BAB II

TEORI PENUNJANG

Inti dari pengujian perangkat lunak dengan Teknik Pengujian Basis Path adalah penentuan lintasan-lintasan eksekusi perangkat lunak untuk penyusunan test case. Oleh karena itu, pada Bab II ini akan diberikan pembahasan beberapa konsep, antara lain : flowgraph, yang digunakan untuk menggambarkan aliran kontrol perangkat lunak, dan algoritma pencarian Depth First Search yang akan digunakan untuk mencari satu set lintasan dari flowgraph serta dasar-dasar rekayasa perangkat lunak.

2.1. Pengujian Perangkat Lunak

Sebelum membahas pengujian perangkat lunak, berikut ini akan dibahas siklus hidup perangkat lunak. Dari siklus hidup perangkat lunak tersebut dapat diketahui dimana kedudukan pengujian pada pengembangan sistem perangkat lunak.

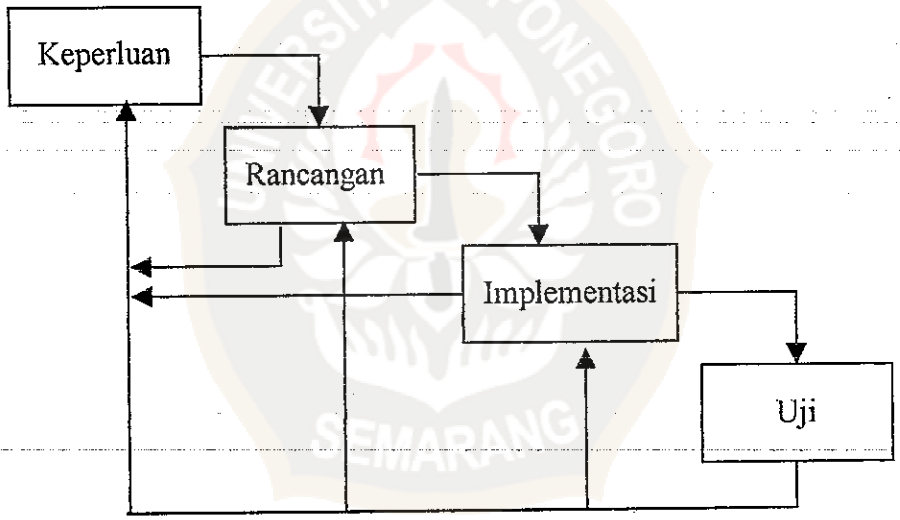
2.1.1. Siklus Hidup Perangkat Lunak (*Software Life Cycle*)

Pengembangan sistem perangkat lunak memerlukan beberapa tahapan yang berbeda. Kemudian di dalam penggunaan perangkat lunak, terdapat tahapan lain yang terkait. Secara bersama-sama tahapan-tahapan tersebut membentuk suatu siklus hidup perangkat lunak.

Menurut Royce yang dikutip oleh Sommerville (Sommerville,1992) model dari proses pengembangan perangkat lunak diturunkan dari aktifitas rekayasa

yang lain. Karena proses pengembangan perangkat lunak mengalir dari satu fase ke fase yang lain maka model ini dikenal sebagai model "*waterfall*" .

Model Waterfall terdiri dari fase-fase antara lain : definisi keperluan, rancangan, implementasi, dan uji. Tahap-tahap tersebut berjalan berturutan dan terkait satu arah. Tapi pada kenyataannya, fase-fase pengembangan perangkat lunak bukanlah model linier yang sederhana tetapi merupakan sebuah barisan iterasi yang melibatkan aktifitas-aktifitas pengembangan (lihat gambar 2.1).



Gambar 2.1
Siklus Pengembangan Perangkat Lunak

Penjelasan dari masing-masing tahapan adalah sebagai berikut :

a. Analisis Keperluan dan Definisi. Berisi kegiatan analisa, memahami dan mencatat masalah yang ingin dipecahkan serta merumuskan fungsi, tujuan, kendala-kendala dan batasan sistem. Perumusan yang dihasilkan perlu mendapatkan persetujuan bersama antara sponsor, pemakai dan

pengembang. Sekali disetujui harus didefinisikan di dalam bentuk yang dimengerti oleh semua pihak.

b. Rancangan Sistem dan Perangkat Lunak. Keperluan diperinci menjadi sistem perangkat keras dan sistem perangkat lunak. Proses ini disebut rancangan sistem. Rancangan perangkat lunak adalah proses menyajikan fungsi dari setiap sistem perangkat lunak dalam bentuk yang dapat diubah menjadi satu atau lebih program komputer.

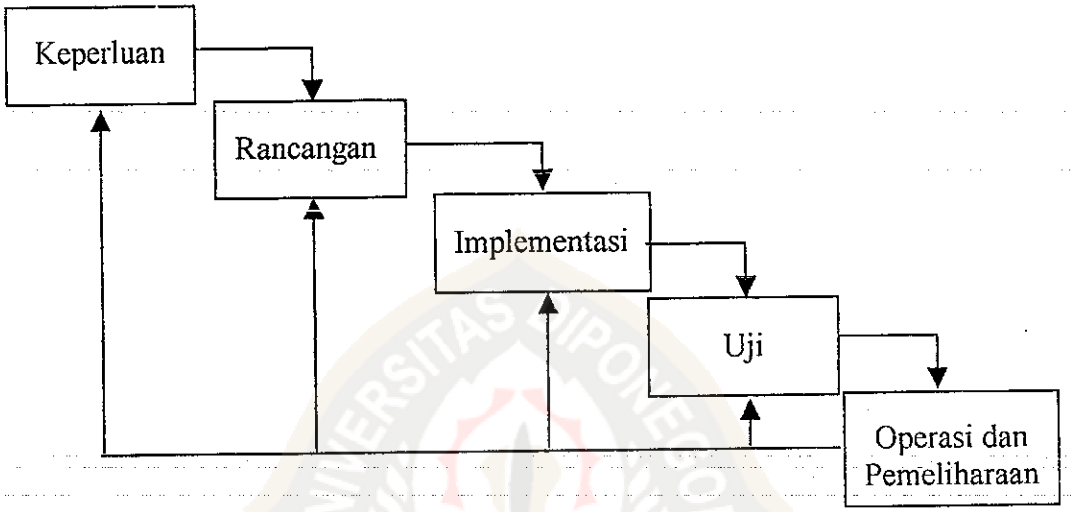
c. Implementasi dan Uji Unit. Rancangan perangkat lunak diwujudkan sebagai sekumpulan program atau unit program yang ditulis dalam bentuk bahasa pemrograman yang dapat dieksekusi. Uji unit meliputi pemeriksaan, agar setiap unit program sesuai dengan spesifikasinya.

d. Uji Sistem. Semua unit program perlu dipadukan, dan diperiksa sebagai sistem yang lengkap untuk memastikan kebutuhan perangkat lunak telah dapat dipenuhi.

e. Operasi dan Pemeliharaan. Sistem di-install untuk penggunaan praktis. Kegiatan pemeliharaan meliputi koreksi kesalahan yang tidak ditemukan pada tahapan siklus hidup sebelumnya, peningkatan implementasi unit sistem, dan meningkatkan layanan sistem.

Selama fase terakhir siklus hidup (operasi dan pemeliharaan) siklus mengarah ke arah fase pengembangan sebelumnya (lihat gambar 2.2). Pada saat sistem dioperasikan kemungkinan akan ditemukan kesalahan atau kekurangan pada sistem tersebut. Oleh karena itu, modifikasi diperlukan pada perangkat lunak

untuk memenuhi kebutuhan sistem yang terus berkembang. Modifikasi ini biasa disebut pemeliharaan perangkat lunak.



Gambar 2.2. Siklus Hidup Perangkat Lunak

Masing-masing tahapan dapat dipecah lagi menjadi sub-sub tahapan, dan tidak ada kesepakatan umum yang dipakai karena setiap proyek mempunyai cara pemecahan yang berbeda-beda.

Uji meliputi pengujian program, menggunakan data yang similar dengan data nyata yang dirancang untuk dieksekusi program, mengamati keluaran program, dan mengambil kesimpulan ada tidaknya kesalahan program atau kelainan pada keluaran.

2.1.2. Proses Pengujian

Proses pengujian seperti halnya proses pemrograman, dilakukan dalam bentuk tahapan demi tahapan yang saling berhubungan.

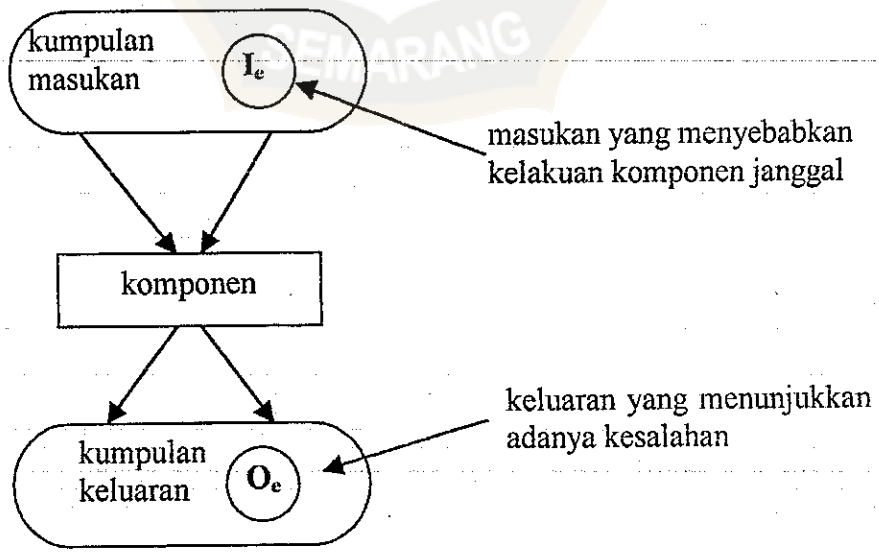
Tahapan uji yang berbeda di dalam proses pengujian adalah :

1. Uji Fungsi atau Uji Unit, merupakan uji tingkat dasar, dimana fungsi yang membentuk suatu modul diuji untuk memastikan bahwa fungsi beroperasi dengan benar.
2. Uji Modul, merupakan uji kerjasama antar unit yang menyusun modul tersebut.
3. Uji Subsistem, merupakan proses pengujian dimana modul secara bersama-sama membentuk subsistem. Uji subsistem hendaknya dikonsentrasikan pada uji antar muka modul.
4. Uji Sistem atau uji integrasi, dilakukan untuk menemukan kesalahan pada fase rancangan dan implementasi, serta menguji apakah sistem telah bekerja sesuai dengan spesifikasinya.
5. Uji Penerimaan, diperlukan untuk menemukan kesalahan yang berupa kesalahan di dalam mendefinisikan keperluan sistem. Keperluan yang didefinisikan mungkin belum mencerminkan fasilitas, dan kinerja yang diperlukan pemakai.

2.1.3. Pengujian Black Box (*Black Box Testing*) dan Pengujian White Box (*White Box Testing*)

Sebagaimana telah disebutkan pada Bab Pendahuluan, terdapat dua pendekatan pengujian perangkat lunak, yaitu Pengujian Black Box dan Pengujian White Box.

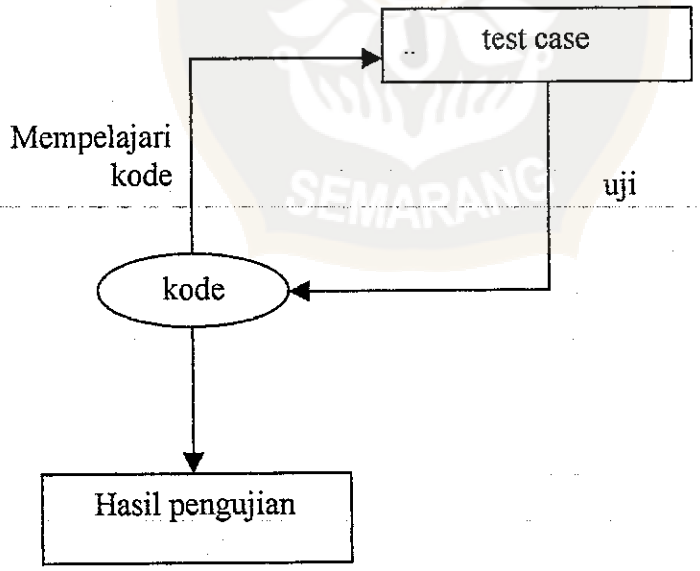
Pengujian Black Box yang juga dikenal sebagai pengujian Fungsional (*Functional Testing*) adalah sebuah pendekatan pengujian dimana spesifikasi dari komponen yang diuji dipergunakan untuk memperoleh test case. Komponen tersebut bersifat "*black box*" yang artinya kelakuan komponen tersebut hanya dapat ditentukan dengan mempelajari masukan dan dihubungkan dengan keluaran yang dihasilkan. Gambar 2.3 berikut ini menggambarkan model komponen yang diangkat pada pengujian Black Box.



Gambar 2.4. Pengujian Black Box

Masalah kunci bagi penguji perangkat lunak adalah pemilihan masukan yang mempunyai kemungkinan yang besar menjadi anggota I_c. Pemilihan yang efektif tergantung pada keahlian dan pengalaman dari penguji.

Pendekatan pengujian yang lain adalah Pengujian White Box atau Pengujian Glass Box atau juga disebut Pengujian Struktural (*Structural Testing*). Gambar 2.4 menggambarkan proses pada Pengujian White Box. Pengujian White Box berlawanan dengan Pengujian Black Box, sebab penguji dapat menganalisa kode program dan menggunakan pengetahuan tentang kode program, dan struktur komponen untuk memperoleh test case. Keunggulan pengujian White Box adalah bahwa test case dapat diperoleh secara sistematik.



Gambar 2.4. Pengujian White Box

2.2. Flowgraph

Pada subbab ini akan dikenalkan konsep flowgraph. Flowgraph merupakan bentuk formal dari struktur kontrol program (Fenton, 1993). Flowgraph adalah bentuk khusus dari digraph (graph berarah), karena flowgraph merupakan digraph yang mempunyai sebuah titik start dan sebuah titik stop. Sehingga notasi dan istilah-istilah pada teori graph digunakan pada pembahasan flowgraph ini. Berikut ini beberapa notasi, dan definisi dari teori graph dan flowgraph.

Definisi 2.1 :

Graph G adalah himpunan yang tidak kosong dari elemen-elemen yang disebut titik dan suatu daftar pasangan elemen yang tidak terurut yang disebut garis. Satu garis berbentuk "vw" atau "wv" dikatakan menghubungkan titik v dan titik w.

Definisi 2.2 :

Misal v dan w adalah titik-titik Graph G. Jika v dan w dihubungkan dengan garis e, maka v dan w dikatakan *adjacent*. Lebih jauh, v dan w dikatakan *incident* dengan e, dan e dikatakan *incident* dengan v dan w.

Definisi 2.3 :

Suatu *walk* (jalan) yang panjangnya k dalam Graph G adalah urutan k titik G yang berurutan. Contoh : uv, vw, wx, ... ,yz, walk ini dinotasikan dengan uvwx ... yz dan disebut walk antara u dan z.

Walk yang setiap garisnya berbeda disebut *trail*.

Definisi 2.4 :

Digraph D adalah himpunan elemen yang disebut titik dan daftar pasangan terurut dari elemen-elemen tersebut yang disebut busur. Jika v dan w adalah titik-titik D maka suatu busur vw dikatakan berarah dari v ke w atau menghubungkan v ke w.

Definisi 2.5 :

Misal D adalah digraph dan v titik pada D. Derajat keluar v adalah banyaknya busur yang incident dari v, dan dinotasikan OD. Dengan cara yang sama, derajat masuk v adalah banyaknya busur yang incident pada v dan dinotasikan dengan ID.

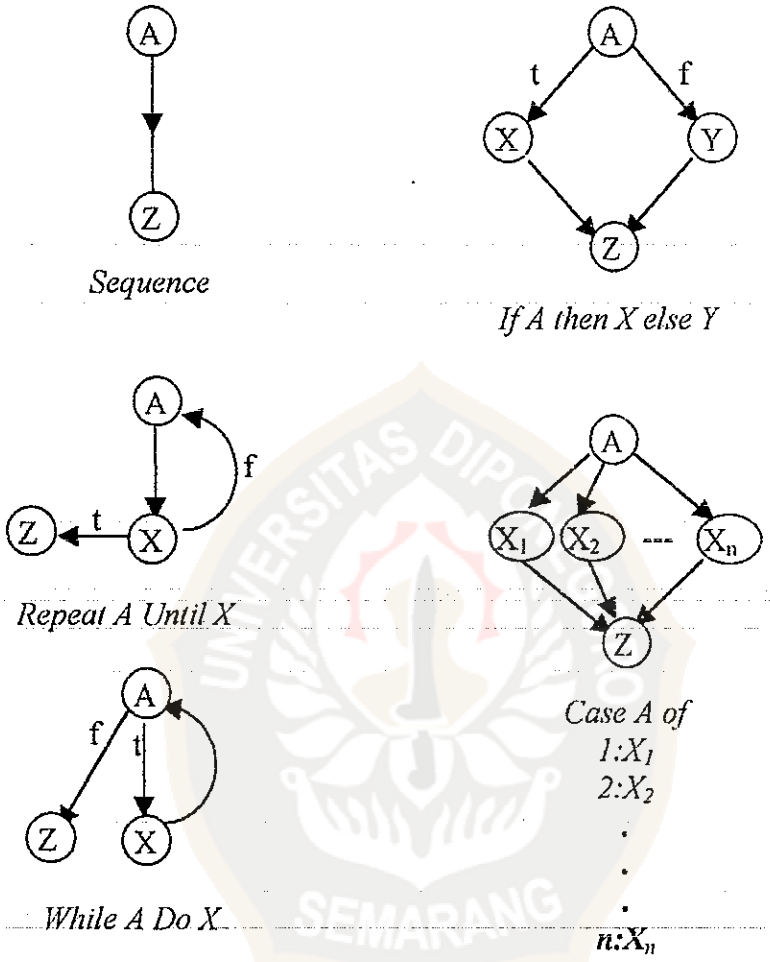
Definisi 2.6 :

Misal D adalah digraph, v dan w adalah titik-titik pada digraph D. Digraph D dikatakan terhubung kuat (*strongly connected*) apabila terdapat walk dari v ke w maupun dari w ke v.

Definisi 2.6 :

Sebuah flowgraph F = (G,a,z) terdiri dari sebuah digraph G, sebuah simpul "a" sebagai simpul start dan sebuah simpul "z" sebagai simpul stop. Simpul stop mempunyai OD sama dengan nol. Setiap simpul harus mempunyai sifat walk, yang berarti bahwa setiap simpul harus berada pada beberapa walk dari a ke z. Simpul-simpul dengan OD sama dengan satu disebut simpul prosedur dan untuk semua simpul kecuali simpul z disebut simpul predikat.

Berikut ini beberapa bentuk flowgraph yang umum :



Gambar 2.5. Bentuk-Bentuk Flowgraph yang Umum

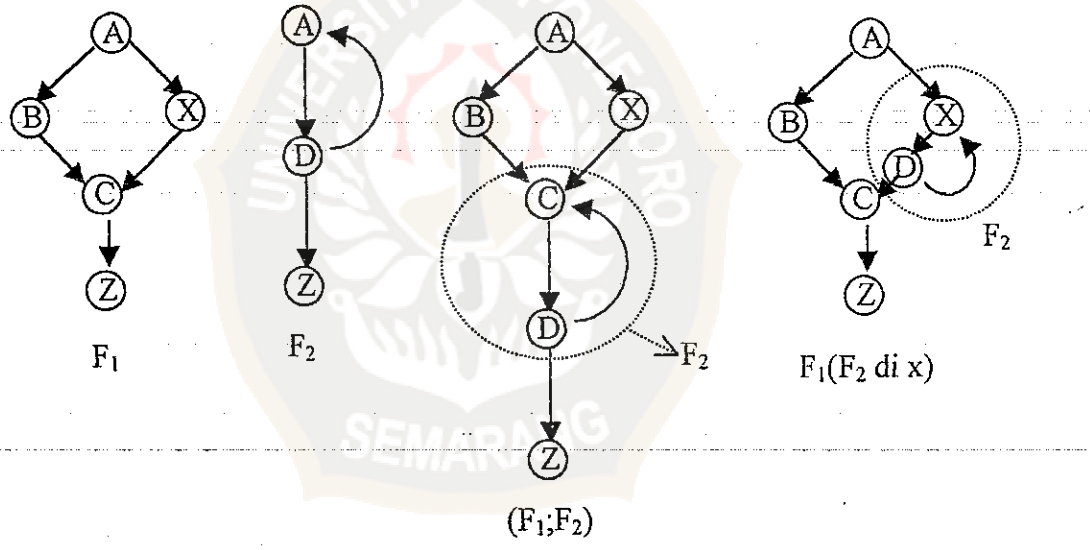
Definisi 2.7 :

Diberikan dua flowgraph F_1 dan F_2, dapat dibuat sebuah flowgraph baru yang disebut barisan F_1 dan F_2 dan ditulis $(F_1;F_2)$ dengan menjadikan simpul stop pada F_1 sebagai simpul start pada F_2.

Definisi 2.8 :

Misalkan F_1 dan F_2 adalah flowgraph dan x adalah simpul prosedur pada F_1. Dapat dibuat sebuah flowgraph baru yang disebut flowgraph tersarang F_2 pada F_1 di x dan ditulis $F_1(F_2$ di x), dengan mengganti sebuah garis yang keluar dari x dengan flowgraph F_2. Sehingga, simpul start F_2 sama dengan x dan simpul stop dari F_2 sama dengan simpul F_1 yang dituju x.

Contoh : operasi pada flowgraph



Gambar 2.6.
Barisan Flowgraph dan Flowgraph Tersarang

2.3. Algoritma Depth First Search

Algoritma Depth First Search dipergunakan untuk mengunjungi semua titik di sebuah graph. Apabila diberikan graph G dan sebuah titik v di G, maka Algoritma Depth First Search digunakan untuk mengunjungi semua titik yang dapat dijangkau dari v, atau dengan kata lain mencari titik yang terhubung ke v.

Algoritma Depth First Search untuk graph G adalah sebagai berikut :

```
Procedure Dfs( v : integer );
  { Graph G  dengan n titik dan sebuah larik boolean
  visited[v] yang  diset dengan nilai false }
Begin
  Visited[v] := true;
  For setiap titik w adjasent ke v do
      If not visited[w] then Dfs(w);
End;
```

Gambar 2.7. Algoritma Depth First Search

Jalannya Algoritma Depth First Search tersebut di atas dapat dijelaskan sebagai berikut : dimulai dengan pemanggilan prosedur Dfs untuk memproses titik v. Nilai variabel visited[v] diubah menjadi true. Hal ini dilakukan untuk menghindari adanya siklus (*cycle*) yaitu lintasan dengan titik awal dan titik akhir sama. Kemudian untuk setiap titik x yang adjacent dengan titik v secara rekursif dipanggil prosedur Dfs apabila nilai visited[x] masih false (belum pernah dikunjungi). Proses akan dihentikan apabila sudah tidak ada lagi titik dengan variabel visited[x] bernilai false dapat dijangkau dari titik-titik yang telah dikunjungi.