Black-Box Testing

SI-318 Testing dan Implementasi Sistem

Rosa Ariani Sukamto, ST

Pendahuluan

- Black-Box Testing terfokus pada spesifikasi fungsional dari perangkat lunak
  - *Tester* dapat mendefinisikan kumpulan kondisi input dan melakukan pengetesan pada spesifikasi fungsional program.
- Black Box Testing bukanlah solusi alternatif dari White-Box Testing tapi lebih merupakan pelengkap untuk menguji hal-hal yang tidak dicakup oleh White-Box Testing.
- Black-Box Testing cenderung untuk menemukan hal-hal berikut:
  - Fungsi yang tidak benar atau tidak ada
  - Kesalahan antarmuka (*interface errors*)
  - Kesalahan pada struktur data dan akses basis data
  - Kesalahan performansi (*performance errors*)
  - Kesalahan inisialisasi dan terminasi.
- Pengujian didesain untuk menjawab pertanyaan-pertanyaan berikut:
  - Bagaimana fungsi-fungsi diuji agar dapat dinyatakan valid?
  - Input seperti apa yang dapat menjadi bahan kasus uji yang baik?
  - Apakah sistem sensitif pada input-input tertentu?
  - Bagaimana sekumpulan data dapat diisolasi?
  - Berapa banyak rata-rata data dan jumlah data yang dapat ditangani sistem?
  - Efek apa yang dapat membuat kombinasi data ditangani spesifik pada operasi sistem?

Black Box Testing

- Equivalence Partitioning
- Boundary Value Analysis/Limit Testing
- Comparison Testing
- Sample Testing
- Robustness Testing
- Behavior Testing
- Requirement Testing
- Performance Testing
- Uji Ketahanan (*Endurance Testing*)
- Uji Sebab-Akibat (*Cause-Effect Relationship Testing*)

Equivalence Partitioning

- Membagi input menjadi kelas-kelas data yang dapat digunakan untuk menggenerasi kasus uji.
- Bertujuan untuk menemukan kelas-kelas kesalahan.
- Berdasarkan pada kesamaan kelas-kelas kondisi input.
- Sebuah kelas yang ekivalen merepresentasikan kumpulan status/kondisi yang valid atau tidak valid
- Sebuah kondisi input dapat berupa nilai numerik yang spesifik, rentang nilai, kumpulan nilai yang berkaitan, atau kondisi *boolean*.
- Kelas ekivalen dapat didefinisikan dengan kondisi berikut:
  - Jika kondisi input mensyaratkan rentang nilai atau nilai yang spesifik, maka sebuah kelas ekivalen yang valid dan dua buah kelas ekivalen yang tidak valid akan terbentuk
  - Jika sebuah kondisi input mensyaratkan sebuah boolean atau anggota dari sebuah himpunan, maka sebuah kelas ekivalen yang valid dan sebuah kelas ekivalen yang tidak valid akan terbentuk.
- Kasus uji untuk setiap domain input item data harus dikembangkan dan dieksekusi.

Contoh (1)

- Spesifikasi sub-program yang harus diuji (status awal dan akhir)
  - Sub-program mengambil sebuah input berupa integer pada rentang [-100,100]
  - Keluaran sub-program adalah tanda dari nilai masukan (0 dianggap positif)
- Dua buah kelas ekivalen yang valid
  - Nilai masukan dengan rentang [-100,0] akan menghasilkan tanda negatif sebagai keluaran
  - Nilai masukan dengan rentang [0,100] akan menghasilkan tanda positif sebagai keluaran
  - Keduanya bisa disatukan menjadi Nilai masukan dengan rentang [-100,100] merupakan rentang yang valid
- Dua buah kelas ekivalen yang tidak valid
  - Nilai masukan < -100
  - Nilai masukan > 100

Aturan Kelas Ekivalen (1)

- Jika sebuah kondisi mengacu pada nilai masukan yang memiliki rentang maka nilai yang ada dalam rentang masuk menjadi sebuah kelas ekivalen valid, dan nilai masukan diluar kedua sisi ambang batas menjadi dua buah kelas ekivalen yang tidak valid.
- Jika sebuah kondisi mengacu pada satu atau sejumlah nilai masukan dengan nilai tertentu maka akan menjadi sebuah kelas ekivalen yang valid yang berisi satu atau sejumlah nilai yang valid dan dua buah kelas ekivalen yang tidak valid yaitu kelas yang berisi masukan kosong dan masukan diluar nilai yang dispesifikasikan
- Jika sebuah kondisi mengacu pada himpunan nilai masukan tertentu maka akan menjadi sejumlah anggota himpunan kelas ekivalen yang valid dan sebuah kelas ekivalen yang tidak valid dimana jika masukan memiliki nilai diluar anggota himpunan
- Jika sebuah kondisi mengekspresikan sebuah kalimat yang berisi “harus” misal, awal masukan harus berupa karakter maka akan menjadi sebuah kelas ekivalen yang valid yaitu semua masukan yang diawal karakter dan sebuah kelas ekivalen yang tidak valid yaitu masukan yang tidak diawali karakter
- Jika ada alasan yang menyatakan bahwa elemen di dalam sebuah kelas ekivalen diproses dengan cara yang berbeda maka kelas ekivalen itu harus dibagi-bagi lagi menjadi kelas-kelas ekivalen yang lebih kecil berdasarkan pendekatan lojik dari sub-program pada level algoritma.

Aturan Kelas Ekivalen (2)

- Jika sub-program memiliki beberapa masukan yang independen, maka kelas ekivalen atau kasus uji harus dibuat berdasarkan per masukan dan kombinasi kemungkinan per masukan dimana per masukan dibuat dua buah kelas ekivalen yang valid dan sebuah kelas ekivalen yang tidak valid
- Tulis kasus uji untuk setiap kelas ekivalen yang tidak valid
  - Contoh:
    - Sebuah spesifikasi menyatakan untuk memasukkan masukan berupa tipe dan nomor mobil
    - Dalam kasus uji harus dibuat dimana jika kedua masukan bukan merupakan nilai yang valid
- Untuk menguji sebuah sub-program yang memiliki beberapa masukan yang saling bergantung maka kelas ekivalen harus dibuat berdasarkan kombinasi masukan

Contoh (2)

- Spesifikasi Kebutuhan
  - Diberikan 3 nilai yang merepresentasikan panjang sisi segitiga, definisikan segitiga apakah merupakan segitiga sama sisi, sama kaki, atau segitiga sembarang
- Tiga buah kelas ekivalen yang valid
  - 3 nilai sama dan positif
  - 3 nilai positif dan 2 nilai sama dan jumlah kedua kaki lebih besar dibandingkan nilai sisi yang bukan kaki
  - 3 nilai positif dan jumlah dua buah nilai lebih besar dari nilai lainnya
- Dua kelas ekivalen yang tidak valid
  - 2 nilai positif dan sebuah nilai negatif atau nol
  - 3 nilai positif dan jumlah dua buah nilai sama atau kurang dari sisi lainnya

Boundary Value Analysis / Limit Testing

- Banyak kesalahan terjadi pada kesalahan masukan.
- BVA mengijinkan untuk menyeleksi kasus uji yang menguji batasan nilai input.
- BVA merupakan komplemen dari *equivalence partitioning*. Lebih pada memilih elemen-elemen di dalam kelas ekivalen pada bagian sisi batas dari kelas.
- Contoh:
  - Untuk rentang yang dibatasi a dan b maka uji (a-1), a, (a+1), (b-1), b, (b+1).
  - Jika kondisi input mensyaratkan sejumlah n nilai maka uji dengan sejumlah (n-1), n dan (n+1) nilai.
  - Aplikasikan dua aturan sebelumnya pada kondisi output (buat tabel pengujian hasil outputnya untuk nilai maksimal dan minimal).
  - Jika struktur data internal dari program memiliki cakupan (misal: ukuran buffer, batas array) gunakan data input yang menguji batas cakupan.

Comparison Testing

- Pada beberapa aplikasi *reliability* dari sebuah perangkat lunak sangat penting.
- Redundansi perangkat keras dan perangkat lunak mungkin digunakan untuk meminimalisir kesalahan *(error)*.
- Untuk redundansi perangkat lunak, gunakan tim yang terpisah untuk mengembangkan setiap versi perangkat lunak yang independen.
- Uji setiap versi dengan data yang sama untuk memastikan semua versi menghasilkan keluaran yang sama.
- Jalankan semua versi dengan paralel dan perbandingan keluaran secara *real-time*.
- Walau hanya dijalankan sebuah versi pada akhirnya, untuk beberapa aplikasi yang penting dapat mengembangkan versi independen dan menggunakan *comparison testing* atau *back-to-back testing*.
- Ketika output dari versi berbeda, maka setiap versi diinvestigasi jika ada kemungkinan *defect*.
- Metode ini tidak untuk menemukan kesalahan dari spesifikasi.

Sample and Robustness Testing

- Sample Testing
  - Melibatkan beberapa nilai yang terpilih dari sebuah kelas ekivalen
  - Mengintegrasikan nilai pada kasus uji
  - Nilai-nilai yang terpilih mungkin dipilih dengan urutan tertentu atau interval tertentu
- Robustness Testing
  - Data input dipilih diluar spesifikasi yang telah didefinisikan
  - Tujuan dari pengujian ini adalah membuktikan bahwa tidak ada kesalahan jika masukan tidak valid

Behavior Testingn dan Performance Testing

- Behavior Testing
  - Hasil uji tidak dapat dievaluasi jika hanya melakukan pengujian sekali, tapi dapat dievaluasi jika pengujian dilakukan beberapa kali, misalnya pada pengujian struktur data stack
- Performance Testing
  - Mengevaluasi kemampuan program untuk beroperasi dengan benar dipandang dari sisi acuan kebutuhan misalnya: aliran data, ukuran pemakaian memori, kecepatan eksekusi, dll
  - Untuk mencari tahu beban kerja atau kondisi konfigurasi program
  - Spesifikasi mengenai performansi didefinisikan pada saat tahap spesifikasi atau desain
  - Dapat digunakan untuk menguji batasan lingkungan program

Requirement Testing

- Spesifikasi kebutuhan yang terasosiasi dengan perangkat lunak (input/output/fungsi/performansi) diidentifikasi pada tahap spesifikasi kebutuhan dan desain.
- Requirement testing melibatkan pembuatan kasus uji untuk setiap spesifikasi kebutuhan yang terkait dengan program
- Untuk memfasilitasinya, setiap spesifikasi kebutuhan bisa ditelusuri dengan kasus uji dengan menggunakan *traceability matrix*

Req.	Spec.	Pre-Design	Det-Design	CSU	Test Proc.	Test Rpt

Endurance Testing

- Endurance Testing melibatkan kasus uji yang diulang-ulang dengan jumlah tertentu dengan tujuan untuk mengevaluasi program apakah sesuai dengan spesifikasi kebutuhan.
- Contoh:
  - Untuk menguji keakuratan operasi matematika (floating point, rounding off, dll)
  - Untuk menguji manajemen sumber daya sistem *(resources)* (pembebasan sumber daya yang tidak benar, dll)
  - input/outputs (jika menggunakan *framework* untuk memvalidasi bagian input dan output)
- Spesifikasi kebutuhan pengujian didefinisikan pada tahap spesifikasi kebutuhan atau desain

Cause-effect Relationship Testing

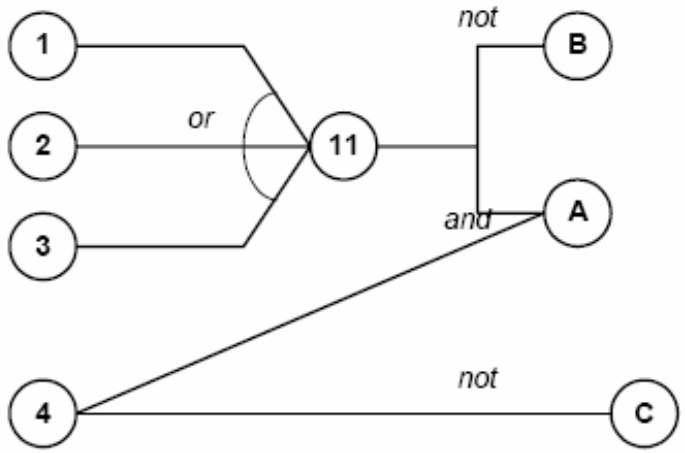
- Teknik ini merupakan suplemen dari *equivalence testing* dengan menyediakan cara untuk memilih kombinasi data input
- Melibatkan kondisi input *(cause)* dan kondisi output *(effect)* untuk mencegah pendefinisian kasus uji yang terlalu banyak
- Langkah
  - Bagi-bagi spesifikasi kebutuhan menjadi bagian yang memiliki kemungkinan kerja
  - Definisikan *cause* dan *effect* berdasarkan spesifikasi kebutuhan
  - Analisa spesifikasi kebutuhan untuk membuat hubungan logika
  - Tandai graf untuk jalur yang tidak mungkin berhubungan dengan kombinasu *cause/effect* sesuai dengan batasan spesifikasi kebutuhan
  - Ubah graf menjadi tabel keputusan
  - kolom --> test case
  - baris --> cause/effect
  - Ubah kolom dari tabel keputusan menjadi kasus uji

Contoh

- Sebuah mobil CITROEN dilambangkan dengan A, B atau C sebagai karakter pertama dan harus memiliki karakter X sebagai karakter kedua. Pesan M1 dan M2 harus dikirim pada pemasukan karakter pertama dan kedua jika terjadi kesalahan *(error)*. Jika masukan benar, maka akan dimasukkan ke basis data.
- Kondisi input (*causes*) dan kondisi output (*effect*) dapat dideterminasikan sebagai berikut:

Input states:
1. 1st character: A
2. 1st character: B
3. 1st character: C
4. 2nd character: X

Output states:
A. database insertion
B. message M1
C. message M2



	1	2	3	4	5
1	0	1	0	0	
2	0	0	1	0	
3	0	0	0	1	
4		1	1	1	0
A		1	1	1	
B	1				
C					1