Pengujian Struktur Program Dengan Pengujian Jalur Dasar (Basis Path Testing) : Teori Dan Aplikasi

Tri A. Kurniawan

***Abstrak*---Pengujian perangkat lunak adalah sebuah proses eksekusi program yang dilakukan untuk menemukan kesalahan-kesalahan (*errors*). Salah satu teknik yang digunakan adalah pengujian struktur program/kotak putih (*structural/white box/glass box testing*) dengan menggunakan pengujian jalur dasar (*basis path testing*).**

Pengujian jalur dasar adalah merupakan suatu teknik pengujian struktur program yang dilakukan berdasarkan jalur-jalur lojik yang ada. Jalur dasar (*basis path*) didefinisikan sebagai suatu jalur unik yang terdapat pada struktur program dimana tidak diperbolehkan terjadinya perulangan lintasan jalur yang sama persis. Semua kemungkinan jalur dasar diidentifikasi dengan melakukan kombinasi-kombinasi jalur yang ada. Untuk bisa melakukan pengujian jalur dasar maka harus ditentukan dulu grafik alir (*flow graph*) yang diterjemahkan dari algoritma/*flow chart* struktur program yang akan diuji. Pengujian jalur dasar menggunakan *cyclomatic complexity/number* untuk mengukur tingkat kompleksitas dari struktur program yang akan diuji sekaligus sebagai dasar untuk menentukan kasus-kasus uji (*test cases*) yang akan dilakukan. Pada pembahasan artikel ini akan dijelaskan bagaimana menggambarkan grafik alir berdasarkan algoritma/*flow chart* yang akan diuji, menghitung *cyclomatic complexity*, menentukan jalur-jalur dasar dan mendefinisikan kasus-kasus uji yang akan dilakukan.

Teknik pengujian jalur dasar diimplementasikan pada pengujian unit (*unit testing*) dan integrasi (*integration testing*). Teknik pengujian ini dapat digunakan untuk melakukan proses penjaminan mutu (*quality assurance*) terhadap produk perangkat lunak sebelum diserahkan kepada pengguna (*customer*).

***Kata kunci*--grafik alir, jalur dasar, simpul, sisi, *cyclomatic complexity*, kasus uji**

I. PENDAHULUAN

Pengujian perangkat lunak merupakan sebuah elemen kritis dari penjaminan mutu perangkat lunak (*software quality assurance*) dan merepresentasikan review secara keseluruhan terhadap hasil-hasil yang diperoleh dari analisis kebutuhan, perancangan dan implementasi perangkat lunak. Di dalam siklus hidup pengembangan perangkat lunak (*software development life cycle*), pengujian perangkat lunak memerlukan alokasi waktu antara 30% - 40% dari total waktu penyelesaian projek perangkat lunak [2]. Lamanya waktu yang diperlukan untuk melakukan pengujian menunjukkan betapa pentingnya sebuah perangkat lunak diuji sebelum produk perangkat lunak tersebut diserahkan kepada pengguna (*customer*).

Pengujian perangkat lunak adalah sebuah proses eksekusi program dengan maksud untuk menemukan kesalahan (*error*) [2]. Sehingga, dalam perspektif pengujian ini maka sebuah pengujian perangkat lunak dikatakan berhasil kalau mampu mendefinisikan sebuah kasus uji yang memiliki probabilitas yang tinggi untuk memunculkan kesalahan dari program yang sedang diuji.

Berdasarkan sumber informasi yang digunakan untuk menentukan kasus uji maka teknik pengujian dibagi menjadi 2 yaitu pengujian kotak hitam (*black box/functional testing*) dan pengujian kotak putih/struktur (*white box/structural testing*) [1]. Pengujian kotak hitam dilakukan berdasarkan spesifikasi fungsi dari perangkat lunak dan sama sekali tidak melihat bagaimana struktur program yang ada. Sebaliknya, pengujian kotak putih dilakukan dengan melihat struktur kontrol dari program. Kedua teknik tersebut dilakukan untuk saling melengkapi dan tidak saling meniadakan, dimana masing-masing teknik masih memiliki beberapa jenis teknik pengujian yang berbeda.

Pada artikel ini akan dibahas salah satu teknik pengujian yang digunakan pada pengujian kotak putih yaitu pengujian jalur dasar (*basis path testing*) sebagai salah satu teknik pengujian yang cukup penting digunakan untuk mendapatkan pengujian yang efektif dan efisien.

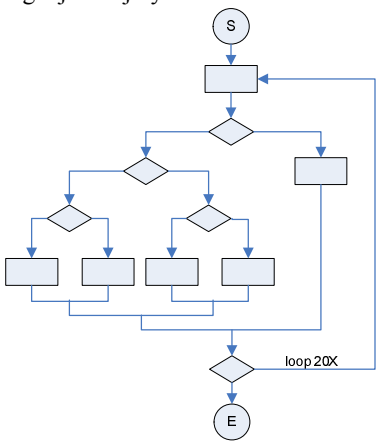
II. TEORI

Pengujian jalur dasar merupakan salah satu teknik pengujian kotak putih yang pertama kali diperkenalkan oleh Tom McCabe pada tahun 1976 [2]. Pengujian ini memungkinkan perancangan kasus uji yang diturunkan dari pengukuran tingkat kompleksitas (*cyclomatic complexity*) struktur program. Ukuran tingkat kompleksitas tersebut menjadi panduan di dalam menentukan jalur-jalur dasar (*basis path*). Kasus uji yang didapat menjadi kasus uji yang digunakan untuk membuktikan bahwa setiap pernyataan (*statement*) dari program akan dieksekusi minimal sekali.

Pengujian jalur dasar digunakan untuk efisiensi dan

Naskah diterima pada tanggal 22 Mei 2007.
Tri A. Kurniawan bekerja di Jurusan Teknik Elektro, Fakultas Teknik, Universitas Brawijaya. Alamat email penulis triak@brawijaya.ac.id.

efektifitas pengujian kotak putih, karena di dalam pengujian kotak putih tidak mungkin seluruh kemungkinan jalur dieksekusi. Sebagai ilustrasi, jika terdapat sebuah struktur program sebagaimana terdapat pada Gambar 1 dengan 5 kemungkinan jalur untuk setiap *loop*, maka akan terdapat 5^{20} kemungkinan jalur yang harus diuji [1]. Jika setiap jalur membutuhkan waktu pengujian selama 1 milidetik maka total waktu yang dibutuhkan untuk pengujian sebanyak jalur tersebut adalah 3.170 tahun. Hal ini sangat tidak mungkin ! Oleh karena itu, dibutuhkan sebuah teknik pengujian yang bisa merepresentasikan jalur-jalur pengujian yang ada dengan menggunakan jalur-jalur dasar sebagai jalur ujinya.



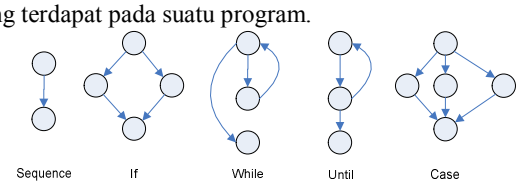
Gambar 1. Flow Chart Struktur Program
Sumber: [1]

Langkah-langkah untuk melakukan pengujian jalur dasar adalah [2] :

- Menggambarkan grafik alir (*flow graph*) berdasarkan algoritma perancangan prosedur/fungsi.
- Menentukan *cyclomatic complexity*.
- Menentukan jalur-jalur dasar sesuai dengan jumlah dari *cyclomatic complexity*.
- Mendefinisikan kasus-kasus uji untuk setiap jalur dasar yang telah ditentukan.

A. *Grafik Alir*

Grafik alir adalah sebuah notasi sederhana yang merepresentasikan aliran kontrol dari sebuah struktur program. Gambar 2 menjelaskan notasi-notasi yang digunakan untuk merepresentasikan jenis-jenis kontrol yang terdapat pada suatu program.



Gambar 2. Notasi Grafik Alir
Sumber: [3]

Dalam sebuah grafik alir, anak panah disebut sebagai sisi (*edge, E*) merepresentasikan aliran kontrol; lingkaran disebut sebagai simpul (*node, N*) merepresentasikan satu atau lebih aksi/pernyataan lojik; daerah yang dibatasi oleh sisi dan simpul disebut area (*region, R*); simpul yang mengandung keputusan disebut sebagai *predicate node, P* yaitu simpul yang mengeluarkan lebih dari satu sisi.

Beberapa panduan penting dalam penggambaran sebuah grafik alir adalah [2] :

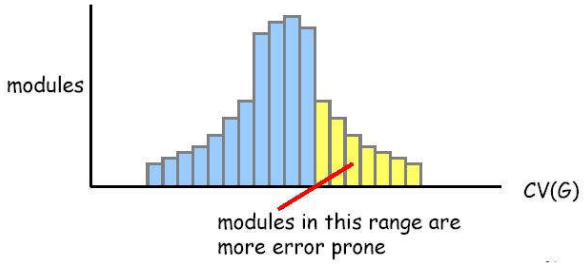
- Serangkaian aksi/pernyataan lojik dan sebuah pernyataan desisi bisa direpresentasikan dalam sebuah sisi.
- Sebuah sisi harus berakhir pada sebuah simpul, walaupun simpul tersebut tidak merepresentasikan sebuah aksi apapun.
- Penghitungan jumlah area melibatkan sebuah area yang berada di luar grafik alir.
- Desisi majemuk (*compound decision*), yaitu desisi yang mengandung operator Boolean (AND, OR, NAND dan NOR) tidak boleh direpresentasikan dalam sebuah simpul.

B. Cyclomatic Complexity

Cyclomatic complexity, *V(G)* adalah sebuah besaran perangkat lunak yang menyatakan ukuran tingkat kompleksitas sebuah program [2]. Angka ini menentukan jumlah jalur dasar yang harus diuji minimal sekali dari sebuah program. Secara matematis, *V(G)* dapat ditentukan salah satu cara dari beberapa pendekatan berikut :

- $V(G)$ = jumlah area (R)
- $V(G) = E - N + 2$
- $V(G) = P + 1$

Berdasarkan studi yang dibuat oleh sejumlah industri, semakin besar nilai *V(G)* maka semakin besar probabilitas terjadinya kesalahan program[1].



Gambar 3. Korelasi Nilai *V(G)* dengan Kesalahan
Sumber: [1]

C. *Jalur Dasar*

Jalur dasar adalah sebuah jalur pada program yang mengandung paling sedikit sebuah pernyataan lojik [2]. Dalam konteks grafik alir, maka sebuah jalur disebut sebagai jalur dasar jika jalur tersebut memiliki paling tidak satu buah simpul yang belum pernah dilalui oleh jalur yang sudah didefinisikan sebelumnya.

Jumlah jalur dasar yang teridentifikasi harus sesuai dengan nilai dari *cyclomatic complexity*, *V(G)*.

D. *Penentuan Kasus Uji*

Kasus uji adalah suatu kondisi yang dibuat untuk menjamin bahwa sebuah jalur dasar dapat diuji. Sebuah kasus uji harus memuat kondisi/data awal, prosedur uji

dan hasil yang diharapkan dari sebuah jalur dasar yang akan diuji. Sebuah jalur dasar mungkin tidak bisa diuji secara mandiri, tetapi harus menjadi bagian dari kasus uji jalur dasar yang lain.

III. APLIKASI

Untuk mengaplikasikan teknik pengujian jalur dasar maka akan digunakan algoritma dari sebuah prosedur pencarian biner (*binary search*), sebagaimana diperlihatkan pada Gambar 4.

```
Procedure BinSearch (Input  tabInt : integer[],
                            X : integer,n : integer
                     Output idX: integer)
{Prosedur mencari nilai X dr sebuah array integer}
{tabInt yang terurut menaik, tabInt0 ≤ tabInt1 ≤...}
{≤ tabIntn, idX akan bernilai -1 jika X tidak}
{ditemukan atau bernilai posisi index dari X yang}
{ditemukan}

Kamus:
kiri, kanan, tengah : integer (indeks pencarian)
found : boolean (TRUE jika ditemukan)

Algoritma:
kiri = 0
found = FALSE
kanan = n(panjang array tabInt) - 1
idX = -1
WHILE (kiri ≤ kanan) AND (NOT found) DO
  tengah = (kiri + kanan)/2
  IF (X = tabIntTengah) THEN
    found = TRUE
    idX = tengah
  ELSE IF (X < tabIntTengah)
    kanan = tengah - 1
  ELSE
    kiri = tengah + 1
  ENDIF
ENDDO
```

Gambar 4. Algoritma Pencarian Biner
Sumber: [4]

Langkah awal adalah menentukan simpul-simpul pada algoritma untuk membangun grafik alir, diperlihatkan pada Gambar 5.

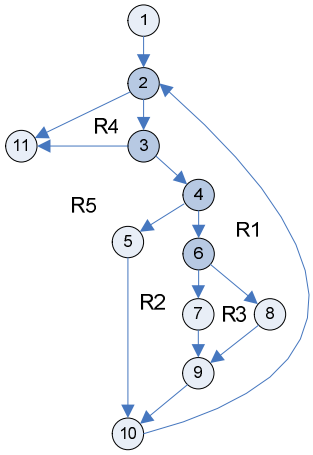
```
Procedure BinSearch (Input  tabInt : integer[],
                            X : integer,n : integer
                     Output idX: integer)
{Prosedur mencari nilai X dr sebuah array integer}
{tabInt yang terurut menaik, tabInt0 ≤ tabInt1 ≤...}
{≤ tabIntn, idX akan bernilai -1 jika X tidak}
{ditemukan atau bernilai posisi index dari X yang}
{ditemukan}

Kamus:
kiri, kanan, tengah : integer (indeks pencarian)
found : boolean (TRUE jika ditemukan)

Algoritma:
kiri = 0                                   (2)
found = FALSE                              (3)
kanan = n(panjang array tabInt) - 1        (1)
idX = -1
WHILE (kiri ≤ kanan) AND (NOT found) DO    (1)
  tengah = (kiri + kanan)/2
  IF (X = tabIntTengah) THEN               (4)
    found = TRUE
    idX = tengah                           (5)
  ELSE IF (X < tabIntTengah)
    kanan = tengah - 1                     (6)
  ELSE
    kiri = tengah + 1                      (7)
  ENDIF
ENDDO                                      (8)
                                           (9) (10)
                                           (11)
```

Gambar 5. Simpul pada Algoritma Pencarian Biner
Sumber: Hasil Perancangan

Selanjutnya adalah membangun grafik alir berdasarkan hasil penentuan simpul, diperlihatkan pada Gambar 6.



Gambar 6. Grafik Alir Pencarian Biner
Sumber : Hasil Perancangan

Dari grafik alir pada Gambar 6 didapatkan jumlah area adalah 5 yaitu R1, R2, R3, R4 dan R5; jumlah *predicate node* adalah 4 yaitu simpul 2, 3, 4 dan 6; jumlah simpul adalah 11 dan sisi sebanyak 14. Sehingga, nilai *cyclomatic complexity V(G)* adalah :

- $V(G)$ = jumlah area (R) = 5
- $V(G) = E - N + 2 = 14 - 11 + 2 = 5$
- $V(G) = P + 1 = 4 + 1 = 5$

Berdasarkan nilai $V(G)$ maka jumlah jalur dasar yang harus diidentifikasi adalah sebanyak 5 buah, yaitu :

- Jalur dasar 1: ***1-2-11***
- Jalur dasar 2: 1-2-***3***-11
- Jalur dasar 3: 1-2-3-***4-5-10***-2
- Jalur dasar 4: 1-2-3-4-***6-7-9***-10-2-...
- Jalur dasar 5: 1-2-3-4-6-***8***-9-10-2-...

Nomor simpul yang dituliskan tebal dan miring pada jalur dasar menunjukkan simpul-simpul yang belum pernah dilewati oleh jalur dasar sebelumnya. Sehingga, dapat dilihat bahwa setiap simpul paling tidak sudah terlewati minimal sekali.

Langkah selanjutnya adalah mendefinisikan kasus-kasus uji berdasarkan jalur dasar yang ada :

- Kasus uji jalur dasar 1: ***1-2-11***
  tabInt = array integer kosong, **n** = 0
  X = nilai integer sembarang
  Expected result = idX bernilai -1
- Kasus uji jalur dasar 2: 1-2-***3***-11
  tabInt = array integer berisi *n* data terurut menaik
  X bukan anggota dari **tabInt**
  Expected result = idX bernilai -1
- Kasus uji jalur dasar 3: 1-2-3-***4-5-10***-2
  tabInt = array integer berisi *n* data terurut menaik
  X anggota dari **tabInt** pada urutan pertengahan ((*n* + 1)/2)
  Expected result = idX bernilai ((*n* + 1)/2) – 1
- Kasus uji jalur dasar 4: 1-2-3-4-***6-7-9***-10-2-...
  tabInt = array integer berisi *n* data terurut menaik
  X bisa anggota dari **tabInt** atau bukan, **X** < nilai **tabInt** pada urutan pertengahan ((*n* + 1)/2)

Expected result = idX bernilai -1 jika **X** bukan anggota **tabInt** ; idX bernilai antara 0 sampai ((n + 1)/2) – 1 jika **X** anggota **tabInt**
Keterangan: jalur ini tidak bisa diuji mandiri, harus menjadi bagian dari pengujian jalur dasar 3.

- Kasus uji jalur dasar 5: 1-2-3-4-6-***8***-9-10-2-...
  tabInt = array integer berisi n data terurut menaik
  X bisa anggota dari **tabInt** atau bukan, **X** > nilai **tabInt** pada urutan pertengahan ((n + 1)/2)
  Expected result = idX bernilai -1 jika **X** bukan anggota **tabInt** ; idX bernilai antara ((n + 1)/2) – 1 sampai (n – 1) jika **X** anggota **tabInt**
  Keterangan: jalur ini tidak bisa diuji mandiri, harus menjadi bagian dari pengujian jalur dasar 3.

Contoh implementasi algoritma pencarian biner dalam bahasa C diperlihatkan pada Gambar 7.

```
void BinSearch(int tabInt[],int size,int X,int* idX){
  int kiri=0, kanan=size-1, tengah;
  *idX=-1;
  boolean found=false;
  while((kiri<=kanan)&&(!found)){
    tengah=(kiri+kanan)/2;
    if(X==tabInt[tengah]){
      found=true;
      *idX=tengah;
    }else if(X<tabInt[tengah]){
      kanan=tengah-1;
    }else{
      kiri=tengah+1;
    }
  }
}
```

Gambar 7. Implementasi Pencarian Biner dalam Bahasa C
Sumber : Hasil Implementasi

Pengujian jalur dasar bisa dilakukan dengan menggunakan nilai riil untuk **tabInt** dan **X** yang ditentukan berdasarkan aturan umum yang ada pada kasus-kasus uji yang telah didefinisikan. Tabel 1 memperlihatkan contoh nilai-nilai **tabInt** dan **X** yang bisa digunakan dalam pengujian.

TABEL 1. CONTOH DATA UJI PENCARIAN BINER

Jalur	tabInt	n	X	Exp. Result, idX
1	{}	0	12	-1
2	{2, 4, 7, 10, 11}	5	1	-1
3	{2, 4, 7, 10, 11}	5	7	2
4	{2, 4, 7, 10, 11}	5	4	1
5	{2, 4, 7, 10, 11}	5	10	3

IV. KESIMPULAN

Pada artikel ini sudah dijelaskan teknik pengujian jalur dasar beserta contoh bagaimana teknik tersebut diaplikasikan pada sebuah algoritma pencarian biner. Beberapa hal yang bisa ditarik kesimpulan adalah :

- Terdapat banyak teknik pengujian yang bisa dilakukan untuk menjamin kualitas perangkat lunak sebelum diserahkan kepada pengguna, salah satunya adalah teknik pengujian jalur dasar.
- Pengujian jalur dasar digunakan untuk menguji struktur program dengan menggunakan jalur-jalur dasar sebagai representasi dari sekian banyak kemungkinan jalur eksekusi yang terjadi.
- Untuk bisa melakukan pengujian jalur dasar maka langkah-langkah yang harus dilakukan adalah: membangun grafik alir, menghitung *cyclomatic complexity*, menentukan jalur-jalur dasar dan mendefinisikan kasus-kasus uji.
- Teknik pengujian jalur dasar diimplementasikan pada pengujian unit dan pengujian integrasi.

DAFTAR PUSTAKA

[1] Lundqvist, I. K., *Introduction to Computers and Programming*, http://ocw.mit.edu/NR/rdonlyres/Aeronautics-and-Astronautics/16-01Fall-2005-Spring-2006/3AFC37EE-19C9-4D31-947B-23F51919DE20/0/13_testing.pdf, 17 Januari 2007.

[2] Pressman, Roger S., 2001, *Software Engineering – A Practitioner's Approach*, Mc-Graw Hill, Fifth Edition.

[3] Sobey, A. J., *Software Testing*, http://www.erau.edu/research/veritas/pdf/SoftwareTesting.pdf, 17 Januari 2007.

[4] Sommerville, Ian, 1996, *Software Engineering*, Addison-Wesley, Fifth Edition.

[5] Williams, Laurie, 2004, *White-Box Testing*, http://agile.csc.ncsu.edu/SEMaterials/WhiteBox.pdf, 17 Januari 2007.