dari metode adaptive random testing seperti pada gambar 2.



Gambar 2. Perkembangan Adaptive Random Testing

Mirror adaptive random testing dimana domain input dari perangkat lunak (*software*) yang akan diuji dibagi menjadi subdomain m beririsan. Satu subdomain adalah ditunjuk sebagai subdomain asli dan dapat diselesaikan dengan algoritma ART. Jika *test case* yang dijalankan pada subdomain asli tidak ditemukan *error rate* maka test case diteruskan dengan cara dicerminkan (mirror) pada subdomain lainnya. Berikut adalah algoritma dalam MART.

```
Let M(t, Di) be the image of t in Di generated by the chosen mirror
function. Let L be an array, storing the mirror domains in the order
to be tested. L[x] is used to denote the xth element of L.
Suppose there are 3 mirror domains {D2, D3, D4} to be tested in the
following order: D2, D4, D3. Thus, L[1], L[2] and L[3] contains D2,
D3 and D4, respectively.

1:   Set n = 0, E = { } and C = { }.
2:   Partition the input domain according to the chosen mirror
        partitioning.
3:   Randomly select a test case, t, from D1 according to the uniform
        distribution
4:   Increment n by 1.
5:   If t reveals a failure, go to Step 12; otherwise, store t in E.
6:   Determine L based on the chosen mirror selection order function
7:   For every L[x], where x from 1 to (h – 1), do Steps 7.1-7.2
     7.1 : Increment n by 1.
     7.2 : If M(t, L[x]) reveals a failure, go to Step 12.
8:   Randomly generate C = {c1, c2, ..., ck} from D1 according to the
        uniform distribution.
9:   For each element in C, find its nearest neighbor in E.
10:  In C, find which element, cj , has the longest distance to its
        nearest neighbor in E.
11:  Let t = cj and go to Step 4.
12:  Return n and t. EXIT.
```

III. RANCANGAN PENELITIAN

Pembahasan meliputi aktifitas uji coba dan karakteristik perangkat lunak yang diuji coba. Dalam penelitian ini pengujian dilakukan pada tiga kasus dengan input domain yang berbeda, yaitu dalam kategori simple, moderate dan kompleks.

a. Kasus Pengujian untuk Tingkat Simple

Dalam kasus ini pengujian diujikan pada fungsi perhitungan bonus tunjangan hari raya (THR) berdasarkan masa kerja karyawan.

Tabel 1. Bonus Tunjangan Hari Raya

No	Masa Kerja (Bulan)	%
11	< 3	0
22	>= 3 dan < 6	25
33	>= 6 dan < 9	50
44	>= 9 dan < 12	75
55	>= 12	100

b. Kasus Pengujian untuk Tingkat *Moderate*

Dalam kasus ini pengujian diujikan pada fungsi untuk perhitungan lembur dan absensi karyawan. Dalam pengujian menggunakan 2 input domain yaitu absensi dan lembur.

Tabel 2. Jenis Absensi

Kode	Singkatan	Jenis Absensi	Gaji
1	MI	Minggu	2 kali gaji pokok
2	MS	Masuk	1 kali gaji pokok
3	SH	Setengah hari	0.5 kali gaji pokok
4	DN	Dinas	1 kali gaji pokok
5	LK	Luar kota	0 kali gaji pokok
6	MB	Absen/Alpha	0 kali gaji pokok
7	PI	Ijin/Saki	0 kali gaji pokok

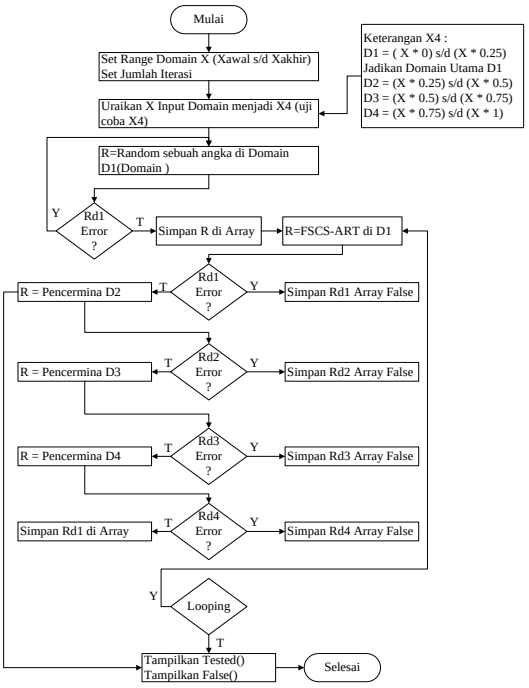
c. Kasus Pengujian untuk Tingkat Komplek

Dalam kasus ini pengujian diujikan untuk fungsi perhitungan pajak penghasilan dengan 4 domain.

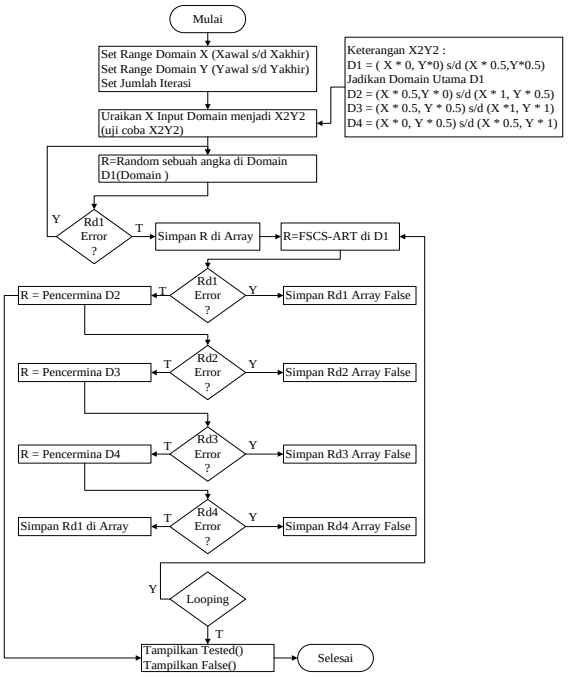
Tabel 3. Penghasilan Kena Pajak

No	Keterangan	Tarif Pajak
1	Penghasilan sampai dengan Rp 50.000.000	5%
2	Di atas Rp 50.000.000 sampai dengan Rp 250.000.000	15%
3	Di atas Rp 250.000.000 sampai dengan Rp 500.000.0000	25%
4	Di atas Rp 500.000.000	30%

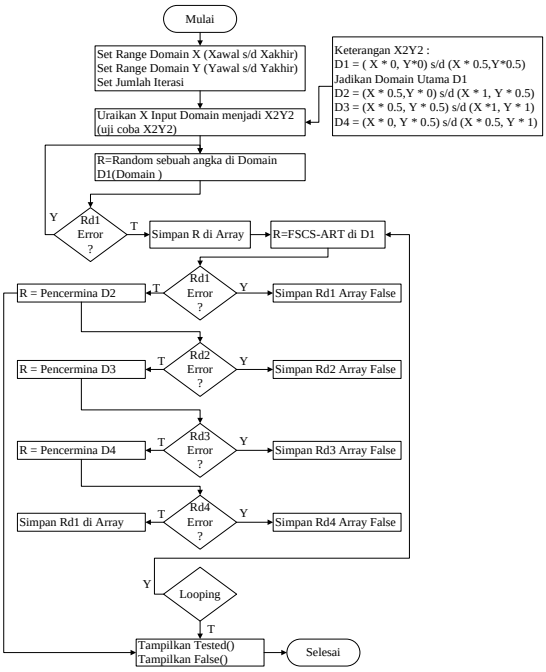
.



Gambar 3. Rancangan Penelitian 1 Input Domain



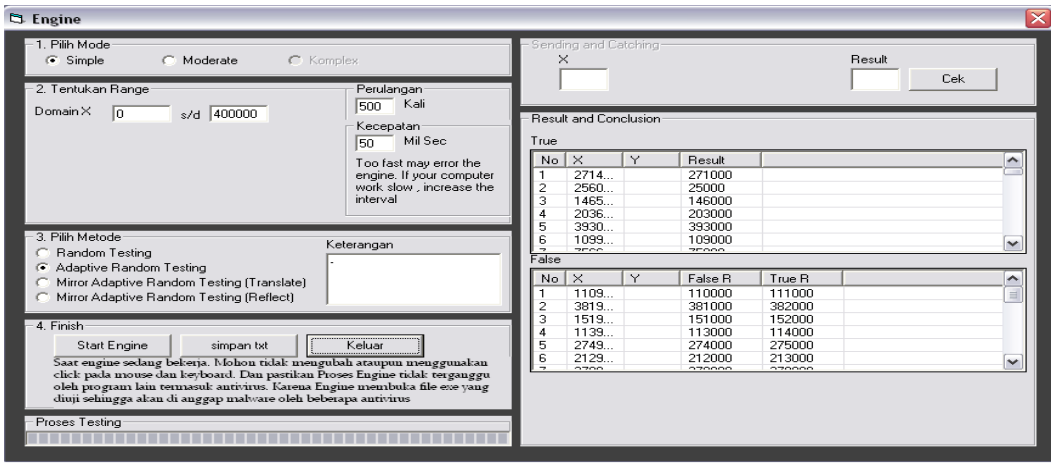
Gambar 4. Rancangan Penelitian Dua Input Domain



Gambar 5. Rancangan Penelitian 4 Input Domain

d. Software Engine

Program engine merupakan program yang didesain untuk menguji terhadap program yang akan diuji, dimana input akan di kirim dari program engine secara random sesuai metode yang dipakai.



Gambar 6. Software Engine IMPLEMENTASI DAN PENGUJIAN

e. Kasus Pengujian untuk Tingkat Simple

Hasil pengujian dengan menggunakan 1 input domain seperti pada tabel 4.

Tabel 4. Rekap Pengujian Untuk Mode Simple Dengan Satu Input Domain

Jumlah Uji Coba	Jumlah Iterasi	F-ART	F-MART Dengan Fungsi Translasi	F-MART Dengan Fungsi Reflect
10 Kali	50	1.31	1.02	1.00
10 Kali	250	1.67	1.01	1.00
10 Kali	500	1.13	1.00	1.00

Berdasarkan hasil pengujian antara metode ART dan MART dengan pengujian menggunakan *translate* dan *reflect* dengan jumlah iterasi pengujian seperti dalam tabel dihasilkan peningkatan diatas 11%.

f. Kasus Pengujian untuk Tingkat Moderate

Hasil pengujian dengan menggunakan 2 input domain seperti pada tabel 5.

Tabel 5. Rekap Pengujian Untuk Mode Moderate Dengan Dua Input Domain

Jumlah Uji Coba	Jumlah Iterasi	F-ART	F-MART Dengan Fungsi Translasi	F-MART Dengan Fungsi Reflect
10 Kali	50	38.46	8.77	11.11
10 Kali	250	29.76	5.27	4.22
10 Kali	500	20.24	4.18	4.36

Berdasarkan hasil pengujian antara metode ART dan MART dengan pengujian menggunakan *translate* dan *reflect* dengan jumlah iterasi pengujian seperti dalam tabel dihasilkan peningkatan diatas 70%.

g. Kasus Pengujian untuk Tingkat Komplek

Hasil pengujian dengan menggunakan 4 input domain seperti pada tabel 6

Tabel 6. Rekap Pengujian Untuk Mode Komplek Dengan Empat Input Domain

Jumlah Uji Coba	Jumlah Iterasi	F-ART	F-MART Dengan Fungsi Translasi	F-MART Dengan Fungsi Reflect
10 Kali	50	27.78	22.73	17.24
10 Kali	250	29.41	17.73	25.25
10 Kali	500	40.65	21.19	24.75

Berdasarkan hasil pengujian antara metode ART dan MART dengan pengujian menggunakan *translate* dan *reflect* dengan jumlah iterasi pengujian seperti dalam tabel dihasilkan peningkatan diatas 39%.

IV. PENUTUP

Dari hasil pengujian yang telah dilakukan, beberapa hal yang perlu diperhatikan menjadi kesimpulan, antara lain:

a. Sistem telah berhasil menggunakan satu sumber kasus uji coba acak yang dilakukan sebelum melakukan pengujian sehingga implementasi metode Adaptive Random Testing (ART) dan Mirror Adaptive Random Testing (MART) dapat menggunakan kasus uji coba dengan urutan kasus yang sama.
b. Sistem telah meningkatkan performance dalam menemukan F-measure diatas 5% untuk tiap pengujian dengan iterasi yang berbeda dengan dihitung berdasarkan F-Measure dan diterapkan dalam 10 kali pengujian untuk tiap iterasinya.
c. Semua pengujian akan berhenti sesuai lama iterasi yang telah ditentukan.
d. Pada saat pengujian dengan ART sangat tergantung pada hasil pengacakan kasus uji coba, sehingga hasil F-measure tidak selalu sama setiap kali pengujian.

V. DAFTAR PUSTAKA

[1] Dave Towey. *Adaptive Random Testing: Ubiquitous Testing to Support Ubiquitous Computing*. Division of Science and Technology. BNU-HKBU United International College.
[2] David Peter Towey. *Studies of Different Variations of Adaptive Random Testing*. Departemen of Science The University of Hong Kong. 2006.
[3] Fei-Ching Ku. *On adaptive Random Testing*. Swinburne University of Technology. 2006.
[4] F.C. Kuo. *Enhancing Adaptaive Random Testing in High Dimensional Input Domains*. SAC'07. Seoul Korea.
[5] Glenford J. Myers. *The Art of Software Testing – Second Edition*. John Wiley & Sons Inc. 2004.

[6] Kwok Ping Chan, T.Y. Chen, dan Dave Towey. *Probabilistic Adaptive Random Testing*. Proceeding of the Sixth International Conference on Quality Software. 2006.
[7] Laurie Williams. *Testing Overview and Black Box Testing Techniques*. IEEE. 2006.
[8] Roger S. Pressman. *Software Engineering A Practitioner's Approach*. Mc Graw Hill. 2011.
[9] T.Y. Chen,H. Leung, and I.K. Mak. *Adaptive Random Testing*. Swinburne University of Technology. 2004.
[10] T. Y. Chen, F. C. Kuo, dan R. G. Merkel,S. P.Ng. *Mirror Adaptive Random Testing*. Third International Conference On Quality Software Texas. 2003.