Jurnal Kreativitas Mahasiswa Informatika
Volume 2 Nomor 1 Tahun 2021 Page 152 - 155
p-ISSN: 2797-6327 e-ISSN: -

Pengujian Website Dengan Metode Black Box Berbasis Data Flow Testing

Achmad Hidayat[1], Andhika[2], Arief Prasetio[3], Erik Saputra[4]
[1-4]Teknik Informatika, Universitas Pamulang, Jl. Raya Puspitek No. 46 buaran, serpong, Kota Tangerang Selatan. Provinsi Banten 15310
[1-4]Jurusan Teknik Informatika, Fakultas Teknik, Universitas Pamulang

e-mail: [1]ahmadhidayat1873@gmail.com, [2]visilanddika93@gmail.com, [3]Ariefprasetio5@gmail.com, [4]erik.saputra1345@gmail.com

Abstrak

Penelitian ini dilakukan buat menjalankan & mengevaluasi sebuah aplikasi atau software secara manual juga otomatis. Perangkat lunak atau aplikasi yang akan di uji menggunakan Metode Black Box berbasis State Transition Testing. Pengujian aplikasi atau software ini adalah suatu fase yang sangat krusial pada menciptakan & membuatkan software atau aplikasi. State Transition Testing adalah pengujian yang serius dalam transisi perpindahan antar status, pengujian pada lakukan menggunakan mengecek apakah perpindahan status ke status lainya telah tepat, baik itu berdasarkan aksi yang dilakukan buat pindah status. Hasil berdasarkan penelitian ini akan menaruh rekomendasi & penilaian atas pengujian terhadap aplikasi atau software.

Kata kunci: uji coba fungsional, uji coba struktural,data flow anomaly, control flow graph(CFG).

I. PENDAHULUAN

Uji coba perangkat lunak sangat dibutuhkan dalam menentukan baik buruknya kualitas suatu perangkat lunak. Hal ini bisa dilakukan dengan dua cara yaitu ujicoba fungsional dan ujicoba struktural. Kedua jenis ini tidak dapat saling menggantikan karena keduanya mempunyai titik berat yang berbeda. Suatu perangkat lunak dikatakan bagus kualitasnya bila secara fungsional telah memenuhi requirement pengguna dan bagus dari segi strukturalnya.

Uji coba fungsional tak dapat menangani hal-hal yang berhubungan dengan struktural program. Untuk program dengan banyak kombinasi jenis input tidaklah tepat diuji dengan metode fungsional. Selain itu uji coba fungsional juga tak dapat menentukan.

Program dan dapat membantu dalam menilai kualitas suatu perangkat lunak.

II. METODE PELAKSANAAN

Banyak aturan/aturan yang dapat digunakan untuk tujuan pengujian perangkat lunak antara lain:

a. Pengujian adalah proses eksekusi program, tujuannya untuk menemukan kesalahan.
b. Kasus uji yang baik kemungkinan akan menemukan kesalahan yang belum pernah ditemukan sebelumnya.
c. Tes yang berhasil adalah tes yang menemukan kesalahan yang belum ditemukan sebelumnya.

Jika pengujian berhasil, pengujian akan menemukan bug/kesalahan pada perangkat lunak. Keuntungan lain adalah bahwa pengujian menunjukkan bahwa fungsi-fungsi dalam perangkat lunak bekerja sesuai kebutuhan sesuai dengan spesifikasi, perilaku program, dan kinerja. Namun, pengujian tidak dapat menunjukkan adanya kesalahan dan cacat

III. HASIL DAN PEMBAHASAN

Untuk mengetahui apakah proses berjalan sesuai dengan yang diharapkan, maka pelaksanaan eksperimen ini dilakukan melalui beberapa skenario. Skenario ini adalah:

- Uji fungsi sederhana tanpa pengecualian (Skenario1)
- Tes fungsi abnormal sederhana (Skenario 2)
- Uji fungsi yang melibatkan pemanggilan fungsi lain (Skenario III)
- Pengujian fungsional yang melibatkan cabang (Skenario IV)
- Uji fungsional yang melibatkan loop (skenario V)

Eksekusi setiap skenario adalah untuk memastikan bahwa aplikasi dapat berjalan di bawah kondisi setiap fungsi dalam skenario yang ditentukan, apakah itu fungsi sederhana tanpa

Jurnal Kreativitas Mahasiswa Informatika
Volume 2 Nomor 1 Tahun 2021 Page 152 - 155
p-ISSN: 2797-6327 e-ISSN: -

pengecualian, atau pengecualian, yang melibatkan pemanggilan fungsi lain, cabang atau loop. Fungsi sederhana mengacu pada fungsi yang hanya melibatkan pernyataan sederhana, seperti cin>>data1.

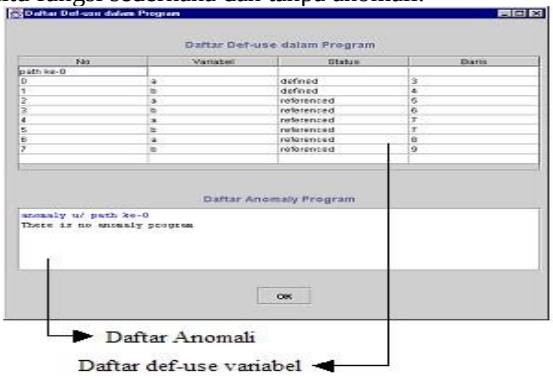
Fungsi yang melibatkan pemnggilan terhadap fungsi lain adalah fungsi yang salah satu pernyataan di dalamnya adalah pemanggilan terhadap fungsi lain, misal check_anomaly(vbes[i], vbes[j]). Fungsi yang melibatkan percabangan adalah fungsi yang didalamnya mengandung pernyataan yang mengakibatkan terjadinya percabangan dalam eksekusi program, misalnya pernyataan if-then-else dan switch-case. Sedangkan fungsi yang melibatkan perulangan adalah fungsi yang didalamnya mengandung pernyataan yang menyebabkan terjadinya perulangan dalam eksekusi program, seperti pernyataan while-do dan for.

Skenario I

Pada skenario I ini diujicobakan suatu fungsi yang sederhana, yaitu fungsi yang semua pernyataannya berjalan secara sekuensial tanpa percabangan maupun loop. Fungsi input yang dipilih pada skenario ini adalah void input_tukar () dengan kode program sebagai berikut:

```
0:    void input_tukar()
1:    {
2:          int a,b; a=55; b=77;
3:          cout<<"main() : a= "<<a<<endl;
4:          cout<<"main() : b= "<<b<<endl;
5:          tukar(a,b);
6:          cout<<"main() : a= "<<a<<endl;
7:          cout<<"main() : b= "<<b<<endl;
8:
9:
10:
          }
```

Pada daftar def-use gambar 1 tercantum bahwa variabel a dan b telah didefinisikan (status=defined) pada baris ketiga dan keempat sebelum digunakan serta tidak ada pengulangan pendefinisian setelah baris tersebut. Dengan demikian tidak ada data flow anomaly dalam fungsi input tersebut. Hasil ini telah sesuai dengan yang diharapkan, yaitu aplikasi dapat membaca baris pernyataan, memberi status tiap variable yang digunakan dalam fungsi serta melakukan pendeksian data flow anomaly dengan benar. Dengan demikian dapat disimpulkan bahwa aplikasi ini telah berhasil untuk skenario I, yaitu fungsi sederhana dan tanpa anomali.



Gambar 1. Def-use variabel skenario I

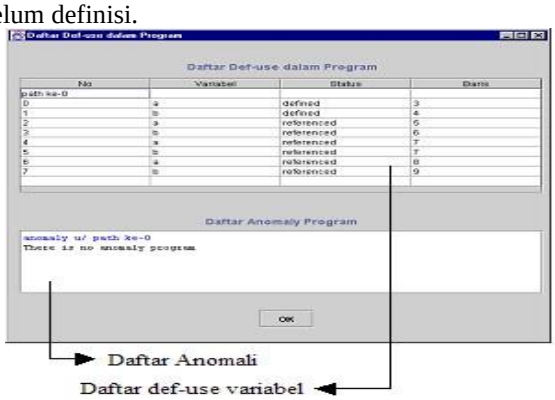
Skenario II

Pada skenario ini akan diujicobakan suatu fungsi yang sederhana, yaitu fungsi yang semua pernyataannya berjalan secara sekuensial tanpa percabangan maupun loop. Fungsi yang akan diinputkan dirancang untuk menghasilkan data flow anomaly. Hal ini dilakukan untuk mengetahui apakah perangkat lunak ini mampu mendeteksi keberadaan data flow anomaly sesuai yang diharapkan. File input yang diinputkan adalah void input_tukar2 ( ), dengan kode program sebagai berikut:

```
0:    void input_tukar2()
1:    {
2:          int a,b;
3:          int c=9;
4:        float d;
5:          a=55;
6:          b=77;
7:          cout<<"main() : a= "<<a<<endl;
8:          cout<<"main() : b= "<<b<<endl;
9:          tukar(a,b);
10:         cout<<"main() : a= "<<a<<endl;
11:         cout<<"main() : b= "<<b<<endl;
12:       cout<<"cek nilai d= "<<d<<endl; }
13:
```

Pada fungsi di atas digunakan empat variabel yaitu a, b, c, dan d. Variabel a dan b telah didefinisikan pada baris kelima dan keenam, dan kemudian direferensikan pada baris ketujuh hingga kesebelas. Variabel c didefinisikan pada baris ketiga dan tidak pernah digunakan. Variabel d direferensikan pada baris kedua belas dan tidak pernah didefinisikan.

Daftar def-use dari adegan ini dapat dilihat pada Gambar 2. Dalam daftar def-use, Anda dapat melihat bahwa variabel c didefinisikan di baris ketiga dan tidak pernah direferensikan oleh variabel lain di baris pernyataan berikut, menghasilkan pengecualian du. Dan di baris kedua belas, status variabel d direferensikan. Bahkan jika variabel belum pernah didefinisikan, sehingga terjadi pengecualian, yaitu diacu sebelum definisi.



Gambar 2. Def-use variabel skenario II

Skenario III

Dalam hal ini, fungsi yang melibatkan pemanggilan fungsi lain akan diuji. Saat memanggil suatu fungsi, parameter dalam fungsi yang dipanggil akan diberi status referensi, yang menunjukkan bahwa variabel telah direferensikan atau digunakan dalam fungsi panggilan. Fungsi yang cocok dengan situasi ini adalah fungsi void input_swap2(). Oleh karena itu, daftar def-use dan control flow graph (CFG) yang

Jurnal Kreativitas Mahasiswa Informatika
Volume 2 Nomor 1 Tahun 2021 Page 152 - 155
p-ISSN: 2797-6327 e-ISSN: -

dihasilkan akan sama dengan daftar yang dihasilkan pada Skenario II.

Dalam daftar def-use, variabel a dan b memiliki status referensi. Oleh karena itu, aplikasi dapat mendeteksi panggilan fungsi apa pun dalam program sehingga parameter dapat dibaca sebagai variabel dengan status referensi.

Skenario IV

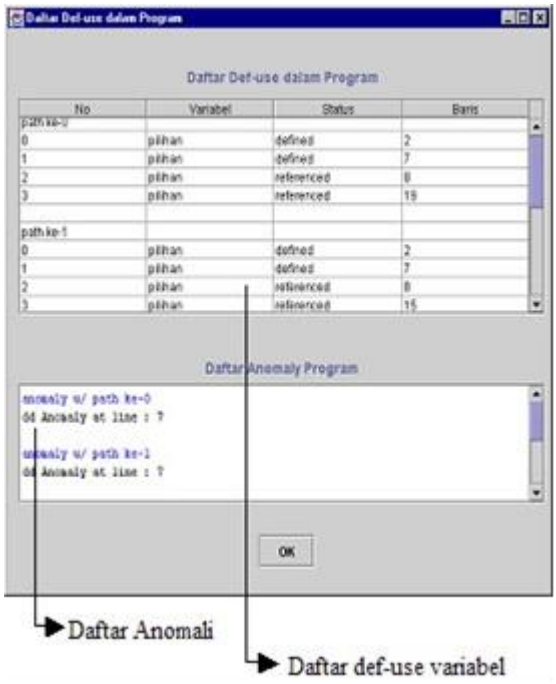
Dalam hal ini, fungsi yang melibatkan cabang akan diuji. Anda perlu memeriksa apakah aplikasi dapat mendeteksi beberapa jalur dalam fungsi input, dan apakah aplikasi dapat dengan benar menghasilkan grafik aliran kontrol (CFG) dari fungsi yang melibatkan percabangan.

Contoh percabangan adalah pernyataan switch-case. Pada fungsi yang akan dimasukkan, percabangan terjadi pada baris kedelapan, dan terdapat empat percabangan pada pernyataan case1, case2, case3, dan default. Variabel yang digunakan adalah pilihan. Di baris kedua, variabel telah didefinisikan dan didefinisikan ulang di baris ketujuh melalui input pengguna. Jadi fungsi ini berisi pengecualian dd pada baris ketujuh. Fungsi input dari pengujian ini adalah void main(), dan kode programnya adalah sebagai berikut:

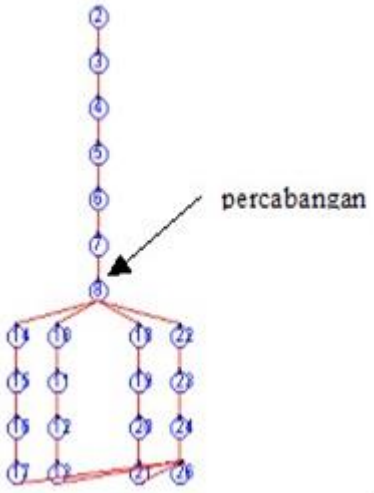
```
 0: void main()
 1: {
 2:  int pilihan=0;
 3: cout<<"pilih operasi yang diinginkan"<<"\n";
 4:    cout<<"1   =   jual   beli"<<"\t"<<"2   =
    ratarata"<<endl;
 5: cout<<"3 = membaca tahun"<<"\t"<<"4
    = tukar nilai"<<endl;
 6: cout<<"pilihan :"<<endl;
 7:  cin>>pilihan;
 8:  switch(pilihan)
 9:  {   10:
case 1:
11: cout<<"pilihan anda : "<<pilihan<<"yaitu jual
    beli"<<endl;
12: jual_beli();
13:  break; 14:
case   2:   15:
cout<<"piliha
n anda :
    "<<pilihan<<"yaitu jual beli"<<endl;
16: rata_rata();
17: break; 18:
case 3:
19: cout<<"pilihan anda : "<<pilihan<<"yaitu jual
    beli"<<endl;
20: membaca_tahun(2000,2004); 21:
break;
22: default:
23:         cout<<"pilihan anda :
            "<<pilihan<<"tak ada dalam
            option"<<endl;
24: printf("Tak ada dalam pilihan\n");
25: }
26: getch();
        27: }
```

Pada Gambar 3 Anda dapat melihat bahwa program telah bekerja secara normal, ia dapat mendeteksi cabang mana pun dengan memberikan daftar variabel def-use untuk setiap jalur yang berbeda. Hal yang sama berlaku untuk daftar pengecualian. Mendeteksi kelainan berdasarkan jalur yang dilalui. Misalnya, di jalur-1, ada pengecualian di baris ketujuh karena variabel pilihan telah didefinisikan atau berada dalam status yang ditentukan di baris kedua tetapi didefinisikan ulang di baris ketujuh.

Grafik aliran kontrol (CFG) dari fungsi void main() ditunjukkan pada Gambar 4. Gambar tersebut menunjukkan bahwa perangkat lunak telah berhasil membaca cabang dalam program, pada baris 8.



Gambar 3. Def-use variabel skenario IV



Gambar 4. Control Flow Graph Skenario IV

IV. SIMPULAN

Kesimpulan dari penelitian ini adalah sebagai berikut:

1. Tanpa benar-benar menjalankan program yang sedang diuji, daftar penggunaan (def-use) variabel dalam program dapat diperoleh melalui metode analisis aliran data statis.
2. Dengan menggunakan metode analisis aliran data statis, dapat diketahui apakah terdapat kelainan aliran data pada program pada semua jalur program yang ada. Jadi semua pernyataan program dilacak setidaknya sekali.
3. Grafik aliran kontrol (CFG) dapat dibangkitkan menggunakan kedalaman setiap pernyataan dalam program.kenyamanan user dalam mengakses sistem web tersebut.

Jurnal Kreativitas Mahasiswa Informatika
Volume 2 Nomor 1 Tahun 2021 Page 152 - 155
p-ISSN: 2797-6327 e-ISSN: -

DAFTAR PUSTAKA

1. W.K. Chan, T.Y. Chen, "An Overview of Integration Testing Techniques for ObjectOriented Programming". 2002. HKU CSIS Tech Report TR-2002-03.
2. Oliver Coul, "White-box Testing". 2003; Available from http://www.ddj.com/documents/s=887/ddj0003a/0003a.htm. Accessed March 19, 2002.
3. Doug Grant, "Project Descriptions". 2002; Available from http://www.it.swin.edu.au/centres/cse/projects.htm. Accessed October 17, 2002.
4. Roger S. Pressman, "Software Engineering: A Practioner's Approach, Fifth Edition". McGrawHill. 2001. h. 347-372.
5. Don Lance, Roland H.Untch, Nancy J. Wahl, "Bytecode-based Java Program Analysis". 1999..