Program Studi Teknik Informatika

FAKULTAS ILMU KOMPUTER
UNIVERSITAS DIAN NUSWANTORO

MATA KULIAH
SOFTWARE QUALITY & TESTING

Black Box Testing

Pengenalan Pengujian Black Box

- Pengujian Black Box berfokus pada persyaratan fungsional perangkat lunak
  - Penguji dapat memperoleh sekumpulan kondisi input yang akan sepenuhnya menjalankan semua persyaratan fungsional untuk suatu program.
- Pengujian Black Box bukanlah suatu alternatif untuk pengujian White Box, melainkan merupakan pendekatan pelengkap yang kemungkinan besar akan mengungkap kesalahan yang berbeda daripada metode White Box.

Pengenalan Pengujian Black Box

- Upaya untuk menemukan:
  - fungsi salah atau hilang
  - kesalahan antarmuka
  - kesalahan dalam struktur data atau akses database eksternal
  - kesalahan kinerja
  - kesalahan inisialisasi dan terminasi.
- Tes dirancang untuk menjawab pertanyaan berikut
  - Bagaimana validitas fungsional diuji?
  - Kelas masukan apa yang akan membuat kasus uji bagus?
  - Apakah sistem sangat sensitif terhadap nilai masukan tertentu?
  - Bagaimana batas-batas kelas data diisolasi?
  - Kecepatan data dan volume data apa yang dapat ditoleransi oleh sistem?
  - Apa pengaruh kombinasi data tertentu terhadap operasi sistem?

Penentuan Hasil yang Diharapkan

- Hasil yang diharapkan tidak dapat ditentukan dengan menjalankan sub program
- Hal ini ditentukan sebelum eksekusi pengujian, pada fase desain pengujian
- Biaya untuk menganalisis dan memprediksi perilaku yang diharapkan untuk setiap kasus uji (test case) dari setiap subprogram mungkin terlalu tinggi
    - Gunakan 'oracle'
        - Sesuatu yang memungkinkan untuk memprediksi hasil yang diharapkan dari pelaksanaan tes yang diterapkan ke subprogram dari subprogram tertentu, dengan data input khusus
        - Bisa berupa program, sekumpulan nilai, dll
- Berbagai kategori oracle mungkin berlaku untuk kasus-kasus sulit:
    - Hasil yang diharapkan diperoleh dengan menjalankan subprogram
        - Ini hanya berlaku jika eksekusi ditindaklanjuti dan diperiksa (untuk menganalisis nilai antara dan variabel internal)
        - Mampu membuktikan bahwa hasil yang diperoleh valid
        - Harus digunakan dengan hati-hati
    - Aplikasi dapat diturunkan dari aplikasi yang telah diuji sebelumnya
    - Menjalankan subprogram yang setara untuk diuji

Black Box Testing

- Equivalence Partitioning
- Boundary Value Analysis/Limit Testing
- Comparison Testing
- Sample Testing
- Robustness Testing
- Behavior Testing
- Requirement Testing
- Performance Testing
- Endurance Testing
- Cause-Effect Relationship Testing

Equivalence Partitioning (1)

- Bagilah domain masukan menjadi beberapa kelas data yang dapat dibuat kasus uji.
- Mencoba mengungkap kelas kesalahan.
- Berdasarkan kelas kesetaraan (equivalence classes) untuk kondisi input.
- Kelas kesetaraan mewakili sekumpulan status yang valid atau tidak valid
- Kondisi input bisa berupa nilai numerik tertentu, rentang nilai, sekumpulan nilai terkait, atau kondisi boolean.
- Kelas kesetaraan dapat ditentukan oleh:
  - Jika kondisi input menentukan rentang atau nilai tertentu, satu kelas ekivalen yang valid dan dua kelas ekivalen yang tidak valid ditentukan.
  - Jika kondisi input menentukan boolean atau anggota himpunan, satu kelas ekivalen yang valid dan satu kelas ekivalen yang tidak valid ditentukan.
- Uji kasus untuk setiap item data domain masukan yang dikembangkan dan dijalankan.

Contoh 1

- Persyaratan subprogram yang akan diuji (keadaan awal & akhir)
  - Subprogram mengambil input bilangan bulat dalam kisaran [-100,100]
  - Output subprogram adalah tanda nilai input (nilai 0 dianggap positif)
- Dua kelas kesetaraan yang valid
  - Rentang masukan [-100,0] yang berisi himpunan bagian dari integer, akan menghasilkan tanda negatif sebagai keluaran
  - Rentang masukan [0,100] akan menghasilkan tanda positif
- Dua kelas kesetaraan yang tidak valid
  - Bilangan bulat kurang dari -100
  - Bilangan bulat lebih dari -100

Aturan dalam Kesetaraan Kelas (Equivalence Classes)

- Jika kondisi mengenai data input menentukan rentang nilai (contoh: penghitung dapat mengambil nilai 1 hingga 999), tentukan persamaan yang valid. Kelas [1,999] dan dua kelas kesetaraan tidak valid (pencacah <1, dan pencacah> 99).
- Jika kondisi mengenai data masukan menentukan jumlah nilai (sesuatu dapat berisi 1 hingga 6 nama), tentukan kelas kesetaraan yang valid dan dua kelas kesetaraan yang tidak valid (tidak ada nama dan lebih dari 6 nama)
- Jika kondisi mengenai data masukan menentukan satu set nilai dan unit software komputer yang diuji secara berbeda (contoh: BUS, TAXI, TRUCK, TRAIN), tentukan kelas ekivalensi yang valid untuk setiap nilai dan kelas ekivalensi yang tidak valid (contoh: SEPEDA MOTOR).
- Jika kondisi masukan diekspresikan oleh kalimat yang berisi 'harus' (contoh karakter pertama harus berupa huruf), tentukan kelas kesetaraan yang valid (huruf) dan kelas kesetaraan yang tidak valid (bukan huruf)
- Jika ada alasan untuk meyakini bahwa unit software komputer yang akan diuji tidak memproses elemen kelas ekivalen dengan cara yang sama, uraikan kelas kesetaraan menjadi kelas kesetaraan yang lebih kecil (berdasarkan pendekatan subprogram logis, di tingkat algoritme).
    - Jika tidak demikian, ini harus dipertimbangkan dalam mendefinisikan kelas kesetaraan

Aturan dalam Kesetaraan Kelas

- Jika sub program dari unit software komputer yang akan diuji memiliki beberapa masukan, kelas ekivalen atau proses definisi kasus uji mungkin atau mungkin tidak memperhitungkan masalah kombinasi.
  - Untuk menguji subprogram yang, memiliki dua input independen (A dan B), dua kelas ekivalensi yang valid (A1, A2, B1 dan B2) ditentukan untuk setiap input
  - Subprogram dapat diuji dengan:
    - Dua kasus uji (kasus pertama menentukan nilai masukan A1, dan nilai masukan B1, dan nilai kedua A2 dan B2) dan mengingat bahwa pengujian ini sudah cukup
    - Atau dengan empat kasus uji (A1 dan B1, A1 dan B2, A2 dan B1, A2 dan B2) jika dianggap perlu.
- Untuk menguji subprogram yang memiliki dua input independen, kelas ekivalensi yang tidak valid, ditentukan untuk setiap input
- Tulis kasus uji untuk setiap kelas kesetaraan yang tidak valid
  - Contoh: persyaratan menentukan untuk memasukkan jenis mobil, dan nomor mobil "
  - Dalam kasus uji di mana data masukan adalah dua nilai yang salah 'XYZ', 0 program mungkin tidak akan memeriksa jumlah item (0), dan akan berhenti dengan pesan 'XYZ' bukan jenis mobil
- Untuk menguji subprogram yang memiliki beberapa input dependen, kelas ekivalen yang akan ditentukan harus memperhitungkan kombinasi input

Contoh 2

- Persyaratan
  - Diketahui tiga nilai, mewakili panjang sisi segitiga, tentukan apakah sama sisi, sama kaki dan tak sama panjang
- Tiga kelas kesetaraan yang valid
  - 3 nilai positif sama
  - 3 nilai positif dan 2 yang sama dan jumlah dua lebih besar dari nilai ketiga
  - 3 nilai positif dan jumlah dua nilai lebih besar dari nilai ketiga
- Dua kelas kesetaraan yang tidak valid
  - 2 nilai positif dan negatif / nol
  - 3 nilai positif sehingga jumlah keduanya sama / kurang

Boundary Value Analysis (BVA)/ Limit Testing (2)

- Analisis Nilai Batas/ Pengujian Batas
- Sejumlah besar kesalahan cenderung terjadi di batas-batas domain masukan.
- BVA mengarah pada pemilihan kasus uji yang menggunakan nilai batas.
- BVA melengkapi partisi kesetaraan. Daripada memilih elemen apa pun dalam kelas ekivalensi, pilih elemen yang ada di '' tepi 'kelas.
- Contoh:
  - Untuk rentang nilai yang dibatasi oleh a dan b, uji (a-1), a, (a + 1), (b-1), b, (b + 1).
  - Jika kondisi input menentukan sejumlah nilai n, uji dengan nilai input (n-1), n dan (n + 1).
  - Terapkan 1 dan 2 ke kondisi keluaran (misalnya, buat tabel dengan ukuran minimum dan maksimum).
  - Jika struktur data program internal memiliki batasan (mis., Ukuran buffer, batas tabel), gunakan data input untuk melatih struktur pada batasan.

Comparison Testing (3)

- Dalam beberapa aplikasi, keandalan software sangat penting.
  - Avionik pesawat terbang, kontrol pembangkit listrik tenaga nuklir
- Perangkat keras dan perangkat lunak yang redundan dapat digunakan untuk meminimalkan kesalahan.
- Untuk software redundan, gunakan tim terpisah untuk mengembangkan versi independen perangkat lunak.
- Uji setiap versi dengan data uji yang sama untuk memastikan semua memberikan keluaran yang identik.
- Jalankan semua versi secara paralel dengan perbandingan hasil waktu nyata.
- Bahkan jika hanya akan menjalankan satu versi dalam sistem final, untuk beberapa aplikasi kritis dapat mengembangkan versi independen dan menggunakan perbandingan pengujian atau pengujian back-to-back.
- Ketika keluaran versi berbeda, masing-masing diselidiki untuk menentukan apakah ada cacat.
- Metode tidak menemukan kesalahan dalam spesifikasi.

Sample Testing (4)

- Pengujian sampel melibatkan pemilihan sejumlah nilai dari kelas ekivalen data masukan
- Mengintegrasikan nilai ke dalam kasus uji
- Nilai-nilai ini dapat dipilih pada interval konstan atau variabel

Robustness (Kekokohan) Testing (5)

- Data dipilih di luar rentang yang ditentukan oleh persyaratan
- Tujuan dari pengujian ini adalah untuk membuktikan bahwa tidak ada peristiwa bencana yang dapat dihasilkan oleh pengenalan item data masukan yang abnormal

Behavior Testing (6)

- Tes perilaku adalah tes yang hasilnya diperoleh tidak dapat dievaluasi setelah eksekusi tunggal dari unit software komputer, tetapi hanya dapat dievaluasi setelah serangkaian panggilan terkait ke subprogram (beberapa panggilan ke subprogram tertentu, panggilan ke beberapa subprogram)

Requirement Testing (7)

- Persyaratan yang terkait dengan perangkat lunak (input / output / fungsi / kinerja) diidentifikasi selama spesifikasi perangkat lunak dan aktivitas desain.
- Pengujian persyaratan melibatkan pembuatan kasus uji untuk setiap persyaratan yang berkaitan dengan unit software komputer (Computer Software Unit - CSU).
- Untuk memfasilitasi penerapan pengujian tersebut, setiap persyaratan dapat ditelusuri ke kasus pengujian akhir melalui matriks penelusuran (Traceability Matrix)

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

Performance Testing (8)

- Pengujian kinerja mengevaluasi kemampuan unit software komputer untuk beroperasi dengan benar berkenaan dengan persyaratan terkait, misalnya:
  - aliran data, ukuran memori, waktu eksekusi, dll
- Untuk memenuhi beban kerja atau kondisi konfigurasi tertentu, asalkan, bagaimanapun, bahwa persyaratan ini dapat tercermin di tingkat unit software komputer.
- Persyaratan ini ditetapkan selama spesifikasi atau aktivitas desain
- Pengujian kinerja juga dapat digunakan untuk mengukur dan mengeksplorasi batasan kinerja unit software komputer

Endurance Testing (9)

- Uji Ketahanan melibatkan pengulangan kasus uji tertentu beberapa kali untuk mengevaluasi kemampuan unit software komputer untuk memenuhi persyaratan.
- Sebagai contoh:
  - keteraturan dalam akurasi operasi matematika tertentu (floating point, pembulatan, dll)
  - manajemen sumber daya sistem (pelepasan sumber daya yang salah, dll)
  - masukan / keluaran (dalam rangka memvalidasi lapisan masukan / keluaran)
- Persyaratan ini ditetapkan selama spesifikasi atau aktivitas desain

Cause-effect Relationship Testing (10)

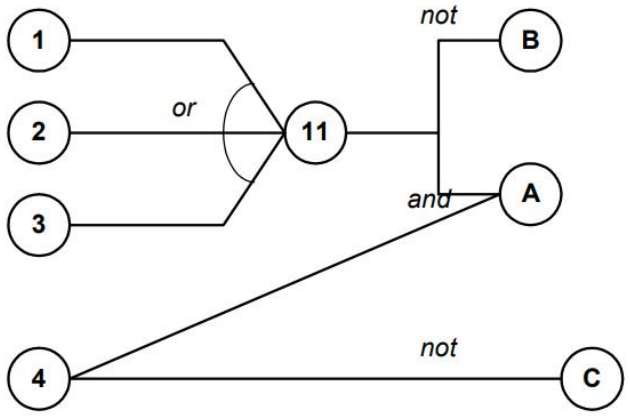
- Teknik ini melengkapi pengujian ekivalensi dengan menyediakan cara untuk menentukan dan memilih kombinasi data masukan
- Melibatkan representasi status input (penyebab) dan status output (efek) yang disimpulkan dari persyaratan fungsional yang terkait dengan unit software komputer dalam bentuk grafik yang diwakili oleh hubungan logis (tabel keputusan), untuk menghindari terlalu banyak kasus uji yang ditentukan
- Langkah-langkahnya adalah
  - Pecahkan persyaratan menjadi subset yang memungkinkan untuk bekerja
  - Definisikan sebab dan akibat berdasarkan kebutuhan
  - Menganalisis persyaratan untuk membuat hubungan yang logis
  - Sorot grafik, ketidakmungkinan kombinasi sebab / akibat karena kendala dari persyaratan
  - Ubah grafik menjadi tabel keputusan
    - Kolom -> kasus uji
    - Baris -> sebab / akibat
  - Ubah kolom tabel keputusan menjadi kasus uji

Contoh

- Mobil CITROEN akan dikenali dengan huruf A, B atau C, dan memiliki huruf X sebagai karakter kedua. Pesan M1 dan M2 harus dikirim jika terjadi kesalahan pada karakter pertama atau kedua. Jika pengenalnya benar, itu dimasukkan ke dalam database.
- Status input (penyebab) dan status output (efek) dapat ditentukan:

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

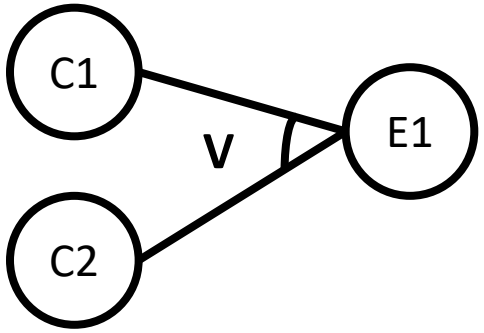


	1	2	3	4	5
1	0	1	0	0	
2	0	0	1	0	
3	0	0	0	1	
4		1	1	1	0
A		1	1	1	
B	1				
C					1

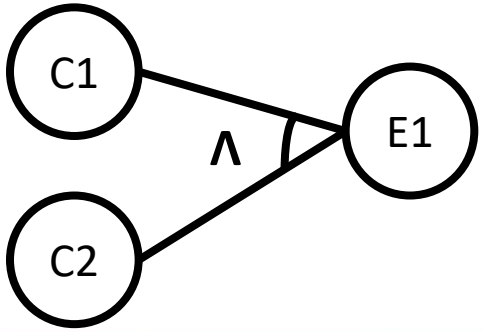
Grafik Sebab dan Akibat (Cause and Effect Graph) #1

Notasi yang Digunakan:

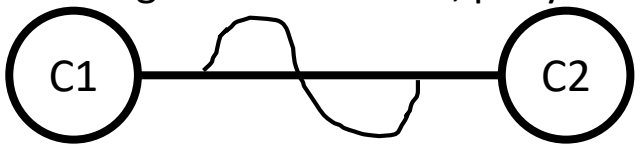
AND - Agar efek E1 benar, penyebab C1 dan C2 harus benar



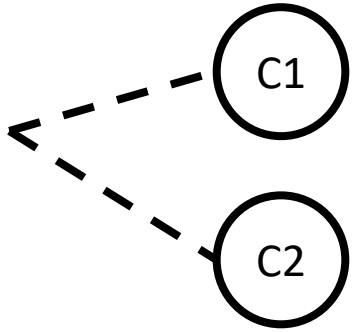
OR - Agar efek E1 benar, salah satu penyebab C1 atau C2 harus benar



NOT - Agar efek E1 benar, penyebab C1 harus salah



MUTUALLY EXCLUSIVE - Ketika hanya satu penyebab yang akan benar



- Gambar grafik Sebab dan Akibat berdasarkan kebutuhan / situasi.
- Setelah grafik sebab dan akibat diberikan, gambar tabel keputusan untuk menggambar kasus uji.

Contoh Kasus

“Cetak Pesan" adalah perangkat lunak yang membaca dua karakter. Tergantung pada nilainya, pesan akan dicetak.

- Karakter pertama harus berupa "A" atau "B".
- Karakter kedua harus berupa digit.
- Jika karakter pertama adalah "A" atau "B" dan karakter kedua adalah digit, maka file harus diperbarui.
- Jika karakter pertama salah (bukan "A" atau "B"), pesan X harus dicetak.
- Jika karakter kedua salah (bukan digit), pesan Y harus dicetak.

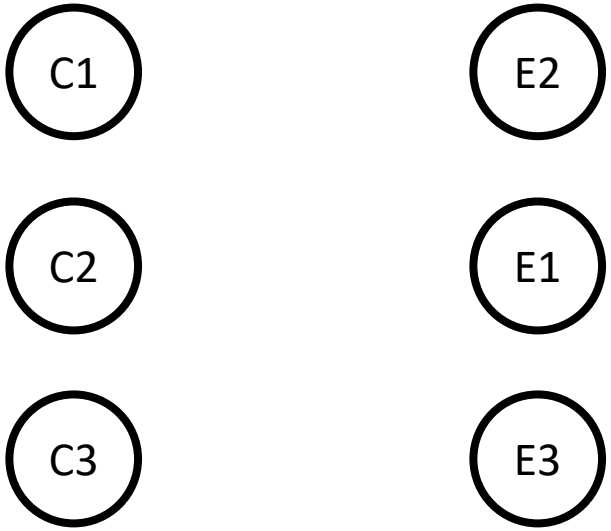
Grafik Sebab dan Akibat “Cetak Pesan”

“Cetak Pesan" adalah perangkat lunak yang membaca dua karakter. Tergantung pada nilainya, pesan akan dicetak.

- Karakter pertama harus berupa "A" atau "B".
- Karakter kedua harus berupa digit.
- Jika karakter pertama adalah "A" atau "B" dan karakter kedua adalah digit, maka file harus diperbarui.
- Jika karakter pertama salah (bukan "A" atau "B"), pesan X harus dicetak.
- Jika karakter kedua salah (bukan digit), pesan Y harus dicetak.

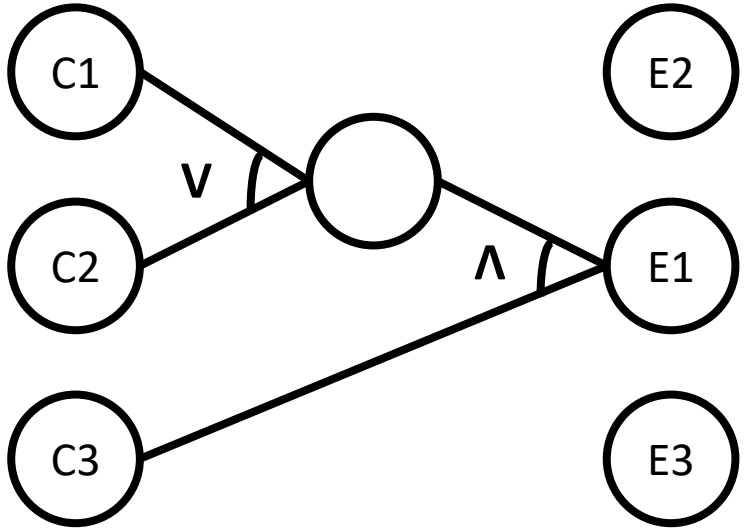
Solusi:
Penyebab dari situasi ini adalah:
C1 - Karakter pertama adalah A
C2 - Karakter pertama adalah B
C3 - Karakter kedua adalah digit

Efek (hasil) untuk situasi ini adalah:
E1 - Perbarui file
E2 - Cetak pesan "X"
E3 - Cetak pesan "Y"



Grafik Sebab dan Akibat “Cetak Pesan”

- Selalu beralih dari Efek ke Sebab (kiri ke kanan). Artinya, untuk mendapatkan efek “E”, penyebab apa yang harus benar.
- Dalam contoh ini, mari kita mulai dengan Efek E1.
- Efek E1 untuk memperbarui file. File diperbarui saat:
  - Karakter pertama adalah "A" dan karakter kedua adalah digit
  - Karakter pertama adalah "B" dan karakter kedua adalah digit
  - Karakter pertama bisa jadi "A" atau "B" dan tidak bisa keduanya.
- Lalu mari kita taruh 3 poin ini dalam bentuk simbolik:
- Agar E1 benar - berikut ini penyebabnya:
  - C1 dan C3 harus benar
  - C2 dan C3 harus benar
  - C1 dan C2 tidak bisa benar bersama. Ini berarti C1 dan C2 saling eksklusif.

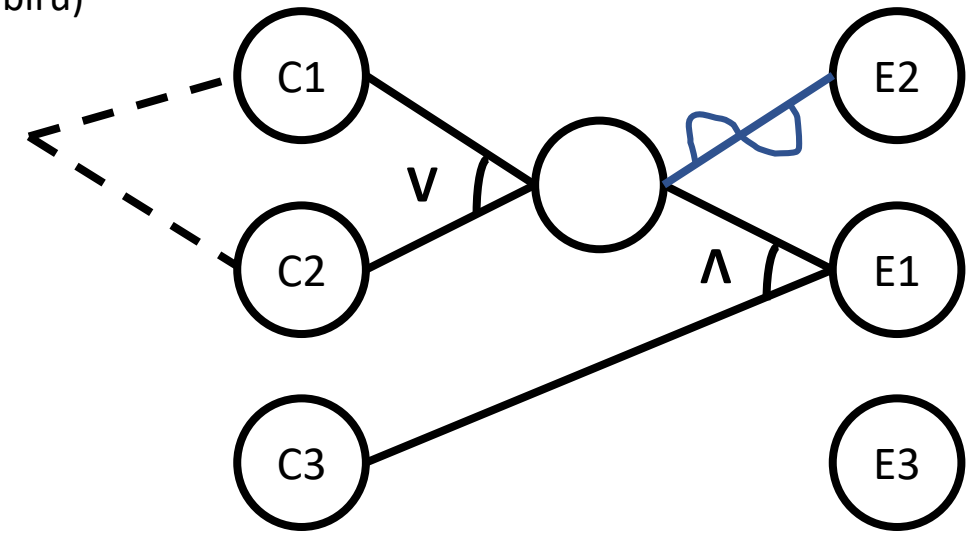


- Sesuai diagram di atas, agar E1 benar kondisinya adalah
- (C1 **∨** C2) **∧** C3
  Lingkaran di tengah hanyalah interpretasi dari titik tengah agar grafik tidak terlalu berantakan.

Grafik Sebab dan Akibat “Cetak Pesan”

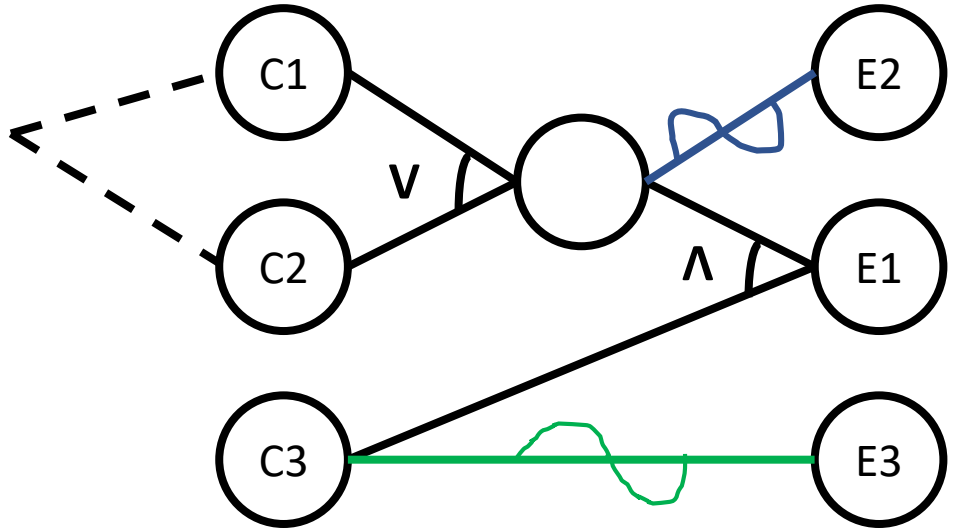
Mari pindah ke Efek E2:

- E2 menyatakan mencetak pesan "X". Pesan X akan dicetak jika karakter Pertama bukan A atau B.
- Ini berarti Efek E2 akan berlaku jika C1 ATAU C2 tidak valid. Jadi grafik untuk Efek E2 ditampilkan sebagai (Dalam garis biru)



Grafik Sebab dan Akibat “Cetak Pesan”

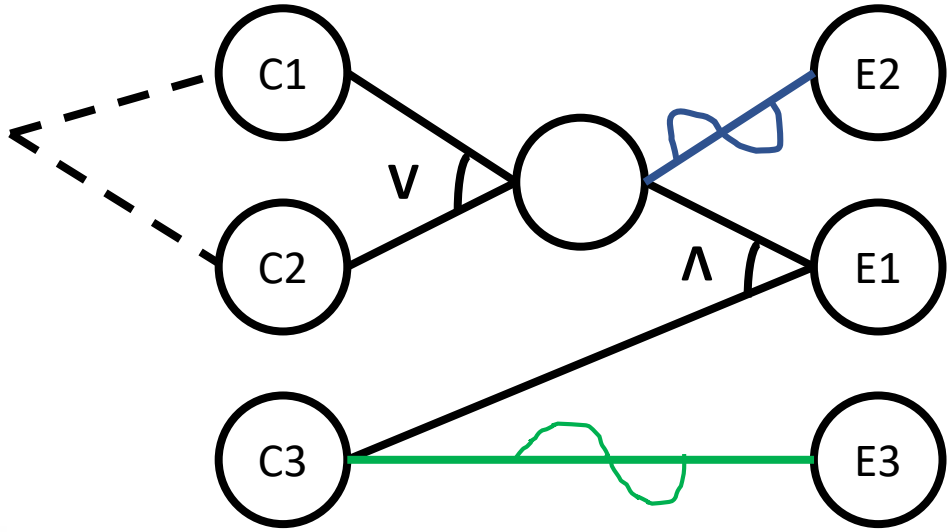
Untuk Efek E3.
E3 menyatakan mencetak pesan "Y". Pesan Y akan dicetak ketika karakter Kedua salah. Berarti Efek E3 akan berlaku jika C3 tidak valid. Jadi grafik untuk Efek E3 ditampilkan sebagai (Dalam garis Hijau)



Tabel Keputusan Berdasarkan Grafik Sebab Akibat

- Pertama, tuliskan Sebab dan Akibat dalam satu kolom
- Mulailah dengan Efek E1. Agar E1 benar, kondisinya adalah (C1 ∨ C2) ∧ C3.
- Representasikan True sebagai 1 dan False sebagai 0.

- Sekarang untuk E1 menjadi "1" (benar), kita memiliki dua kondisi di bawah ini -
- C1 DAN C3 akan menjadi true
- C2 DAN C3 akan menjadi true

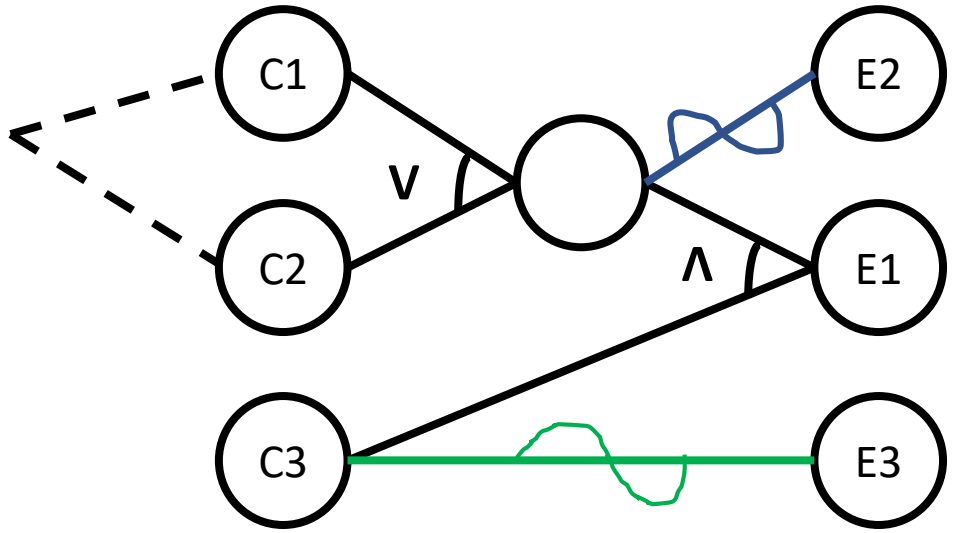


Aksi	
C1	
C2	
C3	
E1	1
E2	
E3	

Aksi		
C1	1	0
C2	0	1
C3	1	1
E1	1	1
E2	0	0
E3	0	0

Tabel Keputusan Berdasarkan Grafik Sebab Akibat

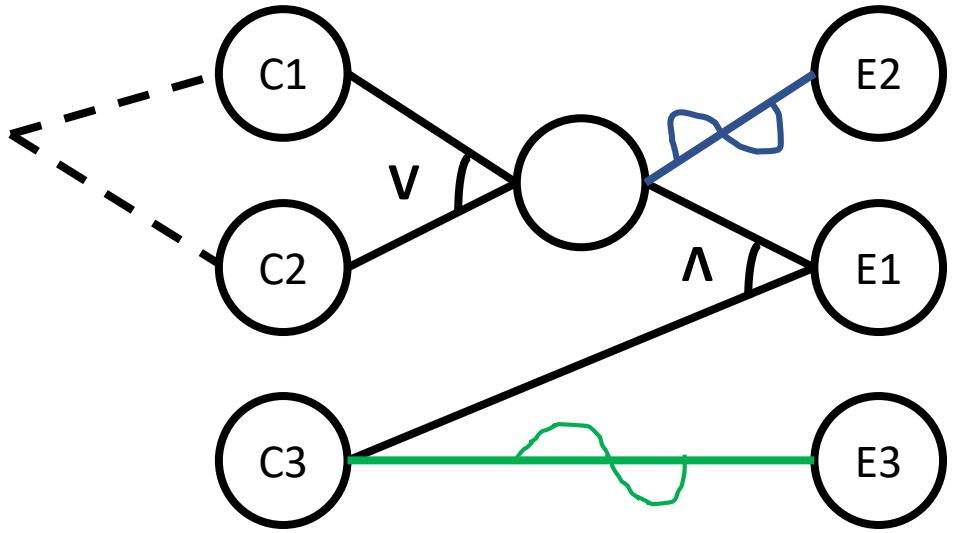
- Untuk E2 menjadi True, baik C1 atau C2 harus False



Aksi				
C1	1	0	0	0
C2	0	1	0	0
C3	1	1	0	1
E1	1	1	0	0
E2	0	0	1	1
E3	0	0	0	0

Tabel Keputusan Berdasarkan Grafik Sebab Akibat

- Agar E3 benar, C3 harus salah.



Aksi						
C1	1	0	0	0	1	0
C2	0	1	0	0	0	1
C3	1	1	0	1	0	0
E1	1	1	0	0	0	0
E2	0	0	1	1	0	0
E3	0	0	0	0	1	1

Menulis Test Case Dari Tabel Keputusan

Di bawah ini adalah contoh kasus uji untuk Kasus Uji 1 (TC1) dan Kasus Uji 2 (TC2).

Aksi	TC1	TC2	TC3	TC4	TC5	TC6
C1	1	0	0	0	1	0
C2	0	1	0	0	0	1
C3	1	1	0	1	0	0
E1	1	1	0	0	0	0
E2	0	0	1	1	0	0
E3	0	0	0	0	1	1

TC ID	TC Name	Description	Steps	Expected Result
TC1	TC1_FileUpdateScenario1	Validasi bahwa sistem memperbarui file ketika karakter pertama adalah A dan karakter kedua adalah digit	1. Buka aplikasi 2. Masukkan karakter pertama berupa huruf “A” 3. Masukkan karakter kedua berupa digit	File diperbarui
TC2	TC1_FileUpdateScenario2	Validasi bahwa sistem memperbarui file ketika karakter pertama adalah B dan karakter kedua adalah digit	1. Buka aplikasi 2. Masukkan karakter pertama berupa huruf “B” 3. Masukkan karakter kedua berupa digit	File diperbarui

Referensi

- Bayu Hendradjaya - http://www.if.itb.ac.id/~bayu
- Cause And Effect Graph – Dynamic Test Case Writing Technique For Maximum Coverage With Fewer Test Cases https://www.softwaretestinghelp.com/cause-and-effect-graph-test-case-writing-technique/

Tugas Kelompok

- Lakukan pengujian Black Box bersama kelompok Anda, pilihlah salah satu teknik pengujian yang ada di slide sebelumnya!
- Tuliskan kasus uji sesuai dengan aplikasi atau software yang Anda uji!
- Software yang diuji adalah software yang telah Anda kembangkan sendiri!
- Tugas diketik rapi, test case dibuat dalam bentuk tabel uji.
- Minggu depan dipresentasikan hasil pengujian dan aplikasi/ software yang Anda uji.