

BAB II

LANDASAN TEORI

2.1 Tinjauan Pustaka

Pengujian merupakan hal yang tidak terpisahkan dari sebuah perangkat lunak. Pentingnya pengujian perangkat lunak mengacu pada kualitas dari perangkat lunak itu sendiri. Pengujian dengan *white box testing* masih sangat jarang ditemui, namun beberapa peneliti sudah melakukan penelitian pengujian dengan metode *white box testing*. Metode pengujian *white box* menekankan pengujian berdasarkan pada *source code* program yang dibuat. Terdapat beberapa pendekatan yang dilakukan dalam metode pengujian *white box*, dengan berbagai macam metode atau teknik yang dilakukan dalam penelitian sebelumnya. Hal tersebut menjadi sebuah dasar atau petunjuk untuk penulis dalam melakukan sebuah penelitian untuk pengujian perangkat lunak dengan menggunakan metode *white box testing*.

Beberapa penelitian yang membahas tentang pengujian perangkat lunak adalah pengujian pada aplikasi inventory Taufan Fish Farm yang menggunakan metode *white box testing* dan *black box testing*. Teknik yang dipakai dalam pengujian *white box testing* yaitu *statement coverage* dan *branch coverage*. Pengujian dilakukan dengan mengambil sampel dari salah satu form di dalam aplikasi Taufan Fish Farm yaitu *login user* yang digunakan untuk memeriksa sebuah masukan *username* dan *password* dari pengguna. Kemudian membuat sebuah skenario uji yang akan menjadi acuan pada proses pengujian (Kurniansyah, Sempati, & Hidayat, 2016). Tabel 2.1 menunjukkan skenario *test form login*.

Tabel 2.1 Skenario Uji *Form Login*

No	Username	Password	Hasil
Login 1	Toko	Ikan	Gagal, muncul pesan
Login 2	Ikan	Toko	Gagal, muncul pesan
Login 3	Ikan	Hias	Gagal, muncul pesan
Login 4	TOKO	IKANHIAS	Gagal, muncul pesan

Pada pengujian ini dilakukan dengan memasukkan *username* dan *password* secara berturut-turut sesuai dengan skenario ujicoba dengan memberikan kondisi dimana *username* dan *password* yang benar adalah “TOKO dan IKANHIAS”. Dan setelah memasukkan *username* dan *password* yang berbeda dari yang sudah ditentukan maka akan muncul suatu pesan kesalahan jika hasilnya adalah gagal karna tidak sesuai. Sedangkan proses pengujian

pada *branch coverage* yang dilakukan yaitu merubah *statement* ke dalam bentuk *flowchart*. Pengujian dilakukan dengan mengecek pada setiap kondisi kemungkinan *true* dan *false* seperti memberikan masukan *username* dan *password* yang berbeda dari “TOKO dan IKANHIAS” maka hasilnya akan muncul pesan *error* dan dikembalikan pada kondisi mengisi *username* dan *password* ulang. Dengan kata lain yang membedakan dari teknik *statement coverage*, yaitu pada *branch coverage* menjalankan semua *statement* yang tidak dilalui pada pengujian *statement* dan memastikan bahwa setiap cabang dari masing-masing titik keputusan dijalankan. Dimana pada penelitian ini yaitu dijelaskan tentang apa itu teknik pengujian *statement coverage* dan *branch coverage*. Kita dapat menjelaskan bahwa sebuah *statement coverage* tidak lain merupakan suatu teknik yang dilakukan untuk menguji setiap baris kode program dengan menjalankan pengujian berdasarkan data uji yang sudah ditetapkan. Sedangkan *branch coverage* yaitu suatu teknik pengujian yang digunakan untuk memvalidasi setiap *statement if* yang ada pada baris kode program (Kurniansyah, Sempati, & Hidayat, 2016).

Selanjutnya penelitian pada Aplikasi Pencarian Rute Angkutan Umum di Bandar Lampung berbasis mobile android dengan pengujian *white box testing*. Pengujian dengan *white box testing* dilakukan untuk mendapatkan sebuah *test case* dalam menguji sebuah algoritma pada kode program, adapun metode yang digunakan dalam pengujian ini yaitu metode *basis path* untuk menguji sebuah algoritma djikstra. Pengujian dilakukan dengan melakukan perhitungan tingkat akurasi algoritma yang digunakan dalam pencarian rute angkutan kota dan bus rapid transit (BRT). Prosedur pengujian pada aplikasi tersebut yaitu dengan memasukkan node awal dan node tujuan, kemudian sistem menampilkan hasil pencarian rute terpendek. Dari hasil pencarian pada sistem tersebut maka akan di cocokkan kembali kesesuaiannya dengan hasil yang di harapkan yang diperoleh dari perhitungan manual. Pengujian pada graph trayek angkot dilakukan dengan 15 data uji yang disajikan, kemudian pengujian pada graph trayek BRT dilakukan juga dengan 15 data uji. Dari hasil pengujian yang telah dilakukan pada 30 data uji tersebut di dapatkan hasil dengan akurasi sebesar 100% dari hasil perhitungan manual maupun dengan pengujian dengan pengujian menggunakan metode *white box testing* (Ariyandi, Kurniawan, & Hijriani, 2016).

Selanjutnya penelitian pada Perancangan Aplikasi Perangkat Lunak Gamelan Virtual Berbasis Android. Pada aplikasi tersebut digunakan pengujian dengan metode *white box testing* dan *black box testing*. Teknik yang digunakan dalam pengujian *white box testing* adalah metode *basis path*. Pengujian dengan *basis path* dilakukan untuk memungkinkan desainer *test case* dalam mengukur kompleksitas logis dari desain procedural dan menggunakannya sebagai pedoman untuk menetapkan basis set dari setiap jalur eksekusi. Dari pengujian didapatkan

hasil pengujiannya yaitu semua jalur program terlewati, dimana setiap jalur yang di eksekusi dijalankan setidaknya satu kali (Harrydhy D, 2013).

Penelitian selanjutnya yaitu untuk mengetahui kelayakan dari aplikasi yang dibangun pada aplikasi e-commerce pada Indo Mandiri Computer Semarang. Pada aplikasi tersebut digunakan metode *white box testing* dan *black box testing* dalam tahap pengujian. Teknik yang digunakan dalam pengujian *white box testing* yaitu *basis path*. Pengujian dilakukan untuk mengukur kompleksitas logis dari perancangan procedural pada aplikasi. Berdasarkan hasil pengujian dari hasil kompleksitas siklomatis yang didapatkan, memperoleh bahwa sistem dikatakan baik (Darmeswara, 2013). Adapun perbandingan dari penelitian-penelitian tersebut dapat dilihat pada Tabel 2.2.

Tabel 2.2 Perbandingan Penelitian

Makalah	Teknik Pengujian	Cara Pengujian
(Kurniansyah, Sempati, & Hidayat, 2016)	Statement Coverage Testing dan Branch Coverage Testing	Melakukan sebuah scenario ujicoba pada sampel form login untuk mengetahui kelayakan jenis input yang dihasilkan dan mengetahui sensitifitas sistem terhadap input.
(Ariyandi, Kurniawan, & Hijriani, 2016)	Basis Path Testing	Melakukan perhitungan tingkat akurasi yang dihasilkan dengan 30 data uji yang diperoleh dari setiap jalur yang dieksekusi pada graph trayek yang dibuat menggunakan algoritma djikstra.
(Harrydhy D, 2013)	Basis Path Testing	Membuat gambaran alur logika dengan notasi-notasi yang telah ditentukan, kemudian menentukan nilai siklomatis kompleksitas untuk menentukan basis set jalur , dan kemudian melakukan eksekusi dari tiap jalur yang didapat dengan menjalankan setidaknya satu kali.
(Darmeswara, 2013)	Basis Path Testing	Membuat diagram alir untuk membantu dalam menentukan nilai siklomatis kompleksitas, kemudian setelah nilai kompleksitas didapat maka selanjutnya membuat basis set dengan menentukan jalur yang akan dieksekusi.

Pengujian yang dilakukan (Kurniansyah, Sempati, & Hidayat, 2016) menggunakan teknik *statement coverage testing* dan *branch coverage testing* pada metode pengujian *white box testing*. Sedangkan (Ariyandi, Kurniawan, & Hijriani, 2016), (Harrydhy D, 2013), dan (Darmeswara, 2013) menggunakan teknik *basis path* yang digunakan untuk mengukur kompleksitas logis dari desain procedural yang kemudian menetapkan basis set untuk jalur

eksekusi. Berdasarkan penelitian-penelitian tersebut, maka dapat diketahui bagaimana gambaran awal dalam melakukan sebuah pengujian perangkat lunak menggunakan metode *white box testing*. Pada penelitian-penelitian sebelumnya pengujian dengan *white box* hanya dilakukan dengan menentukan setiap jalur proses pada program, tidak memastikan apakah setiap jalur yang dieksekusi tersebut sesuai atau tidak dengan kerja pada program itu sendiri. Di samping itu pengujian dengan *white box testing* masih jarang ditemui karena pada umumnya banyak pengembang sistem hanya melakukan pengujian berdasarkan pada tampilan pada sistem atau yang biasa disebut dengan metode *black box testing*. Dan belum adanya penelitian yang melakukan perbandingan pada kedua teknik tersebut.

Pada penelitian ini akan dilakukan pengujian pada sebuah sistem informasi yang masih dalam tahap pengembangan yaitu sistem informasi berbasis website (Sistem Informasi Monitoring Community TB-HIV Care ‘Aisyiyah Tanggamus). Pengujian sistem dilakukan dengan menggunakan 2 teknik dari metode *white box testing* yaitu *statement coverage* dan *branch coverage*, yang bertujuan memastikan setiap statament dan kondisi cabang yang ada pada *source code* perangkat lunak di eksekusi dengan tepat. Dengan menggunakan 2 teknik tersebut diharapkan alur logika pada program dapat berjalan sesuai dengan fungsionalitasnya dan mengurangi segala kesalahan yang ada pada program sehingga sistem dapat berjalan sesuai dengan yang diharapkan dan dapat digunakan dengan baik oleh pengguna.

2.2 Community TB-HIV Care ‘Aisyiyah

‘Aisyiyah adalah organisasi otonom khusus Muhammadiyah yang didirikan di Indonesia pada tahun 1917 dengan kepedulian terhadap isu-isu sosial keagamaan. Sejak 2010 ‘Aisyiyah dipilih oleh *Global Fund for AIDS, Tuberculosis* dan Malaria (GF-ATM) untuk melaksanakan program “*Community TB-HIV Care*” sebagai *Principal Recipient* (PR). Proyek *Global Fund* tersebut merupakan upaya memerangi penyakit *Tuberculosis* yang difokuskan pada kegiatan berbasis masyarakat dan dikelola oleh Majelis Kesehatan Pimpinan Pusat ‘Aisyiyah dan berkoordinasi dengan PR Kementerian Kesehatan RI untuk periode 2016-2017 (Round N-FM).

Program ini di bentuk sebagai upaya untuk meningkatkan peran serta aktif masyarakat dalam menyuluh, menemukan penderita TB-HIV dan mengawal pengobatan pasien hingga sembuh. Di samping itu program ini termasuk mensosialisasikan hak dan kewajiban penderita TB-HIV (*Patient Charter*) serta upaya meningkatkan peran serta aktif perempuan dalam menanggulangi TB-HIV di Indonesia. Sebagai *Principal Recipient* (PR), ‘Aisyiyah bertanggung jawab mengkoordinasikan kegiatan di tingkat masyarakat dari 18 Sub Recipient (SR) yang mencakup 16 Pimpinan Wilayah ‘Aisyiyah dan 8 LSM lokal (YARSI, PKPU, KMP

TB Sidobinangun Lampung Tengah, LKC, Pelkesi, Peridhaki, PETA). Dan pada tahun 2016 yang lalu GF-ATM Aisyiyah sudah bekerjasama dengan Sub Recipient (SR) yang tersebar di 25 provinsi di Indonesia dan Sub-sub Recipient (SSR) yang tersebar di 160 Kabupten/Kota.

2.3 Pengujian Perangkat Lunak

Pengujian perangkat lunak dilakukan untuk mengetahui apakah pada sebuah program atau sistem tersebut sudah sesuai dengan hasil yang di harapkan. Pengujian merupakan bagian yang tidak terpisahkan dari sebuah perangkat lunak. Dengan berjalannya waktu sekarang ini banyak sistem atau program yang dibangun dengan tujuan memudahkan aktifitas yang berjalan pada sebuah instansi ataupun organisasi, sehingga perlu adanya peningkatan yaitu dengan melakukan pengujian pada sebuah perangkat lunak agar aplikasi atau sistem dapat berjalan dengan baik ataupun fitur-fitur yang ada pada sistem bisa digunakan dengan baik. Pentingnya pengujian perangkat lunak dan implikasinya mengacu pada kualitas perangkat lunak. Perangkat lunak adalah elemen kritis dari jaminan kualitas perangkat lunak dan merepresentasikan kajian pokok dari spesifikasi, desain, dan pengkodean. Meningkatnya visibilitas perangkat lunak sebagai suatu elemen sistem dan biaya yang muncul akibat kegagalan perangkat lunak, memotivasi dilakukannya perencanaan yang baik melalui pengujian yang teliti.

Desain pengujian perangkat lunak dan produk rekayasa lain dapat sama-sama menantang dengan desain awal dari produk itu sendiri. Berdasarkan pada obyektifitas *testing*, pentingnya melakukan desain test untuk tujuan menemukan kesalahan (*error*) yang sering terjadi, dengan usaha dan waktu yang minimum (B, 2006). Saat ini berbagai macam metode desain *test case* telah berkembang, yang digunakan dalam pengujian perangkat lunak. Metode-metode tersebut memberikan kepada pengembang perangkat lunak sebuah pendekatan yang sistematis untuk melakukan pengujian. Dan terlebih penting lagi, metode-metode ini menyediakan mekanisme yang dapat membantu untuk memastikan kelengkapan dari testing dan menyediakan kemungkinan tertinggi untuk mendapatkan *error* pada perangkat lunak (Jatnika & Irwan, 2010).

Terdapat beberapa pendekatan pengujian yang dilakukan pada perangkat lunak, antara lain:

- a. Dengan berdasarkan pada fungsi yang dispesifikasikan dari produk, pengujian dilakukan untuk memperlihatkan bahwa masing-masing fungsi sudah beroperasi sepenuhnya, pada waktu yang sama mencari kesalahan pada setiap fungsi (B, 2006).
- b. Dengan berdasarkan pada suatu kinerja internal dari produk, pengujian dilakukan dengan memastikan semua komponen pada program dapat berjalan dengan baik sebagaimana mestinya (Jatnika & Irwan, 2010).

Pendekatan pengujian cara pertama biasa disebut dengan *black box testing* dan pendekatan cara kedua biasa disebut dengan *white box testing*.

2.4 White Box Testing

Pengujian *white box testing* biasa disebut dengan *glass box*, adalah metode desain *test case* yang menggunakan struktur kontrol desain prosedural untuk memperoleh sebuah *test case* (B, 2006). Tujuan dari penggunaan pengujian *white box* yaitu menguji semua *statement* pada program. Metode pengujian *white box* dapat menjamin:

- a. Semua jalur (*path*) yang independen / terpisah dapat di tes setidaknya satu kali tes.
- b. Semua logika keputusan dapat dites dengan jalur yang salah dan atau jalur yang benar.
- c. Semua loop dapat dites terhadap batasannya dan ikatan operasionalnya.
- d. Semua struktur internal data dapat dites untuk memastikan validitasnya.

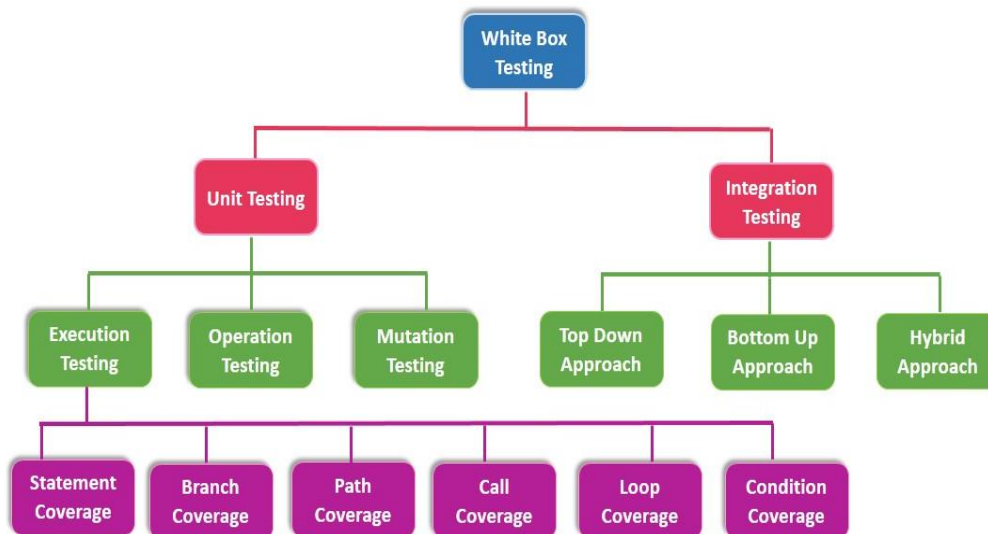
Metode pengujian *white box* seringkali diasosiasikan sebagai suatu teknik dalam pengukuran cakupan tes (*test coverage metrics*), yaitu sehubungan dengan menghitung persentase dari jalur-jalur yang dieksekusi berdasarkan test case yang dipilih (Jatnika & Irwan, 2010). Pada dasarnya sekarang ini, banyak para penguji perangkat lunak jarang melakukan pengujian dengan menggunakan *white box testing*, tetapi langsung melakukan pengujian pada tampilan antarmuka sistem. Namun demikian pengujian tidak bisa hanya dilakukan untuk menguji pada tampilan sistem, tetapi juga perlu dilakukan pengujian pada struktur dan kontrol logika pada kode program. Pengujian dengan metode *white box testing* dilakukan pada suatu perangkat lunak dengan tujuan untuk menemukan kesalahan-kesalahan yang mungkin tidak ditemukan ketika melakukan pengujian pada tampilan antarmuka sistem atau yang biasa disebut dengan metode *black box testing*.

Adapun penjelasan mengenai kesalahan-kesalahan yang mungkin dapat ditemukan seperti:

- a. Kesalahan logika dan asumsi yang tidak benar kebanyakan dilakukan ketika *coding* untuk “kasus tertentu”. Dibutuhkan kepastian bahwa eksekusi jalur ini telah dites (Jatnika & Irwan, 2010).
- b. Asumsi bahwa adanya kemungkinan terhadap eksekusi jalur yang tidak benar. Dengan *white box testing* dapat ditemukan kesalahan ini (Jatnika & Irwan, 2010).
- c. Kesalahan penulisan yang acak, seperti berada pada jalur logika yang membingungkan pada jalur normal (Jatnika & Irwan, 2010).

Terdapat beberapa teknik dalam pengujian menggunakan metode *white box testing*. Berikut ini adalah teknik-teknik yang terdapat pada *white box testing* yang dapat dilihat pada Gambar 2.1.

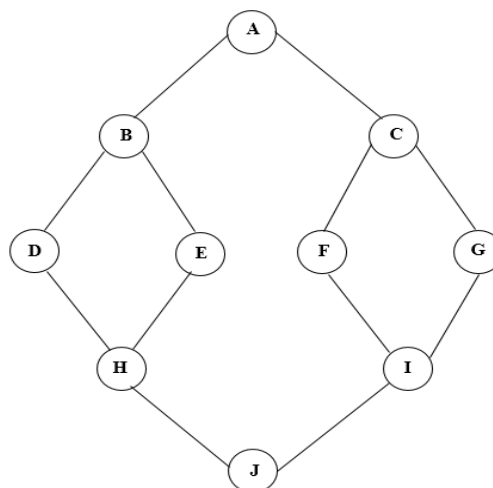
Types of White Box Testing



Gambar 2.1 Teknik-teknik pengujian *white box testing*

2.4.1 *Statement Coverage Testing*

Statement dalam bahasa pemrograman tidak lain adalah sebuah baris kode. *Statement coverage testing* merupakan teknik pengujian *white box* yang memastikan setiap *executable statement* dijalankan setidaknya satu kali. Satu *statement* yang dijalankan hanya merupakan bagian dari satu *test case*, sehingga tidak mungkin ada nya *test case* yang menjalankan *statement* sama. Dengan demikian hal tersebut membuktikan bahwa pada *statement coverage* memastikan semua pernyataan dijalankan tanpa efek samping. Sebagai ilustrasi terdapat contoh *flowgraph* yang menggambarkan pengujian dengan teknik *statement coverage* dapat dilihat pada Gambar 2.2.



Gambar 2.2 Contoh *Flowgraph*

Pada contoh *flowgraph* diatas terdapat 10 titik, sebagai contoh suatu jalur eksekusi program melewati titik-titik a,b,d,h,k. Maka jumlah titik yang terlewati adalah 5 dari seluruh jumlah titik yang ada yaitu 10. Sehingga hasil dari satu test case yang dilakukan tersebut memperoleh nilai *statement* sebesar 50% (Jatnika & Irwan, 2010). Adapun rumus untuk menghitung nilai *coverage* pada teknik *statement coverage testing* adalah sebagai berikut:

$$\text{Statement coverage} = \frac{\text{Number of Statements Exercised}}{\text{Total Number of Statements}} \quad (2.1)$$





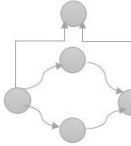
2.4.2 Branch Coverage Testing

Branch dalam bahasa pemrograman seperti “*Statement IF*”. Dimana *statement* memiliki dua cabang yaitu *true* dan *false*. *Branch coverage testing* merupakan teknik pengujian *white box* yang memastikan bahwa setiap cabang di eksekusi minimal satu kali. Pada *branch coverage* memastikan bahwa setiap cabang dari masing-masing titik keputusan telah dijalankan dengan memvalidasi cabang yang benar ataupun cabang yang salah. Berdasarkan pada *flowgraph* yang dijabarkan sebelumnya, terdapat 6 anak panah cabang. Sebagai contoh, suatu jalur eksekusi program melewati titik-titik a,b,d,h,k, jalur tersebut meninjau 2 dari 6 anak cabang yang ada, sehingga nilai *branch* yang didapat yaitu 33% (Jatnika & Irwan, 2010). Adapun rumus untuk menghitung nilai *coverage* pada teknik *branch coverage testing* adalah sebagai berikut:

$$\text{Branch coverage} = \frac{\text{Number of Branch Exercised}}{\text{Total Number of Branch}} \quad (2.2)$$

2.5 Notasi Diagram Alir (*Path Graph Notation*)

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. Setiap representasi rancangan procedural dapat diterjemahkan kedalam *flowgraph*.

Notasi	Arti
	Skema Sequence
	Skema If
	Skema White (...) DO (...)
	Skema Repeat (...) Until (...)
	Skema Case (...) Of

Gambar 2.3 Gambar Notasi Diagram Alir

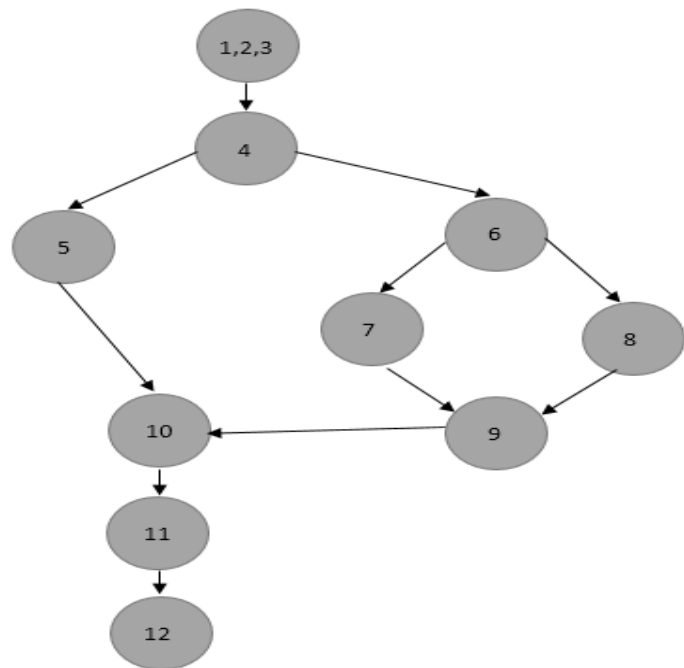
Berikut Gambar 2.4 menunjukkan contoh bagian PDL (Program Design Language) dan flow graph.

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 2.4 Contoh Notasi Diagram Alir

2.6 Kompleksitas Siklomatis (*Cyclomatic Complexity*)

Cyclomatic Complexity adalah metode pengukuran perangkat lunak yang memberikan pengukuran kuantitatif terhadap kompleksitas logika sebuah program. Dalam mengukur sebuah kompleksitas pada sebuah program maka dilakukan perhitungan *Cyclomatic Complexity* (cc), nilai yang didapat berdasarkan perhitungan tersebut untuk menentukan

jumlah jalur independent dalam himpunan path, serta akan memberi nilai batas atas bagi jumlah pengujian yang harus dilakukan, untuk memastikan bahwa semua pernyataan telah dieksekusi setidaknya satu kali. jalur independen adalah jalur yang terdapat dalam program yang mengintroduksi sedikitnya satu rangkaian pernyataan proses atau kondisi baru. *Cyclomatic Complexity* dapat dicari dengan salah satu dari 3 cara sebagai berikut:

1. Jumlah region dari grafik alur mengacu kepada kompleksitas siklomatis.
2. *Cyclomatic complexity* dilambangkan dengan $V(G)$. Rumus penghitungan cc adalah:

$$V(G) = E - N + 2 \quad (2.3)$$

Dimana:

E = jumlah *edge* pada *flowgraph*

N = jumlah *node* pada *flowgraph*

3. *Cyclomatic complexity* dilambangkan dengan $V(G)$. Rumus penghitungan cc adalah:

$$V(G) = P + 1 \quad (2.4)$$

Dimana:

P = jumlah *predicates nodes* pada *flowgraph*

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

Berdasarkan contoh PDL sebelumnya, maka *Cyclomatic Complexity* dapat dihitung sebagai berikut:

$$\begin{aligned} V(G) &= E - N + 2 \\ &= 13 - 12 + 2 \\ &= 3 \end{aligned}$$

Hasil dari *Cyclomatic Complexity* menggambarkan banyaknya path dan batas atas sejumlah ujicoba yang harus dirancang dan dieksekusi untuk seluruh perintah dalam program.

Berdasarkan perhitungan cc sebelumnya, maka jalur independent yang diperoleh:

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

2.7 Test Case

Test case adalah dokumen yang memiliki satu set data tes, prasyarat, hasil yang diharapkan dan postconditions, dikembangkan untuk skenario tes tertentu untuk memverifikasi kepatuhan terhadap persyaratan tertentu (Suhartono, 2016). *Test case* merupakan titik awal untuk melakukan tes dan setelah menerapkan nilai-nilai input ke sistem, akan didapat hasil yang definitif dan meninggalkan sistem di beberapa titik akhir atau juga dikenal sebagai postcondition eksekusi.

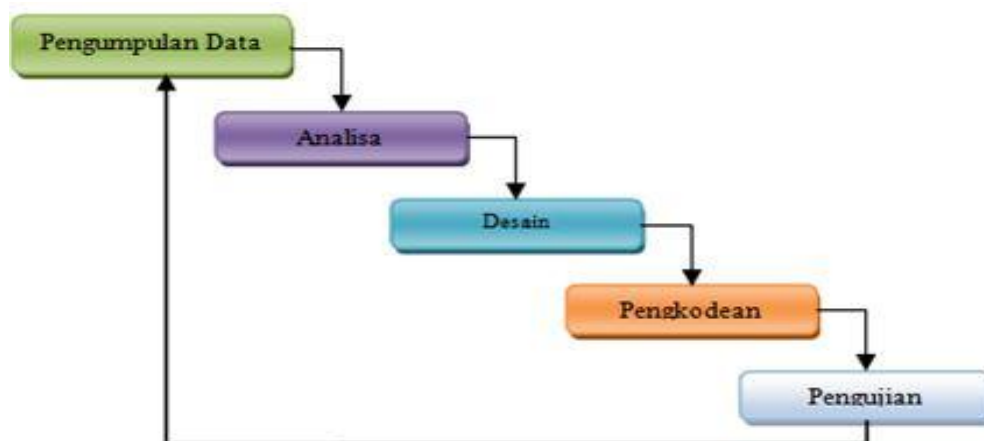
Adapun model *test case* yang biasa digunakan pada suatu pengujian, dapat dilihat pada Tabel 2.3.

Tabel 2.3 Model *Test Case*

Test Case Id	Test Case Name	Test Case Descriptiv	Test Steps			Test Case Status
			Step	Expect	Actual	

2.8 Metode SDLC (Systems Development Life Cycle)

SDLC (*Systems Development Life Cycle*) atau *Systems Life Cycle* (Siklus Hidup Sistem), dalam rekayasa sistem dan rekayasa perangkat lunak, merupakan proses pembuatan dan perubahan sistem serta model dan metodologi yang digunakan dalam mengembangkan sebuah sistem informasi (Joulisinolungan, 2014). Dalam rekayasa perangkat lunak, konsep SDLC mendasari berbagai jenis metodologi pengembangan perangkat lunak yang salah satunya yaitu metode *waterfall* atau biasa lebih dikenal dengan istilah siklus kehidupan klasik. Dimana sebuah sistem dibangun sesuai dengan tahap-tahap yang dirancang dengan baik dan teratur. Adapun tahapan yang dilakukan dalam SDLC *waterfall* dapat dilihat pada Gambar 2.5.



Gambar 2.5 Metode *Waterfall*

- a) Pengumpulan data: Pengumpulan data merupakan usaha yang dilakukan untuk memperoleh informasi dalam bentuk data yang dibutuhkan dalam penelitian. Metode pengumpulan data dapat dilakukan dengan wawancara, observasi, dan dokumentasi.
- b) Tahap Analisis: Tahap berikutnya setelah data terkumpul adalah tahap analisis kebutuhan sistem. Kegiatan ini diintensifkan dan difokuskan pada sistem, yaitu menganalisa kebutuhan dan persyaratan proses pada sistem yang akan dibangun (Agung, 2016).
- c) Tahap Perancangan: Tahapan ini bertujuan untuk membuat rancangan dari hasil analisa yang telah dilakukan pada tahap sebelumnya. Perancangan yang akan dilakukan dalam pengembangan sistem ini meliputi, perancangan struktur data, perancangan proses, perancangan antarmuka (Agung, 2016).
- d) Tahap Pengkodean: Tahapan implementasi merupakan tahapan mengubah rancangan yang telah dibuat menjadi kumpulan kode atau instruksi yang akan dijalankan oleh komputer.
- e) Tahap Pengujian: Pengujian sistem dilakukan untuk mengukur kelayakan dan kesesuaian sistem yang dibangun. Semua fungsi harus diuji supaya bebas dari error dan dapat berjalan sebagaimana yang diharapkan (Agung, 2016).