

BAB 6

METODE PENGUJIAN

Metode pengujian adalah cara atau teknik untuk menguji perangkat lunak, mempunyai mekanisme untuk menentukan data uji yang dapat menguji perangkat lunak secara lengkap dan mempunyai kemungkinan tinggi untuk menemukan kesalahan.

Perangkat lunak dapat diuji dengan dua cara, yaitu :

1. Pengujian dengan menggunakan data uji untuk menguji semua elemen program (data internal, loop, logika, keputusan dan jalur). Data uji dibangkitkan dengan mengetahui struktur internal (kode sumber) dari perangkat lunak.
2. Pengujian dilakukan dengan mengeksekusi data uji dan mengecek apakah fungsional perangkat lunak bekerja dengan baik. Data uji dibangkitkan dari spesifikasi perangkat lunak.

1. WHITE BOX TESTING

Pengujian *white box* (*glass box*) adalah pengujian yang didasarkan pada pengecekan terhadap detail perancangan, menggunakan struktur kontrol dari desain program secara procedural untuk membagi pengujian ke dalam beberapa kasus pengujian. Penentuan kasus uji disesuaikan dengan struktur system, pengetahuan mengenai program digunakan untuk mengidentifikasi kasus uji tambahan.

Tujuan penggunaan white box untuk menguji semua statement program.

Penggunaan metode pengujian *white box* dilakukan untuk :

1. memberikan jaminan bahwa semua jalur independen suatu modul digunakan minimal satu kali
2. menggunakan semua keputusan logis untuk semua kondisi *true* atau *false*
3. mengeksekusi semua perulangan pada batasan nilai dan operasional pada setiap kondisi.
4. menggunakan struktur data internal untuk menjamin validitas jalur keputusan.


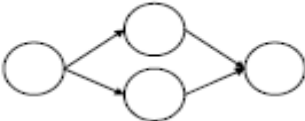
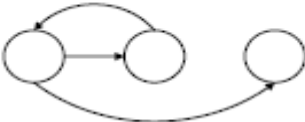
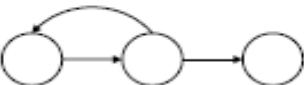
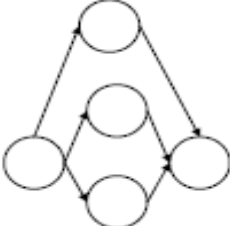
1.1 Pengujian Basis Path

Pengujian basis path adalah pengujian white box yang diusulkan pertama kali oleh Tom McCabe. Metode ini memungkinkan penguji dapat mengukur kompleksitas logis dari desain procedural dan menggunakannya sebagai pedoman untuk menetapkan himpunan basis dari semua jalur eksekusi.

A. Notasi Diagram Alir

Notasi yang digunakan untuk menggambarkan jalur eksekusi adalah notasi diagram alir (atau grafik program), yang menggunakan notasi lingkaran (simpul atau node) dan anak panah (link atau edge). Notasi ini menggambarkan aliran control logika yang digunakan dalam suatu bahasa pemrograman.

Tabel – Notasi Diagram Alir

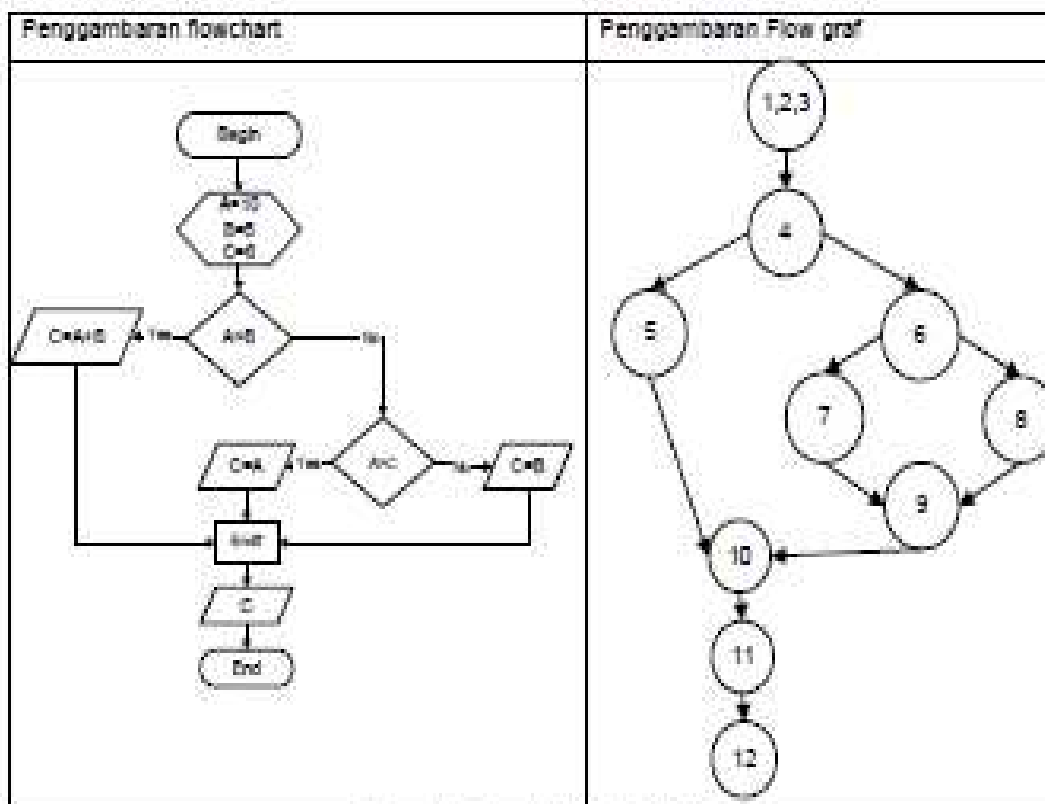
NOTASI	ARTI
	Langkah sequence
	Skema If
	Skema While (...) Do (...)
	Skema Repeat (...) Until (...)
	Skema Case (...)Of

Contoh Suatu PDL

```

Var
    A, B, C : integer
Begin
    A := 10;           (1)
    B := 5;             (2)
    C := 6;             (3)
    If A > B            (4)
        then C := A + B (5)
        Else if A > C   (6)
            then C := A (7)
            Else C := B; (8)
        Endif           (9)
    Endif               (10)
    Writeln('Nilai C = ', C); (11)
End.                  (12)

```



Gambar – Contoh Metode Basis Path

B. Kompleksitas Siklomatis

Kompleksitas Siklomatis adalah metrics perangkat lunak yang memberikan pengukuran kuantitatif terhadap kompleksitas logis suatu program, nilai yang didapat akan menentukan jumlah jalur independen dalam himpunan path, serta akan memberi nilai batas atas bagi jumlah pengujian yang harus dilakukan, untuk memastikan bahwa semua pernyataan telah dieksekusi sedikitnya satu kali.

Jalur independen adalah jalur yang terdapat dalam program yang mengintroduksi sedikitnya satu rangkaian pernyataan proses atau kondisi baru.

Berdasarkan contoh PDL sebelumnya, maka jalur independent yang didapat :

Jalur 1 : 1,2,3 – 4 – 5 – 10 – 11 – 12

Jalur 2 : 1,2,3 – 4 – 6 – 7 – 9 – 10 – 11 – 12

Jalur 3 : 1,2,3 – 4 – 8 – 9 – 10 – 11 – 12

Penghitungan Siklomatis

1. Region (daerah muka) grafik alir

2. $V(G) = E - N + 2$

E adalah jumlah edge, dan N adalah jumlah node

3. $V(G) = P + 1$

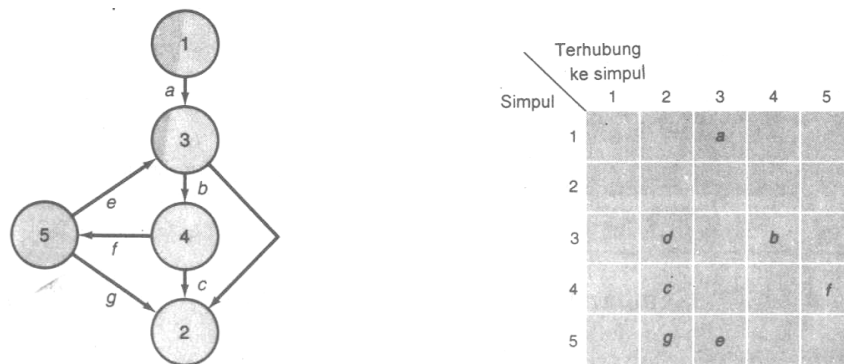
P adalah simpul predikat

Simpul Predikat adalah penggambaran suatu node yang memiliki satu atau lebih inputan, dan lebih dari satu output.

C. Matriks Grafis (Graph Matrik)

Bentuk struktur data yang sering digunakan untuk menggambarkan pengujian adalah dengan matriks grafis. Matriks grafis adalah matriks bujursangkar yang berukuran sama dengan jumlah simpul pada grafik alir. Inputan dalam matriks harus bersesuaian dengan arah sisi dengan simpul.

Matriks grafis selanjutnya disebut sebagai matriks koneksi, dan digambarkan serupa dengan matriks ketetanggaan dengan memperhatikan arah in-out dari edge.



Gambar – Contoh Matriks Koneksi

Pada gambar flowgraph masing-masing node ditandai dengan angka dan edge dengan huruf kecil, kemudian diterjemahkan ke graph matrik. Contoh hubungan node 3 dengan node 4 pada graph ditandai dengan huruf b. Hubungan bobot menyediakan tambahan informasi tentang aliran kontrol. Secara simpel hubungan bobot dapat diberi nilai 1 jika ada hubungan antara node atau nilai 0 jika tidak ada hubungan.

		Terhubung ke simpul				
		Simpul				
		1	2	3	4	5
1				1		
2						
3			1		1	
4			1			1
5			1	1		

Gambar - Hubungan bobot

Koneksi :

Simpul 1 : $1 - 1 = 0$

Simpul 2 : $2 - 1 = 1$

Simpul 3 : $2 - 1 = 1$

Simpul 4 : $2 - 1 = 1$

$$\begin{aligned} \text{cyclomatic complexity} &= \text{Jumlah simpul} + 1 \\ &= 3 + 1 = 4 \end{aligned}$$

1.2 Pengujian Struktur Kontrol

Teknik pengujian basis path merupakan salah satu dari sejumlah teknik untuk pengujian struktur control, namun pengujian basis path tidak memadai untuk beberapa kasus uji.

A. Pengujian Kondisi

Pengujian kondisi menggunakan kondisi logis sederhana yang terdapat dalam program. Kondisi sederhana dengan menggunakan variable Boolean, dengan bentuk persamaan.

$$E1 \text{ (operator-relasional) } E2$$

Bila suatu kondisi tidak benar, maka akan terdapat paling tidak satu komponen dari kondisi yang salah, sehingga tipe kesalahan pada suatu kondisi meliputi :

- 1) Kesalahan operator Boolean
- 2) Kesalahan variable Boolean
- 3) Kesalahan tanda kurung Boolean
- 4) Kesalahan operator relasional
- 5) Kesalahan persamaan aritmatika

Contoh :

C1 : B1 & B2

B1 & B2 adalah dua variable Boolean sehingga akan terdapat 2^2 pengujian. Dengan area pengujian :

B ₁ = T ; B ₂ = T	{(T,T), (T,F), (F,T), (F,F)}
B ₁ = T ; B ₂ = F	
B ₁ = F ; B ₂ = T	
B ₁ = F ; B ₂ = F	

B. Pengujian Aliran Data

Metode pengujian aliran data melakukan pengujian dengan menggunakan definisi variable dalam program, efektif digunakan untuk melindungi kesalahan, tetapi akan memiliki cakupan pengukuran dan pemilihan jalur uji yang kompleks.

C. Pengujian Loop

Loop atau skema pengulangan sering digunakan dalam pembuatan program. Terdapat beberapa macam loop, yaitu :

1. Loop Sederhana

Pengujian yang dilakukan harus memperhatikan hal-hal berikut :

- Mengabaikan keseluruhan loop
- Hanya terdapat satu jalur yang melewati loop
- Suatu variable akan melewati loop jika bernilai lebih besar dari nilai yang ditentukan
- Harus memperhatikan batasan variable pada loop. (kondisi : $n-1$, n , $n+1$)

2. Loop Tersarang

Jumlah pengujian akan bertambah secara geometris sesuai jumlah persarangan loop yang ada. Untuk menyederhanakan pengujian, maka harus diperhatikan hal-hal berikut:

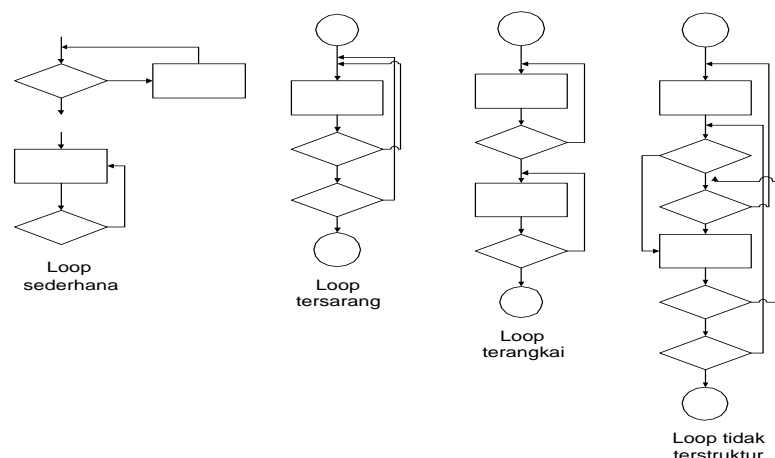
- Memulai pengujian dari loop terdalam, dengan memulai pengujian dari nilai minimum
- Melakukan pengujian loop sederhana dari loop terdalam hingga loop terluar, dengan memperhatikan parameter yang digunakan.

3. Loop Terangkai

Pengujian untuk loop terangkai harus disesuaikan dengan independensi variable antara loop tersebut. Jika variable yang digunakan dalam loop kedua tidak bergantung dengan loop pertama, maka digunakan pengujian loop sederhana, sedangkan bila loop kedua bergantung secara nilai dengan loop kedua, maka lakukan pengujian tersarang.

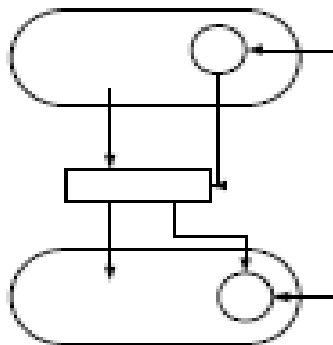
4. Loop Tidak Terstruktur

Sama sekali tidak dianjurkan untuk digunakan dalam membuat program.



2. BLACK BOX TESTING

Pengujian black box merupakan pendekatan komplementer dari teknik white box, karena pengujian black box diharapkan mampu mengungkap kelas kesalahan yang lebih luas dibandingkan teknik white box. Pengujian black box berfokus pada pengujian persyaratan fungsional perangkat lunak, untuk mendapatkan serangkaian kondisi input yang sesuai dengan persyaratan fungsional suatu program.



Gambar – Black Box

Pengujian *black box* adalah pengujian aspek fundamental sistem tanpa memperhatikan struktur logika internal perangkat lunak. Metode ini digunakan untuk mengetahui apakah perangkat lunak berfungsi dengan benar. Pengujian black box merupakan metode perancangan data uji yang didasarkan pada spesifikasi perangkat lunak. Data uji dibangkitkan, dieksekusi pada perangkat lunak dan kemudian keluaran dari perangkat lunak dicek apakah telah sesuai dengan yang diharapkan.

Pengujian black box berusaha menemukan kesalahan dalam kategori :

1. fungsi-fungsi yang tidak benar atau hilang
2. kesalahan interface
3. kesalahan dalam struktur data atau akses database eksternal
4. kesalahan kinerja
5. inisialisasi dan kesalahan terminasi.

Berbeda dengan pengujian white box, pengujian black box cenderung diaplikasikan selama tahap akhir pengujian. Pengujian black box harus dapat menjawab pertanyaan sebagai berikut :




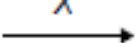
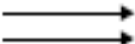
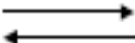
- a) Bagaimana validitas fungsional diuji
- b) Kelas input apa yang akan membuat kasus pengujian menjadi lebih baik
- c) Apakah system akan sangat sensitive terhadap harga input tertentu
- d) Bagaimana batasan dari suatu data diisolasi
- e) Kecepatan data apa dan volume data apa yang akan ditoleransi oleh system
- f) Apa pengaruh kombinasi tertentu dari data terhadap system operasi.

2.1 Graf Based Testing

Langkah pertama pada pengujian black box adalah memahami objek yang terdapat dalam model perangkat lunak dan menentukan hubungan yang dimiliki antara objek-objek tersebut. Pengujian berbasis model graf dilakukan terhadap perilaku system.

Graf Based Testing menggambarkan graf yang mewakili hubungan antar objek pada modul sehingga tiap objek dan hubungannya dapat diuji. Pengujian ini dimulai dari mendefinisikan semua simpul dan bobot simpul, dimana objek dan atribut diidentifikasi, serta memberikan indikasi titik mulai dan berhenti.

Tabel – Notasi Pengujian berbasis graf

NOTASI	ARTI
	Simpul atau node Menggambarkan suatu objek
	Link Menggambarkan hubungan antar objek
	Node weight Menggambarkan properti atau nilai dari data
	Link weight Menggambarkan karakteristik link
	Link paralel Menggambarkan hubungan yang berbeda yang dibangun antar simpul
	Link simetris Menggambarkan hubungan dua arah antara dua objek

Terdapat tiga pola link weight, yaitu :

- *Transitivitas*, yaitu hubungan antara tiga objek atau lebih yang menentukan bagaimana pengaruh hubungan tersebut menyebar pada objek yang ditentukan
- *Simetris*, yaitu hubungan antara dua objek secara dua arah
- *Refleksif*, yaitu hubungan yang mengarah pada node itu sendiri atau *loop null*

Beizer menyebutkan beberapa metode pengujian black box yang menggunakan graf, yaitu :

1. Transaction Flow Modeling, metode ini menggunakan node sebagai representasi langkah pada transaksi, dan link sebagai representasi hubungan logika antara langkah-langkah tersebut
2. Finite state modeling, metode ini menggunakan node sebagai representasi status dan link sebagai representasi transisi. Statechart atau state transition diagram dapat digunakan untuk membuat graf.
3. Data flow modeling, metode ini menggunakan node sebagai representasi objek data dan link sebagai transformasi dari satu objek data ke objek data yang lain.
4. Timing modeling, metode ini menggunakan node sebagai representasi objek program dan link sebagai hubungan sekuensial antara objek.

2.2 Equivalence Partitioning (Partisi ekuivalensi)

Partisi ekuivalensi adalah metode yang membagi domain input dari suatu program ke dalam kelas data, menentukan kasus pengujian dengan mengungkapkan kelas-kelas kesalahan, sehingga akan mengurangi jumlah keseluruhan kasus pengujian.

Bila suatu *link weight* mempunyai pola *transitivitas*, *simetris*, dan *refleksif* maka akan terdapat kelas ekuivalensi. Kelas ekuivalensi merepresentasikan serangkaian kondisi valid dan invalid untuk kondisi inputan. Secara khusus, suatu kondisi input dapat berupa harga numeric, suatu rentang harga, serangkaian harga yang terkait, atau suatu kondisi Boolean.

Penentuan Kelas Ekuivalensi

1. Bila kondisi input menentukan suatu range, maka satu kelas ekuivalensi valid dan dua yang invalid ditentukan
2. Bila suatu kondisi input memerlukan suatu harga khusus, maka satu kelas ekuivalensi valid dan dua yang invalid ditentukan
3. Bila suatu kondisi menentukan anggota suatu himpunan, maka satu kelas ekuivalensi valid atau dua yang invalid ditentukan
4. Bila suatu kondisi input adalah Boolean, maka satu kelas valid dan satu yang lain ditentukan.

Contoh :

Nomor telepon untuk SLJJ

Ketentuan :

Kode area : kosong atau ada 3 – 4 digit yang dimulai dengan 0
 Prefiks : 3 digit dengan tidak dimulai dengan 1 atau 0
 Sufiks : 4 digit

Kondisi input :

Kode area	: kondisi input	→ Boolean – boleh ada boleh tidak ada
	kondisi input	→ range – nilai >200 atau nilai < 999
Prefiks	: kondisi input	→ range – nilai > 200 atau nilai < 999
Sufiks	: kondisi input	→ harga – 4 digit

2.3 Boundary Value Analysis (Analisis Nilai Batas)

Analisis nilai batas adalah teknik desain proses yang melengkapi partisi ekuivalensi, dengan berfokus pada domain output.

Pedoman untuk menentukan analisis nilai batas :

1. Bila suatu kondisi input mengkhususkan suatu range dibatasi oleh nilai a dan b, maka pengujian harus didesain dengan nilai a dan b, persis di atas dan di bawah a dan b secara bersesuaian
2. Bila suatu kondisi input mengkhususkan sejumlah nilai, maka pengujian harus dikembangkan dengan menggunakan jumlah minimum dan maksimum. Nilai tepat di atas dan di bawah minimum dan maksimum juga diuji.
3. Pedoman 1 dan 2 juga diaplikasikan ke kondisi output.
4. Bila struktur data program telah memesan suatu batasan, maka pengujian akan dilakukan sesuai dengan batasan struktur data tersebut.

2.4 Comparison Testing

Pengujian perbandingan adalah metode pembangkitan data uji yang dilakukan pada perangkat lunak yang dibuat redundan. Perangkat lunak yang redundan mempunyai dua tim pengembang yang masing-masing mengembangkan perangkat lunak sendiri-sendiri untuk spesifikasi yang sama.

Metode pengujian perbandingan digunakan untuk membangkitkan data uji untuk salah satu perangkat lunak, yang kemudian digunakan sebagai masukan pada pengujian perangkat lunak yang lain.

Comparison Testing digunakan untuk system yang menganut redundancy, kasus uji yang dirancang untuk satu versi perangkat lunak dijadikan masukan pada pengujian versi perangkat lunak lainnya, dan diharapkan hasil kedua versi perangkat lunak harus sama. Jika hasil pengujian kedua perangkat lunak tersebut berbeda maka kedua perangkat lunak itu akan dicek untuk mencari yang salah.