BAB 9
PENGUJIAN PERANGKAT LUNAK

Pengujian PL adalah elemen kritis dari jaminan kualitas PL dan merepresentasikan spesifikasi, desain dan pengkodean.
Meningkatnya visibilitas PL sbg suatu elemen sistem dan "biaya" yg muncul akibat kegagalan PL, memotivasi dilakukan perencanaan yg baik melalui pengujian yg teliti.

Dalam melakukan uji coba ada 2 masalah penting yang akan dibahas, yaitu :

A. Teknik uji coba PL
B. Strategi uji coba PL

TEKNIK UJI COBA PL

Pada dasarnya, pengujian merupakan suatu proses rekayasa PL yg dapat dianggap (secara psikologis) sebagai hal yg destruktif daripada konstruktif.

SASARAN PENGUJIAN (Glen Myers) :

1. Pengujian adalah proses eksekusi suatu program dengan maksud menemukan kesalahan.
2. Test case yg baik adalah test case yg memiliki probabilitas tinggi untuk menemukan kesalahan yg belum pernah ditemukan sebalumnya.
3. Pengujian yg sukses adalah pengujian yg mengungkap semua kesalahan yg belum pernah ditemukan sebelumnya.

PRINSIP PENGUJIAN (diusulkan Davis) :

- Semua pengujian harus dapat ditelusuri sampai ke persyaratan pelanggan.
- Pengujian harus direncanakan lama sebelum pengujian itu dimulai.
- Prinsip Pareto berlaku untuk pengujian PL. Prinsip Pareto mengimplikasikan 80% dari semua kesalahan yg ditemukan selama pengujian sepertinya akan dapat ditelusuri sampai 20% dari semua modul program.
- Pengujian harus mulai "dari yg kecil" dan berkembang ke pengujian "yang besar".
- Pengujian yg mendalam tidak mungkin.
- Paling efektif, pengujian dilakukan oleh pihak ketiga yg independen.

TESTABILITAS

Testabilitas PL adalah seberapa mudah sebuah program komputer dapat diuji. Karena pengujian sangat sulit, perlu diketahui apa yg dapat dilakukan untuk membuatnya menjadi mudah.

Karakteristik PL yg diuji :

- OPERABILITAS, semakin baik dia bekerja semakin efisien dia dapat diuji.
- OBSERVABILITAS, apa yg anda lihat adalah apa yg anda uji.
- KONTROLABILITAS, semakin baik kita dapat mengontrol PL semakin banyak pengujian yg adapat diotomatisasi dan dioptimalkan.

- DEKOMPOSABILITAS, dengan mengontrol ruang lingkup pengujian kita dapat lebih cepat mengisolasi masalah dan melakukan pengujian kembali.
- KESEDERHANAAN, semakin sedikit yg diuji semakin cepat pengujian.
- STABILITAS, semakin sedikit perubahan semakin sedikit gangguan pengujian.
- KEMAMPUAN DIPAHAMI, semakin banyak informasi yg dimiliki semakin detail pengujiannya.

ATRIBUT PENGUJIAN YG BAIK :

- Memiliki probabilitas yg tinggi menemukan kesalahan.
- Tidak redundan.
- Harusnya 'jenis terbaik'.
- Tidak boleh terlalu sederhana atau terlalu kompleks.

DESAIN TEST CASE

Terdapat bermacam-macam rancangan metode test case yg dapat digunakan, semua menyediakan pendekatan sistematis untuk uji coba, yg terpenting metode menyediakan kemungkinan yg cukup tinggi menemukan kesalahan.

Terdapat 2 macam test case:

1. Pengetahuan fungsi yg spesifik dari produk yg telah dirancang untuk diperlihatkan, test dapat dilakukan untuk menilai masing-masing fungsi apakah telah berjalan sebagaimana yg diharapkan.
2. Pengetahuan tentang cara kerja dari produk, test dapat dilakukan untuk memperlihatkan cara kerja dari produk secara rinci sesuai dengan spesifikasinya.

Dua macam pendekatan test yaitu :

1. Black Box Testing

Test case ini bertujuan untuk menunjukkan fungsi PL tentang cara beroperasinya, apakah pemasukan data keluaran telah berjalan sebagaimana yang diharapkan dan apakah informasi yang disimpan secara eksternal selalu dijaga kemutakhirannya.

2. White Box Testing

Adalah meramalkan cara kerja perangkat lunak secara rinci, karenanya logikal path (jalur logika) perangkat lunak akan ditest dengan menyediakan test case yang akan mengerjakan kumpulan kondisi dan atau pengulangan secara spesifik. Secara sekilas dapat diambil kesimpulan white box testing merupakan petunjuk untuk mendapatkan program yang benar secara 100%.

UJI COBA WHITE BOX

Uji coba white box adalah metode perancangan test case yang menggunakan struktur kontrol dari perancangan prosedural untuk mendapatkan test case. Dengan rnenggunakan metode white box, analis sistem akan dapat memperoleh test case yang:

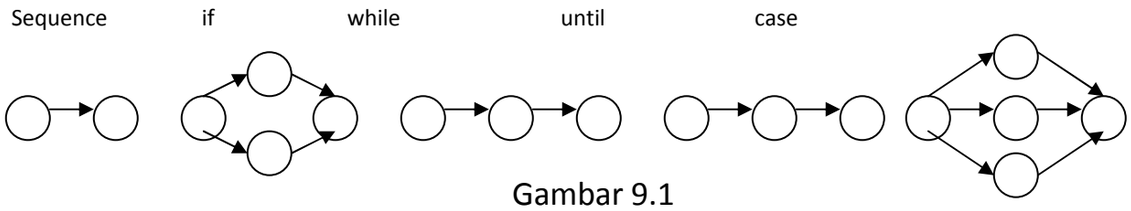
- menjamin seluruh independent path di dalam modul yang dikerjakan sekurang-kurangnya sekali

- mengerjakan seluruh keputusan logikal
- mengerjakan seluruh loop yang sesuai dengan batasannya
- mengerjakan seluruh struktur data internal yang menjamin validitas

1. UJI COBA BASIS PATH

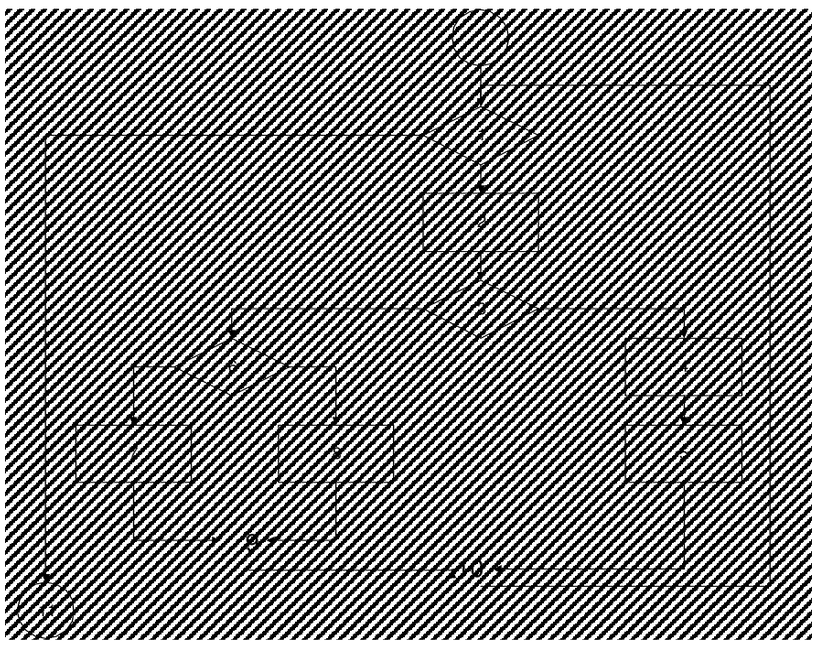
Uji coba basis path adalah teknik uji coba white box yg diusulkan Tom McCabe. Metode ini memungkinkan perancang test case mendapatkan ukuran kekompleksan logical dari perancangan prosedural dan menggunkan ukuran ini sbg petunjuk untuk mendefinisikan basis set dari jalur pengerjaan. Test case yg didapat digunakan untuk mengerjakan basis set yg menjamin pengerjaan setiap perintah minimal satu kali selama uji coba.

1.1. Notasi diagram alir



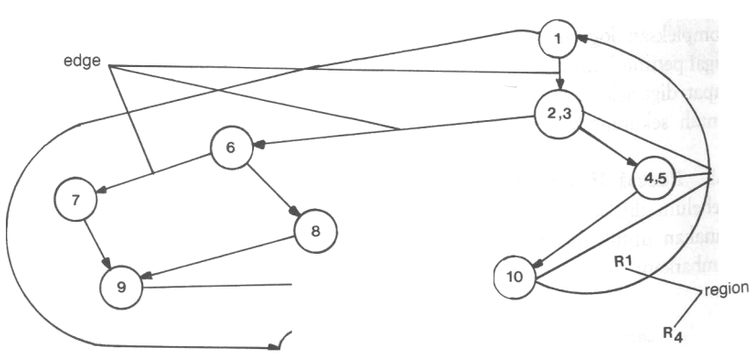
Gambar 9.1

Untuk menggambarkan pemakaian diagram alir diberikan contoh perancangan prosedural dalam bentuk flowchart



Gambar 9.2 Diagram Alir

Selanjutnya diagram alir diatas dipetakan ke grafik alir



Gambar 9.3 Grafik Alir

Lingkaran/node :

menggambarkan satu/lebih perintah prosedural. Urutan proses dan keputusan dapat dipetakan dalam satu node.

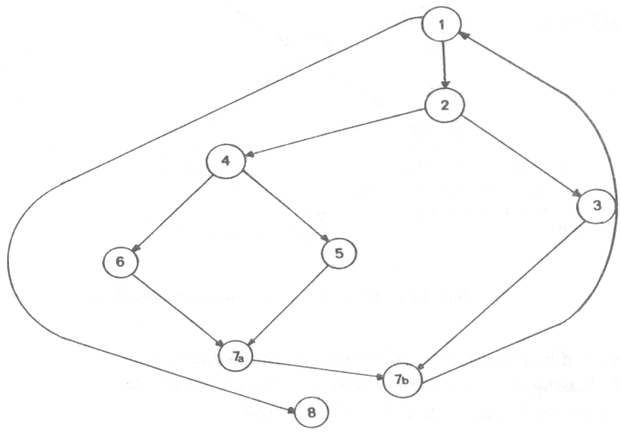
Tanda panah/edge :

menggambarkan aliran kontrol. Setiap node harus mempunyai tujuan node

Region :

adalah daerah yg dibatasi oleh edge dan node. Termasuk daerah diluar grafik alir.

Contoh menterjemahkan pseudo code ke grafik alir

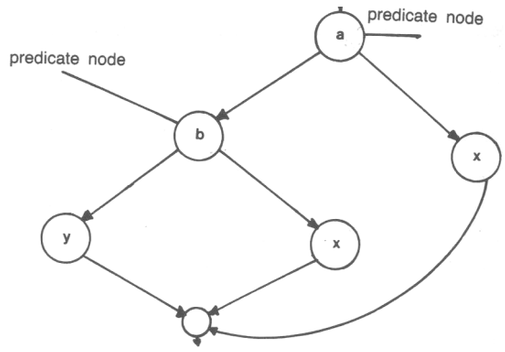



```
1:  do while record masih ada
        baca record
2: if record ke 1 = 0
3: then proses record
        simpan di buffer
        naikan kounter
4: else if record ke 2 = 0
5       then reser kounter
6       proses record
                simpan pada file
7a: endif
        endif
7b: enddo
8 :  end
```

Gambar 9.4 Menerjemahkan PDL ke grafik Alir

Nomor pd pseudo code berhubungan dengan nomor node. Apabila diketemukan kondisi majemuk (compound condition) pada pseudo cade pembuatan grafik alir menjadi rumit. Kondisi majemuk mungkin terjadi pada operator Boolean (AND, OR, NAND, NOR) yg dipakai pada perintah if.

Contoh :




```
if A or B
     then procedure x
     else procedure y
endif
```

Gambar 9.5 Logika Gabungan

Node dibuat terpisah untuk masing-masing kondisi A dan B dari pernyataan IF A OR B. Masing-masing node berisi kondisi yg disebut **pridicate node** dan mempunyai karakteristik dua atau lebih edge darinya.

1.2. CYCLOMATIC COMPLEXITY

Cyclomatic complexity adalah metrik PL yang menyediakan ukuran kuantitatif dari kekompleksan logikal program. Apabila digunakan dalam kontek metode uji coba basis path, nilai yang dihitung untuk cyclomatic complexity menentukan jumlah jalur independen dalam basis set suatu program dan memberi batas atas untuk jumlah uji coba yang harus dikerjakan untuk menjamin bahwa seluruh perintah sekurang-kurangnya telah dikerjakan sekali.

Jalur independent adalah jalur yang melintasi atau melalui program dimana sekurang-kurangnya terdapat proses perintah yang baru atau kondisi yang baru.

Dari gambar 9.3 :
Path 1 : 1 - 11
Path 2 : 1 - 2 - 3 - 4 - 5 - 10 - 1 - 11
Path 3 : 1 - 2 - 3 - 6 - 8 - 9 ...: 10 - 1 - 11
Path 4 ': 1 - 2 - 3 - 6 - 7 - 9 - 10 - 1 - 11
Path 1,2,3,4 yang telah didefinisikan di atas merupakan basis set untuk diagaram alir.

Cyclomatic complexity digunakan untuk mencari jumlah path dalam satu flowgraph. Dapat dipergunakan rumusan sbb :
1. Jumlah region grafik alir sesuai dengan cyclomatic complexity.
2. Cyclomatix complexity V(G) untuk grafik alir dihitung dengan rumus:
   $V(G) = E - N + 2$
   dimana:
   E = jumlah edge pada grafik alir
   N = jumlah node pada grafik alir
3. Cyclomatix complexity V(G) juga dapat dihitung dengan rumus:
   $V(G) = P + 1$

dimana P = jumlah predicate node pada grafik alir

Pada Gambar 9.3 dapat dihitung cyclomatic complexity:

1. Flowgraph mempunyai 4 region
2. V(G) = 11 edge - 9 node + 2 = 4
3. V(G) = 3 predicate node + 1 = 4

Jadi cyclomatic complexity untuk flowgraph Gambar 9.3 adalah 4

1.3. MELAKUKAN TEST CASE

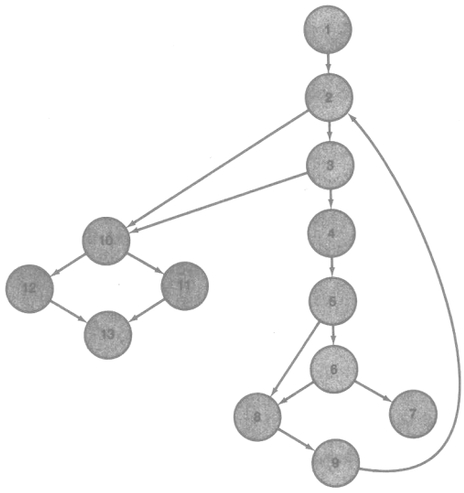
Metode uji coba basis path juga dapat diterapkan pada perancangan prosedural rinci atau program sumber. Pada bagian ini akan dijelaskan langkah-langkah uji coba basis path. Prosedur rata-rata pada bagian berikut akan digunakan sebagai contoh dalam pembuatan test case.

```
PROCEDURE RATA-RATA

INTERFACE RESULT rata, total, input, total.valid
INTERFACE RESULT nilai, minim, max
TYPE NILAl (1:100) IS SCALAR ARRAY;
TYPE rata, total. input, total.valid, max.minim, jumlah IS SCALAR;
TYPE I IS INTEGER;
I = 1;
total. input = total. valid = 0;
jumlah = 0;
DO WHILE nilai(i) <> -999 .and. total.input < 100
tambahkan total.input dengan 1;
IF nilai(i) >= minimum .and. nilai(i} <=max;
THEN tambahkan total.valid dengan I;
  jumlah=jumlah + nilai(i);
ELSE skip;
END IF
tambahkan i dengan 1;
ENDDO
  IF total. valid> 0
THEN rata =jumlah/total. valid;
ELSE rata = -999;
ENDIF
END
```

Langkah-langkah pembuatan test case:

1. Dengan mempergunakan perancangan prosedural atau program sumber sebagai dasar, digambarkan diagram alirnya.



Gambar 9.6 Diagram Alir prosedur rata

2. Tentukan cyclomatic complexity untuk diagram alir yang telah dibuat:
   V(G) = 6 region .
   V(G) = 17 edge - 13 node + 2 = 6
   V(G) = 5 predicate node + 1 = 6
3. Tentukan independent path pada flowgraph
   Dari hasil perhitungan cyclomatic complexity terdapat 6 independent path yaitu:

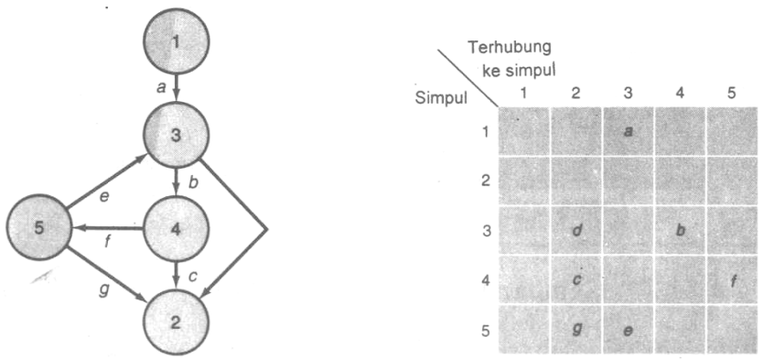
   path 1 : 1-2-10-11-13
   path 2 : 1-2-10-12-13
   path 3 : 1-2-3-10-11-13
   path 4 : 1-2-3-4-5-8-9-2-..
   path 5 : 1-2-3-4-5-6-8-9-2-..
   path 6 : 1-2-3-4-5-6-7-8-9-2-...

4. Buat test case yang akan mengerjakan masing-masing path pada basis set. Data yang dipilih harus tepat sehingga setiap kondisi dari predicate node dikerjakan semua.

1.4. GRAPH METRIK

Graph metrik merupakan PL yang dikembangkan untuk membantu uji coba basis path atau struktur data. Graph metrik adalah matrik empat persegi yang mempunyai ukuran (sejumlah baris dan kolom) yang sama dengan jumlah node pada flowgraph. Masing-masing baris dan kolom mempunyai hubungan dengan node yang telah ditentukan dan pemasukan data matrik berhubungan dengan hubungan (edge) antanode.

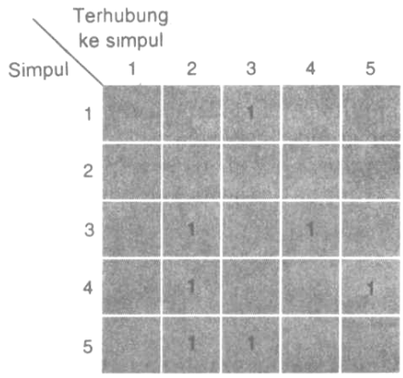
Contoh sederhana pemakaian graph matrik dapat digambarkan sbb :



Gambar 9.7. Graph matrik

Pada gambar flowgraph masing-masing node ditandai dengan angka clan 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. Dapat juga hubungan bobot diberi tanda dengan:

- kemungkinan link (edge) dikerjakan
- waktu yang digunakan untuk proses selama traversal dari link
- memori yang diperlukan selama traversal link
- sumber daya yang diperlukan selama traversal link



Gambar 9.8 Hubungan bobot

Koneksi :
1 – 1 = 0
2 – 1 = 1
2 – 1 = 1
2 – 1 = 1
3 + 1 = 4 cyclomatic complexity

2. PENGUJIAN LOOP

Loop merupakan kendala yang sering muncul untuk menerapkan algoritma dengan tepat. Uji coba loop merupakan teknik pengujian white box yg fokusnya pada validitas dari loop.
Kelas loop yaitu :

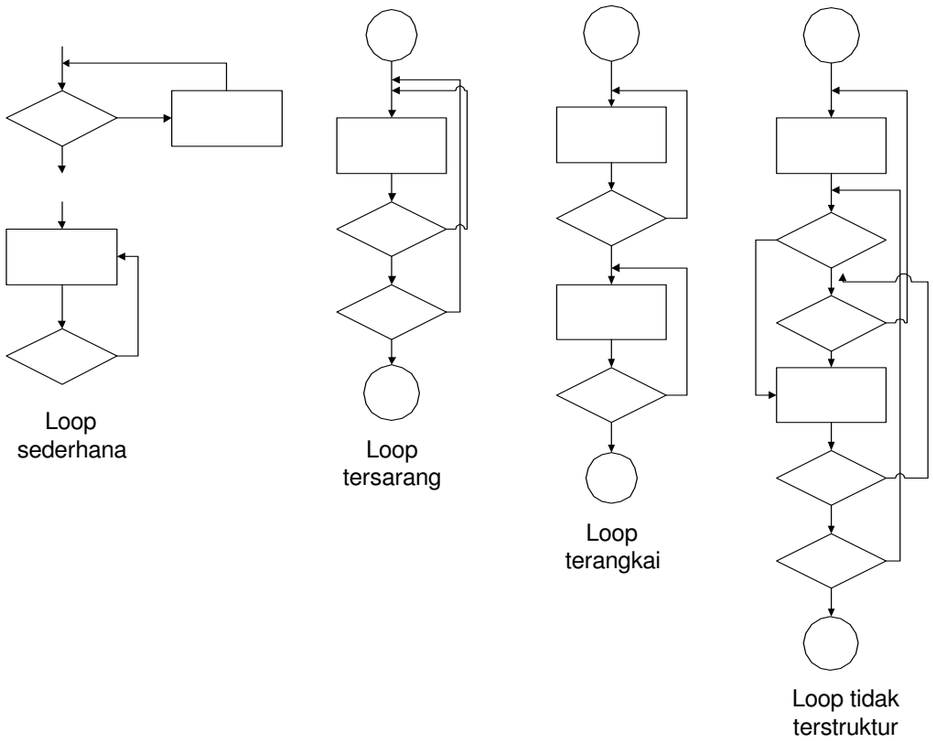
a. **Loop Sederhana**, pengujian loop sederhana dilakukan dgn mudah, dimana *n* jumlah maksimum yg diijinkan melewati loop tsb.

1. Lewati loop secara keseluruhan
2. Hanya satu yg dapat melewati loop
3. *m* dapat melewati loop dimana *m< n*

b. **Loop Tersarang**, pengujian loop ini menggunakan pendekatan loop sederhana. Petunjuk pengujian loop tersarang :
1. Dimulai dari loop paling dalam. Atur semua loop ke nilai minimum.
2. Kerjakan dgn prinsip loop sederhana untuk loop yg paling dalam sementara tahan loop yg di luar pada parameter terkecil (nilai kounter terkecil)
3. Kemudian lanjutkan untuk loop yg diatasnya.
4. Teruskan sampai semua loop selesai di uji.

c. Loop Terangkai, pengujian loop ini menggunakan pendekatan loop sederhana bila masing-masing loop independen, tetapi bila dua loop dirangkai dan pencacah loop 1 digunakan sebagai harga awal loop 2 maka loop tsb jadi tidak independen, maka pendekatan yg diaplikasikan ke loop tersarang direkomendasikan.

d. Loop Tidak Terstruktur, Kapan saja memungkinkan, loop ini didisain kembali agar mencerminkan penggunaan komsepsi pemrograman tertruktur.



Gambar 9.9. Macam-macam loop

PENGUJIAN BLACK-BOX

Pengujian black-box berfokus pada persyaratan fungsional PL. Pengujian inimemungkinkan analis system memperoleh kumpulan kondisi input yg akan mengerjakan seluruh keperluan fungsional program.
Tujuan metode ini mencari kesalaman pada:

- Fungsi yg salah atau hilang
- Kesalahan pada interface
- Kesalahan pada struktur data atau akses database
- Kesalahan performansi
- Kesalahan inisialisasi dan tujuan akhir

Metode ini tidak terfokus pada struktur kontrol seperti pengujian white-box tetapi pada domain informasi.

Pengujian dirancang untuk menjawab pertanyaan sbb:

- Bagaimana validitas fungsional diuji?
- Apa kelas input yg terbaik untuk uji coba yg baik?
- Apakah sistem sangat peka terhadap nilai input tertentu?
- Bagaimana jika kelas data yang terbatas dipisahkan?
- Bagaimana volume data yg dapat ditoleransi oleh sistem?
- Bagaimana pengaruh kombinasi data terhadap pengoperasian system?

1. EQUIVALENCE PARTITIONING

Equivalence partitioning adalah metode pengujian black-box yg memecah atau membagi domain input dari program ke dalam kelas-kelas data sehingga test case dapat diperoleh.

Perancangan test case equivalence partitioning berdasarkan evaluasi kelas equivalence untuk kondisi input yg menggambarkan kumpulan keadaan yg valid atau tidak. Kondisi input dapat berupa nilai numeric, range nilai, kumpulan nilai yg berhubungan atau kondisi Boolean.

Contoh :
Pemeliharaan data untuk aplikasi bank yg sudah diotomatisasikan. Pemakai dapat memutar nomor telepon bank dengan menggunakan mikro komputer yg terhubung dengan password yg telah ditentukan dan diikuti dengan perintah-perintah. Data yg diterima adalah :

Kode area	: kosong atau 3 digit
Prefix	: 3 digit atau tidak diawali 0 atau 1
Suffix	: 4 digit
Password	: 6 digit alfanumerik
Perintah	: check, deposit, dll

Selanjutnya kondisi input digabungkan dengan masing-masing data elemen dapat ditentukan sbb :

Kode area	: kondisi input, Boolean – kode area mungkin ada atau tidak kondisi input, range – nilai ditentukan antara 200 dan 999
Prefix	: kondisi input range > 200 atau tidak diawali 0 atau 1
Suffix	: kondisi input nilai 4 digit
Password	: kondisi input boolean – pw mungkin diperlukan atau tidak kondisi input nilai dengan 6 karakter string
Perintah	: kondisi input set berisi perintah-perintah yang telah didefinisikan

2. BOUNDARY VALUE ANALYSIS

Untuk permasalahan yg tidak diketahui dg jelas cenderung menimbulkan kesalahan pada domain outputnya. BVA merupakan pilihan test case yg mengerjakan nilai yg

telah ditentukan, dgn teknik perancangan test case melengkapi test case equivalence partitioning yg fokusnya pada domain input. BVA fokusnya pada domain output.

Petunjuk pengujian BVA :

1. Jika kondisi input berupa range yg dibatasi nilai a dan b, test case harus dirancang dgn nilai a dan b.
2. Jika kondisi input ditentukan dgn sejumlah nilai, test case harus dikembangkan dgn mengerjakan sampai batas maksimal nilai tsb.
3. Sesuai petunjuk 1 dan 2 untuk kondisi output dirancang test case sampai jumlah maksimal.
4. Untuk struktur data pada program harus dirancang sampai batas kemampuan.

STRATEGI PENGUJIAN PL

Strategi uji coba PL memudahkan para perancang untuk menentukan keberhasilan system yg telah dikerjakan. Hal yg harus diperhatikan adalah langkah-langkah perencanaan dan pelaksanaan harus direncanakan dengan baik dan berapa lama waktu, upaya dan sumber daya yg diperlukan.

Strategi uji coba mempunyai karakteristik sbb :

- Pengujian mulai pada tingkat modul yg paling bawah, dilanjutkan dgn modul di atasnya kemudian hasilnya dipadukan.
- Teknik pengujian yang berbeda mungkin menghasilakn sedikit perbedaan (dalam hal waktu)
- Pengujian dilakukan oleh pengembang perangkat lunak dan (untuk proyek yang besar) suatu kelompok pengujian yang independen.
- Pengujian dan debugging merupakan aktivitas yang berbeda, tetapi debugging termasuk dalam strategi pengujian.

Pengujian PL adalah satu elemen dari topik yang lebih luas yang sering diacu sebagai *verifikasi dan validasi (V& V).*

Verifikasi : Kumpulan aktifitas yg menjamin penerapan PL benar-benar sesuai dgn fungsinya.

Validasi : Kumpulan aktivitas yang berbeda yang memastikan bahwa PL yang dibangun dapat memenuhi keperluan pelanggan.

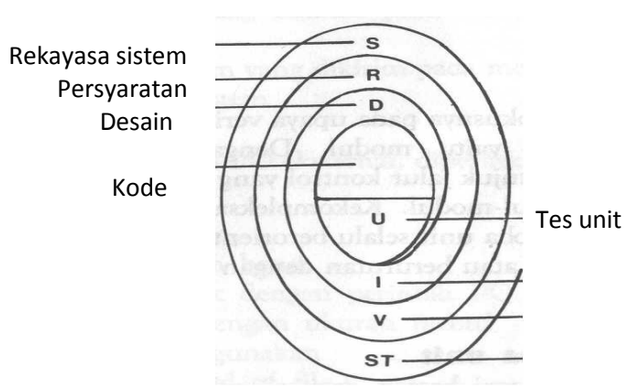
Dgn kata lain :

Verifikasi : " Apakah kita membuat produk dgn benar?"

Validasi : " Apakah kita membuat benar-benar suatu produk?"

Definisi dari V&V meliputi berbagai aktivitas yang kita rujuk sebagai jaminan kualias PL (SQA).

Pengujian merupakan salah satu tugas yg ada dlm arus siklus pengembangan system yg dapat digambarkan dalam bentuk spiral :



Tes integrasi
Tes validasi

Tes sistem

Gambar 9.10. Strategi Uji Coba

1. PENGUJIAN UNIT

Unit testing (uji coba unit) fokusnya pada usaha verifikasi pada unit terkecil dari desain PL, yakni modul. Uji coba unit selalu berorientasi pada white box testing dan dapat dikerjakan paralel ayau beruntun dengan modul lainnya.

1.1 Pertimbangan Pengujian Unit

Interface diuji cobakan untuk menjamin informasi yg masuk atau yg ke luar dari unit program telah tepat atau sesuai dgn yg diharapkan. Yg pertama diuji coba adalah interface karena diperlukan untuk jalannya informasi atau data antar modul.
Myers mengusulkan checklist untuk pengujian interface:

- Apakahjumlah parameter input sama dengan jumlah argumen?
- Apakah antara atribut dan parameter argumen sudah cocok?
- Apakah antara sistem satuan parameter dan argumen sudah cocok?
- Apakah jumlah argumen yang ditransmisikan ke modul yang dipanggil sama dengan jumlah parameter?
- Apakah atribut dari argumen yang ditransmisikan ke modul yang dipanggil sama dengan atribut parameter?
- Apakah sistem unit dari argumen yang ditransmisikan ke modul yang dipanggil sama dengan sistem satuan parameter?
- Apakah jumlah atribut dari urutan argumen ke fungsi-fungsi built-in sudah benar?
- Adakah referensi ke parameter yang tidak sesuai dengan pain entri yang ada?
- Apakah argumen input-only diubah?
- Apakah definisi variabel global konsisten dengan modul?
- Apakah batasan yang dilalui merupakan argumen?

Bila sebuah modul melakukan I/O ekstemal, maka pengujian interface tambahan harus dilakukan.

- Atribut file sudah benar?
- Pemyataan OPEN/CLOSE sudah benar?
- Spesifikasi format sudah cocok dengan pernyataan I/O?
- Ukuran buffer sudah cocok dengan ukuran rekaman?
- File dibuka sebelum penggunaan?
- Apakah kondisi End-of-File ditangani?
- Kesalahan I/O ditangani?

- Adakah kesalahan tekstual di dalam informasi output?

Kesalahan yang umum di dalam komputasi adalah:

- kesalah-pahaman atau prosedur aritmatik yang tidak benar
- operasi mode yang tercampur
- inisialisasi yang tidak benar
- inakurasi ketelitian
- representasi simbolis yang tidak benar dari sebuah persamaan.

Test case harus mengungkap kesalahan seperti

- perbandingan tipe data yang berbeda
- preseden atau operator logika yang tidak benar
- pengharapan akan persamaan bila precision error membuat persamaan yang tidak mungkin
- perbandingan atau variabel yang tidak benar
- penghentian loop yang tidak ada atau tidak teratur
- kegagalan untuk keluar pada saat terjadi iterasi divergen
- variabel loop yang dimodifikasi secara tidak teratur.

1.2. Prosedur Pengujian Unit

Program sumber telah dikembangkan, ditunjang kembali dan diverifikasi untuk sintaksnya, maka perancangan test case dimulai. Peninjauan kembali perancangan informasi akan menyediakan petunjuk untuk menentukan test case. Karena modul bukan program yg berdiri sendiri maka driver (pengendali) dan atau stub PL harus dikembangkan untuk pengujian unit.

Driver adl program yg menerima data untuk test case dan menyalurkan ke modul yg diuji dan mencetak hasilnya.
Stub melayani pemindahan modul yg akan dipanggil untuk diuji.

2. PENGUJIAN INTEGRASI

Pengujian terintegrasi adl teknik yg sistematis untuk penyusunan struktur program, pada saat bersamaan dikerjakan uji coba untuk memeriksa kesalahan yg nantinya digabungkan dengan interface.

Metode pengujian

- top down integration
- buttom up integration

2.1. TOP DOWN INTEGRATION

Merupakan pendekatan inkrmental untuk penyusunan struktur program. Modul

dipadukan dgn bergerak ke bawah melalui kontrol hirarki dimulai dari modul utama. Modul subordinat ke modul kontrol utama digabungkan ke dalam struktur baik menurut depth first atau breadth first.

Proses integrasi:

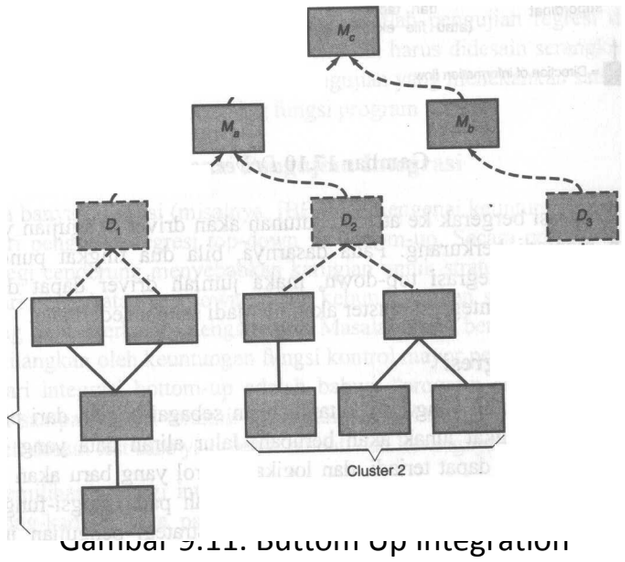
- modul utama digunakan sebagai test driver dan stub yg menggantikan seluruh modul yg secara langsung berada di bawah modul kontrol utama.
- Tergantung pada pendekatan perpaduan yg dipilih (depth / breadth)
- Uji coba dilakukan selama masing-masing modul dipadukan
- Pada penyelesaian masing-masing uji coba stub yg lain dipindahkan dgn modul sebenarnya.
- Uji coba regression yaitu pengulangan pengujian untuk mencari kesalahan lain yg mungkin muncul.

2.2. BOTTOM UP INTEGRATION

Pengujian buttom up dinyatakan dgn penyusunan yg dimulai dan diujicobakan dgn atomic modul (yi modul tingkat paling bawah pd struktur program). Karena modul dipadukan dari bawah ke atas, proses yg diperlukan untuk modul subordinat yg selalu diberikan harus ada dan diperlukan untuk stub yg akan dihilangkan.

Strategi pengujian :

- Modul tingkat bawah digabungkan ke dalam cluster yg memperlihatkan subfungsi PL
- Driver (program kontrol pengujian) ditulis untuk mengatur input test case dan output
- Cluster diuji
- Driver diganti dan cluster yg dikombinasikan dipindahkan ke atas pada struktur program



Gambar 9.11. Buttom Up Integration

3. UJI COBA VALIDASI

Setelah semua kesalahan diperbaiki maka langkah selanjutnya adalah validasi terting. Pengujian validasi dikatakan berhasil bila fungsi yg ada pada PL sesuai dgn yg diharapkan pemakai.
Validasi PL merupakan kumpulan seri uji coba black box yg menunjukkan sesuai dgn yg diperlukan.

Kemungkinan kondisi setelah pengujian:

1. Karakteristik performansi fungsi sesuai dgn spesifikasi dan dapat diterima.
2. Penyimpangan dari spesifikasi ditemukan dan dibuatkan daftar penyimpangan.

Pengujian BETA dan ALPHA
Apabila PL dibuat untuk pelanggan maka dapat dilakukan aceeptance test sehingga memungkinkan pelanggan untuk memvalidasi seluruh keperluan. Test ini dilakukan karena memungkinkan pelanggan menemukan kesalahan yg lebih rinci dan membiasakan pelanggan memahami PL yg telah dibuat.
Pengujian Alpha
Dilakukan pada sisi pengembang oleh seorang pelanggan. PL digunakan pada setting yg natural dgn pengembang “yg memandang” melalui bahu pemakai dan merekam semua kesalahan dan masalah pemakaian.

Pengujian Beta
Dilakukan pada satu atau lebih pelanggan oleh pemakai akhir PL dalam lingkungan yg sebenarnya, pengembang biasanya tidak ada pada pengujian ini. Pelanggan merekan semua masalah (real atau imajiner) yg ditemui selama pengujian dan melaporkan pada pengembang pada interval waktu tertentu.

4. UJI COBA SISTEM

Pada akhirnya PL digabungkan dgn elemen system lainnya dan rentetan perpaduan system dan validasi tes dilakukan. Jika uji coba gagal atau di luar skope dari proses daur siklus pengembangan system, langkah yg diambil selama perancangan dan pengujian dapat diperbaiki. Keberhasilan perpaduan PL dan system yg besar merupakan kuncinya.

Sistem testing merupakan rentetan pengujian yg berbeda-beda dgn tujuan utama mengerjakan keseluruhan elemen system yg dikembangkan.

4.1. Recovery Testing
Adalah system testing yg memaksa PL mengalami kegagalan dalam bermacam-macam cara dan memeriksa apakah perbaikan dilakukan dgn tepat.

4.2. Security Testing
Adalah pengujian yg akan melalukan verifikasi dari mekanisme perlindungan yg akan dibuat oleh system, melindungi dari hal-hal yg mungkin terjadi.

4.3. Strees Testing

Dirancang untuk menghadapi situasi yg tidak normal pada saat program diuji. Testing ini dilakukan oleh system untuk kondisi seperti volume data yg tidak normal (melebihi atau kurang dari batasan) atau frkkuensi.