STRUKTURISASI KEBUTUHAN SISTEM: PEMBUATAN MODEL LOGIKA

Sebelumnya, kita telah mempelajari bagaimana proses yang merubah data menjadi informasi merupakan bagian penting dari sistem informasi. Sebaik-baiknya diagram aliran data atau DFD mengidentifikasi proses, DFD tidak cukup baik untuk menunjukan logika yang ada dalam suatu proses. Bahkan, DFD pada tingkat primitive tidak menunjukan langkah-langkah pemrosesan paling mendasar. Apa yang terjadi dalam suatu proses ? Bagaimana data masukan dirubah menjadi informasi keluaran ? Karena DFD dirancang *tidak* untuk menunjukan logika pemrosesan secara terinci, anda harus membuat model logika proses menggunakan teknik lain. Bahasan ini membicarakan tentang teknik-teknik yang anda pakai untuk membuat model logika proses.

Pertama, anda akan dikenalkan dengan Bahasa Inggris Terstruktur, versi bahasa Inggris alami yang telah dimodifikasi yang bermanfaat dalam merepresentasikan logika proses dalam sistem informasi. Anda bisa memakai Bahasa Inggris Terstuktur untuk merepresentasikan tiga pernyataan dasar yang diperlukan dalam pemrograman terstruktur, yaitu: choice, repetition, dan sequence.

Pembuatan Model Logika

Pada bahasan sebelumnya anda telah mempelajari bagaimana kebutuhan-kebutuhan untuk sistem informasi dikumpulkan. Analis membuat struktur informasi kebutuhan tersebut ke dalam diagram aliran data atau DFD yang memeragakan aliran data masuk, mengalir dan akhirnya keluar dari suatu proses dalam sistem informasi.

Meski, DFD, merupakan teknik berguna dan baik, DFD tidak cukup memadai untuk memperagakan semua kompleksitas dari sistem informasi. Meski dekomposisi memungkinkan anda merepresentasi proses sampai ke tingkat terinci, nama proses tidak itu sendiri tidak cukup menggambarkan apa yang proses lakukan dan bagaimana melakukannya. Dengan alasan tersebut di atas anda harus merepresentasi logika yang ada dalam simbol-simbol proses di DFD dengan teknik model yang lain.

Pembuatan Model Logika dengan Bahasa Inggris Terstruktur

Anda harus memahami lebih daripada sekedar aliran yang masuk, mengalir, dan keluar dari sistem informasi. Anda juga harus

memahami apa yang dilakukan oleh setiap proses yang teridentifikasi dan bagaimana setiap proses tersebut melakukannya.

Bahasa Inggris Terstruktur, untuk selanjutnya singkat BIT, merupakan satu bentuk modifikasi dari Bahasa Inggris yang dipergunakan untuk menetapkan isi dari kotak hitam proses dalam suatu DFD. Ia berbeda dengan bahasa Inggris umumnya dalam arti bahwa BIT memakai sebagian kosa kata bahasa Inggris untuk menyatakan proses pada sistem informasi.

Kata kerja yang dipakai untuk nama proses dalam suatu DFD, juga dapat dipakai dalam BIT. Kata kerja ini diantaranya, read, write, print, sort, move, merge, add, subtract, multiply, dan devide. BIT juga menggunakan frasa kata benda untuk mendeskripsikan struktur data seperti nama-pelanggan dan alamat-pelanggan.

Tidak seperti bahasa Inggris alami, BIT tidak menggunakan kata sifat dan kata keterangan. Maksud utama pemakaian BIT ialah menjelaskan proses secara ringkas, yaitu agar pemakai dan pemrogram dapat lebih mudah membaca dan memahaminya. Karena tidak ada versi standar maka tiap analis menggunakan dialek BIT sendiri-sendiri.

Bit dapat merepresentasikan ketiga proses yang umumnya ada dalam pemrograman terstruktur, yaitu: sekuen, pernyataan bersyarat, dan perulangan. Sekuen tidak memerlukan struktur khusus tetapi direpresentasikan dengan satu rangkaian pernyataan mengikuti pernyataan lainnya.

Pernyataan-pernyataan bersyarat dapat direpresentasikan dengan struktur seperti berikut:

```
BEGIN IF
        IF Kuantitas-Stok =< Kuantitas-Pesan-Minimum
        THEN GENERATE pesanan-baru
        ELSE tak-lakukan-apapun
ENDIF
```

Tipe lain dari pernyataan bersyarat adalah pernyataan case dimana terdapat banyak aksi berbeda yang program dapat ikuti, tetapi hanya satu yang terpilih. Pernyataan case dapat direpresentasikan sebagai berikut:

```
READ Kuantitas-Stok
```

```
SELECT CASE
        CASE 1 (Kuantitas-Stok > Kuantitas-Pesan-Minimum)
                DO tak-lalukan-apapun
        CASE 2 (Kuantitas-Stok = Kuantitas-Pesan-Minimum)
                DO tak-lakukan-apapun
        CASE 3 (Kuantitas-Stok < Kuantitas-Pesan-Minimum)
                GENERATE pesanan-baru
        CASE 4 (Stok habis)
                INITIATE rutin-pesanan-darurat
AND CASE
```

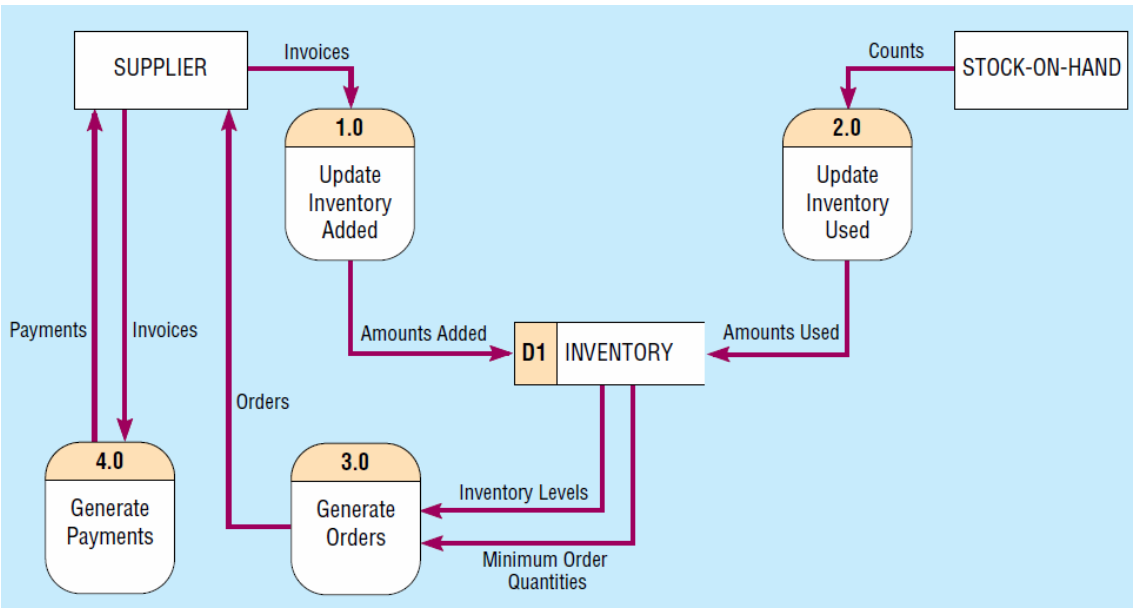
Repetisi dapat berbentuk Do-Until loop atau Do-While loop. Do-Until loop bisa berbentuk seperti berikut ini:

```
 DO
        READ record Inventori
        BEGIN IF
                IF Kuantitas-Stok < Kuantitas-Pesan-Minimum
                THEN GENERATE pesanan-baru
                ELSE DO tak-lakukan-apapun
        END IF
UNTIL End-of-file
```

Do-While loop bisa direpresentasikan sebagai berikut:

```
READ record-Inventori
WHILE NOT End-of-file DO
        BEGIN IF
                IF Kuantitas-Stok < Kuantitas-Pesan-Minimum
                THEN GENERATE pesanan-baru
                ELSE DO tak-lalukan-apapun
        END IF
END DO
```

Kita lihat satu contoh bagaimana BIT akan merepresentasikan logika dari sebagian proses yang telah diidentifikasi pada sistem kontrol inventori dari restoran cepat saji, seperti ditunjukan dalam Gambar 9-2 di bawah ini.



Gambar 9-2: Logika DFD Sistem Kontrol Inventori saat ini pada Restoran Cepat Saji.

Ada empat proses diilustrasikan pada Gambar 9-2, Membarui Ditambah Inventori, Membarui Inventori Dipakai, Membuat Pesanan, Membuat Pembayaran. Representasi BIT dari setiap proses ditunjukan dalam Gambar 9-3

Proses 1.0: Memperbaharui Inventori Penambahan

DO

READ rekord-Item-invoice berikutnya

FIND Rekord-inventori yang sama

ADD Kuantitas-ditambahkan dari Rekord-item-invoice ke Kuantitas-stok pada Record-inventori

UNTIL End-of-file

Proses 2.0: Memperbaharui Inventori Terpakai

DO

READ Rekord-item-stok berikutnya

FIND Rekord-inventori yang sama

SUBTRACT Kuantitas-terpakai pada Rekord-item-stok dari Kuantitas-stok pada Rekord-inventori

```
UNTIL End-of-file
```

```
Proses 3.0: Membuat Pesanan
DO
        READ Rekord-inventori berikutnya
        BEGIN IF
                IF Kuatitas-stok kurang dari Kuantitas-pesanan-minimum
                THEN GENERATE Pesanan
        END IF
UNTIL End-of-file
```

```
Proses 4.0: Membuat Pembayaran
READ Tanggal-hari-ini
DO
        SORT Rekord-invoice menurut Tanggal
        READ record-invoice berikutnya
        BEGIN IF
                IF Tanggal adalah 30 hari atau lebih besar dari Tanggal-hari-ini
                THEN GENERATE Payments
        END IF
UNTIL End-of-file
```

Amati bahwa dalam versi BIT ini nama file dihubungkan tanda penghubung "-" serta nama file dan nama variable memakai huruf besar. Istilah logika perbandingan, seperti lebih besar, lebih kecil, dan semacamnya ditandai dengan symbol aritmatika tidak dengan ejaan pengucapan.

Juga amati bagaimana pendeknya spesifikasi dalam BIT. Semua spesifikasi di atas mendeskripsikan proses tingkat-0. Spesifikasi akhir menjelaskan logika dalam DFD pada tingkat terendah. Dengan membaca deskripsi proses pada Gambar 9-3, jelas bahwa diperlukan lebih terinci lagi untuk melaksanakan proses-proses sepetti tersebut di atas. Pembuatan proses memakai BIT dapat membantu anda menentukan apakah suatu DFD perlu lebih lanjut dilakukan dekomposisi.

Pembuatan Model Logika Dengan Tabel Keputusan

Bahasa Inggris Terstruktur dipakai untuk merepresentasikan logika yang terkandung dalam proses sistem informasi. Seringkali, logika proses dapat menjadi kompleks. Jika terdapat beberapa kondisi dan kombinasi kondisi ini mendikte beberapa aksi yang harus dilaksanakan maka BIT tidak cukup memadai untuk merepresentasikan logika yang ada dalam pilihan yang kompleks tersebut. Hal ini bukan karena BIT tidak dapat merepresentasikan logika ini tetapi BIT menjadi lebih sulit dimengerti dan diverifikasi.

Tabel keputusan adalah suatu diagram dari logika proses dimana logikanya cukup rumit. Semua kemungkinan pilihan, kondisi dan aturan direpresentasi dalam bentuk tabel, seperti ditunjukan dalam Gambar 9-4.

	Conditions/ Courses of Action	Rules					
		1	2	3	4	5	6
Condition Stubs	Employee type	S	H	S	H	S	H
	Hours worked	<40	<40	40	40	>40	>40
Action Stubs	Pay base salary	X		X		X	
	Calculate hourly wage		X		X		X
	Calculate overtime						X
	Produce Absence Report		X				

Gambar 9-4: Tabel Keputusan untuk Sistem Penggajian

Tabel keputusan dalam Gambar 9-4 memeragakan logika sistem penggajian. Ada tiga bagian dari table ini, yaitu: stub kondisi, stub aksi, dan aturan. Pada Gambar 9-4 terdapat dua stub kondisi untuk tipe pegawai dan jam bekerja. Tipe pegawai memiliki dua nilai, yaitu “S” untuk pegawai tetap dan “H” untuk pegawai honorer. Jam bekerja memiliki tiga nilai, yaitu kurang dari 40 jam, tepat 40 jam, dan lebih dari 40 jam. Stub aksi berisi semua kemungkinan aksi yang disebabkan oleh gabungan nilai dari stub kondisi. Ada empat kemungkinan aksi dalam tanel ini, Membayar gaji pokok, Menghitung upah per jam, Menghitung lembur, Membuat Laporan Absensi.

Untuk membaca aturan-aturan ini, mulai dengan membaca nilai kondisi pada kolom 1. Tipe pegawai ‘S’ dan jam kerja ‘<40’. Ketika kedua kondisi ini terjadi maka system penggajian menghitung pembayaran gaji pokok. Kolom berikutnya, nilai ‘H’ dan ‘<40’ berarti

bahwa pegawai honorer bekerja kurang dari 40 jam. Pada keadaan ini system penggajian menghitung upah menurut jam kerja dan membuat laporan kehadiran. Demikain untuk kolom lainnya.

Aturan 1, 3, dan 5 memiliki aksi yang sama dan menentukan hanya pegawai tetap. Kondisi jam bekerja tidak mempengaruhi hasil untuk aturan 1, 3, dan 5. Untuk aturan-aturan seperti ini, jam bekerja, dikatakan kondisi takberbeda, dalam arti nilai-nilainya tidak mempengaruhi aksi yang dilakukan.

Karena kondisi takberbeda ini, kita dapat mengurangi jumlah aturan dengan mengabungkan aturan 1, 3, dan 5 menjadi hanya satu aturan, seperti ditunjukan dalam Gambar 9-5 di bawah.l

Conditions/ Courses of Action	Rules			
	1	2	3	4
Employee type	S	H	H	H
Hours worked	–	<40	40	>40
Pay base salary	X			
Calculate hourly wage		X	X	X
Calculate overtime				X
Produce Absence Report		X		

Gambar 9-5: Tabel Keputusan yang direduksi untuk Sistem Penggajian

Prosedur dasar dalam pembuatan table keputusan adalah sebagai berikut:

1. *Beri nama kondisi dan nilai-nilai yang mungkin untuk setiap kondisi.* Tentukan semua kondisi yang relevan dengan masalah anda. Selanjutnya tentukan semua nilai yang mungkin untuk tiap kondisi. Untuk kondisi tertentu nilainya mungkin hanya berupa ‘ya’ atau ‘tidak’. Kondisi dengan nilai seperti ini disebut entri terbatas. Untuk kondisi-kondisi yang lain seperti Gambar 9-4 dan 9-4 memiliki nilai lebih banyak disebut entri diperluas.

2. *Beri nama semua aksi yang mungkin*. Maksud pembuatan tabel keputusan ialah untuk menentukan satu rangkaian aksi yang tepat dengan sekumpulan kondisi yang telah diketahui.

3. *Daftar semua kemungkinan aturan*. Ketika pertama kali anda membuat satu tabel keputusan, anda harus membuat satu kumpulan aturan yang lengkap. Setiap kombinasi kondisi yang mungkin harus dinyatakan, meski pada akhirnya menghasilkan sejumlah aturan yang rangkap atau tidak nalar. Hanya dengan mendaftar setiap aturan dapat dipastikan tidak ada kemungkinan atruran yang terlewati. Untuk menentukan jumlah semua aturan yang mungkin kalikan jumlah nilai untuk setiap kondisi dengan jumlah nilai setiap kondisi lainnya. Pada tabel keputusan Gambar 9-4 kita memiliki dua kondisi masing-masing dengan jumlah nilai 2 dan 3 sehingga jumlah aturan yang mungkin adalah 2x3=6.

4. *Tentukan aksi unutk setiap aturan*. Bila semua aturan yang mungkin telah diidentifikasi, selanjutnya anda menentukan satu aksi untuk setiap aturan. Dalam contoh, kita telah dapat menemukan semua aksi dan semua aksi tersebut nalar. Jika suatu aksi tidak nalar maka anda dapat membuat satu baris 'Tak mungkin' pada stub atau bagian aksi pada tabel.

5. *Sederhanakan tabel keputusan*. Buat tabel keputusan sesederhana mungkin dengan menghilangkan aksi-aksi yang tidak mungkin. Konsultasikan dengan pemakai mengenai aturan-aturan dimana aksi sistem tidak jelas serta menentukan aksi atau menghilangkan aturan. Cari pola-pola aturan khususnya untuk kondisi takberbeda. Kita telah dapat mengurangi jumlah aturan dalam contoh sistem penggajian dari enam menjadi empat.

Selain BIT dan table keputusan masih terdapat teknik lain untuk memperagakan logika proses, yaitu pohon keputusan.

Pembuatan Model Logika Dengan Pohon Keputusan

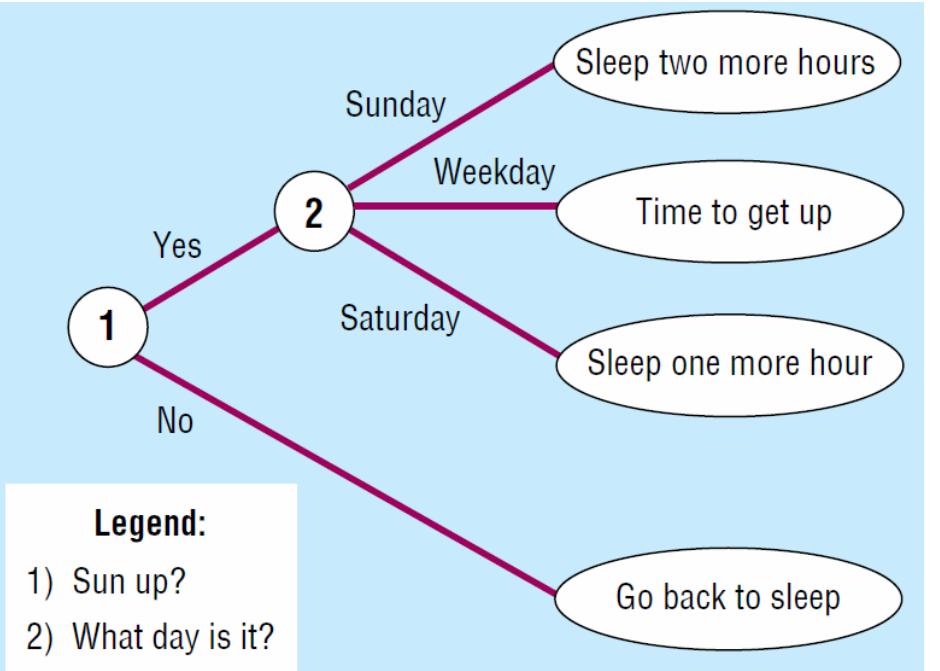
Pohon keputusan adalah representasi dari suatu situasi keputusan dimana simpul-simpul terhubung secara bersama oleh busur (satu busur untuk setiap alternative keputusan) dan berakhir pada oval-oval (aksi dimana hasil semua keputusan dibuat sepanjang jalur yang mengarah ke oval tersebut).

Pohon keputusan pertama kali dipakai sebagai teknik dalam sains manajemen guna menyederhanakan suatu pilihan dimana sebagian informasi yang diperlukan tidak diketahui secara pasti. Dengan

bergantung pada probabilitas dari kejadian-kejadian tertentu, ilmuan sains manajemen menggunakan pohon keputusan untuk memilih aksi yang terbaik. Disini kita memakai pohon keputusan yang dimofifikasi (yaitu tanpa probabilitas) untuk membuat diagram situasi keputusan yang sama dimana kita memakai tabel keputusan.

Mengapa kita perlu mempelajari teknik diagram untuk melakukan apa yang bisa dilakukan oleh tabel keputusan ? Tabel keputusan dan pohon keputusan keduanya merupakan alat komunikasi yang dibuat untuk mempermudah analis berkomunikasi dengan para pemakai. Menentukan teknik mana yang terbaik tergantung pada barbagai faktor yang akan dibahas berikutnya, setelah anda memahami pohon keputusan.

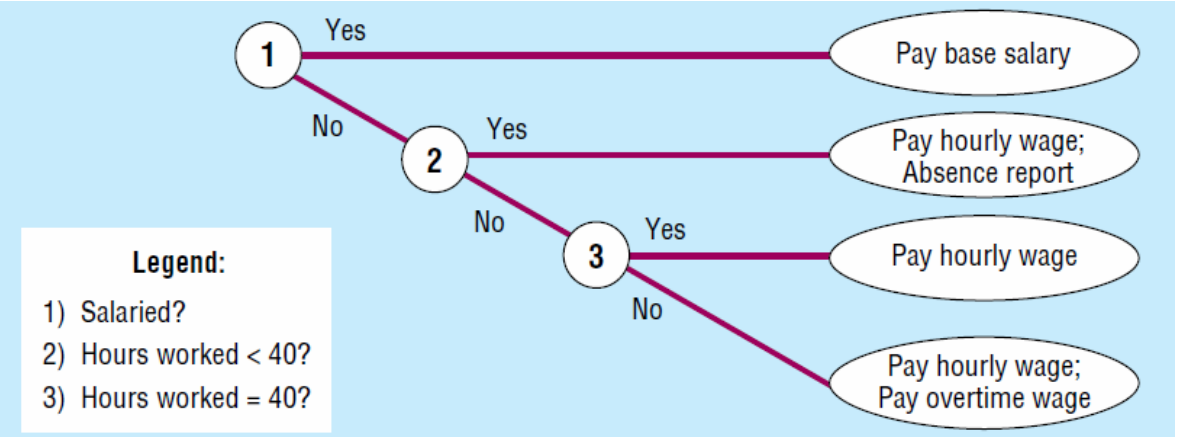
Ketika dipakai dalam strukturisasi kebutuhan, pohon keputusan memiliki dua komponen: titik keputusan, yang diwakili oleh simpul, dan aksi yang diwakili oleh oval. Gambar 9-8 menunjukan satu pohon keputusan umum.



Gambar 9-8: Pohon Keputusan Umum

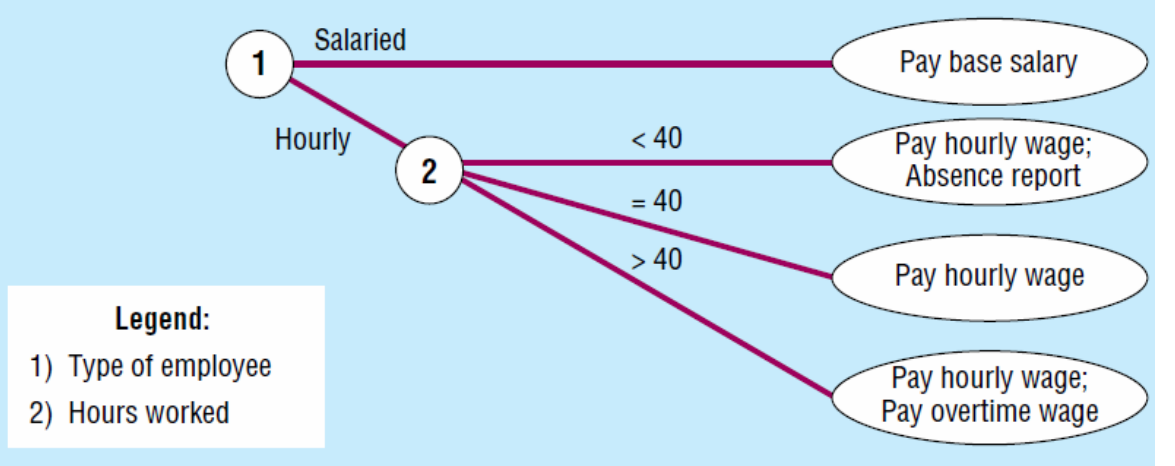
Untuk membaca pohon keputusan, anda mulai dari simpul akar yang berada di ujung kiri. Setiap simpul diberi nomor dan setiap nomor bersesuaian dengan suatu pilihan, pilihan ini dinyatakan pada legenda yang menyertai diagram. Setiap jalur yang meninggalkan suatu simpul bersesuaian dengan satu opsi untuk pilihan tersebut. Dari setiap

simpul paling tidak ada dua jalur yang mengarah ke langkah berikutnya, yang bisa berupa titik keputusan lain atau suatu aksi. Dan akhirnya, semua kemungkinan aksi didaftar pada ujung paling kanan dari diagram dalam simpul akhir. Setiap aturan direpresentasikan dengan penelusuran pada rangkaian jalur dari simpul akar, bergerak ke satu jalur menuju simpul berikutnya, dan seterusnya, sampai satu oval aksi dicapai.



Gambar 9-9: Pohon keputusan representasi logika keputusan dalam table keputusan Gambar 9-4 dan 9-5 dengan hanya dua pilihan per titik keputusan.

Kembali ke tabel keputusan yang kita buat untuk logika tabel penggajian sebelumnya, Gambar 9-4 dan 9-5. Paling tidak ada dua cara untuk merepresentasi informasi yang sama ini dengan pohon keputusan. Pertama dengan Gambar 9-9. Disini semua pilihan dibatasi hanya untuk dua keluaran, yaitu ya atau tidak. Melihat bagaimana kondisi dinyatakan dalam tabel keputusan, anda ingat bahwa jam bekerja memiliki tiga nilai bukan dua. Untuk menjaga keaslian logika keputusan, anda bisa menggambarkan tabel keputusan anda seperti ditunjukan dalam Gambar 9-10.



Gambar 9-10: Pohon keputusan representasi dari logika keputusan dalam table keputusan Gambar 9-4 dan 9-5, dengan banyak pilihan per titik keputusan

Menentukan Antara Bahasa Inggris Terstruktur, Tabel Keputusan, dan Pohon Keputusan

Bagaimana kita menentukan apakah memakai BIT, tabel keputusan, atau pohon keputusan ketika kita akan memperagakan logika proses ? Pada satu jawabannya ialah metoda yang paling anda sukai dan pahami. Sebagai contoh, sebagian analis dan pemakai menyukai melihat logika keputusan yang rumit dalam bentuk tabular, seperti tabel keputusan, sedang lainnya lebih menyukai struktur grafis seperti pohon keputusan.

Masalahnya lebih dari sekedar suka. Hanya karena anda mahir menggunakan palu tidak berarti bahwa palu adalah alat terbaik untuk mereparasi seluruh rumah. Ada saatnya obeng atau bor adalah alat terbaik. Hal yang sama berlaku untuk teknik peragaan atau modeling logika. Anda harus mempertimbangkan tugas yang anda laksanakan dan kegunaan teknik tersebut guna menentukan teknik terbaik. Kelebihan dan kekurangan dari BIT, tabel keputusan, dan pohon keputusan untuk berbagai situasi yang berbeda ditunjukan dalam Tabel 9-2 dan 9-3.

Tabel 9-2: Kriteria Pengunaan Bahasa Inggris Terstruktur, Tabel Keputusan, dan Pohon Keputusan

Kriteria	Bahasa Inggris Terstruktur	Tabel Keputusan	Pohon Keputusan
Penentuan kondisi dan Aksi	Terbaik 2	Terbaik 3	Terbaik

Transformasi kondisi dan aksi ke dalam urutan pernyataan	Terbaik	Terbaik 3	Terbaik
Pemeriksaan konsistensi dan kelengkapan	Terbaik 3	Terbaik	Terbaik

Tabel 9-2 merupakan temuan hasil penelitian untuk membandingkan ketiga teknik. Studi disimpulkan dalam tabel yang membandingkan Bahasa Inggris Terstruktur dengan tabel keputusan dan pohon keputusan untuk dua tugas berbeda. Tugas pertama adalah menentukan kondisi dan aksi yang benar dari suatu deskripsi masalah, situasi sama yang analis jumpai ketika menetapkan kondisi dan aksi setelah mewawancarai pemakai. Studi ini menemukan bahwa pohon keputusan merupakan teknik terbaik untuk proses ini karena pohon keputusan secara alami memisahkan kondisi dan aksi sehingga logika dari aturan keputusan tampak lebih jelas. Meski Bahasa Inggris Terstruktur tidak memisahkan kondisi dan aksi, ia dianggap sebagai teknik terbaik kedua untuk tugas ini. Tabel keputusan merupakan teknik terburuk.

Tugas kedua adalah melakukan konversi kondisi dan aksi menjadi rangkaian pernyataan yang berurut, mirip dengan apa yang analis lakukan ketika mengkonversi kondisi dan aksi yang telah ditetapkan menjadi rangkaian pseudocode atau bahasa pemrograman. Untuk tugas ini Bahasa Inggris Trestruktur merupakan teknik terbaik, karena ia sudah tertulis secara berurut atau sekuensial, tetapi para peneliti menemukan juga bahwa pohon keputusan sama baiknya. Tabel keputusan kembali menjadi yang terakhir.

Meski demikian, pohon keputusan dan dan tabel keputusan memiliki paling tidak satu kelebihan dibanding Bahasa Inggris Terstruktur. Dengan pohon dan tabel keputusan kita dapat memeriksa mengenai kelengkapan, konsistensi, dan tingkat kerangkapan. Kita memeriksa tabel keputusan mengenai kelengkapannya dengan cara memastikan bahwa setiap tabel awal yang dibuat telah memasukan semua aturan yang mungkin. Kita mengetahui bahwa suatu tabel itu lengkap saat kita mengalikan jumlah nilai untuk setiap kondisi untuk mendapatkan jumlah total aturan yang mungkin.

Mengikuti langkah-langkah yang telah dijelaskan di awal bahasan ini juga akan membantu anda memeriksa konsistensi dari tabel keputusan. Prosedur yang sama juga dengan mudah bisa diadopsi untuk pohon keputusan. Sebaliknya, tidak ada cara yang mudah untuk melakukan validasi Bahasa Inggris Terstruktur.

Kriteria	Tabel Keputusan	Pohon Keputusan
Peragaan logika rumit	Baik	Buruk
Peragaan Masalah sederhana	Buruk	Baik
Pembuatan keputusan	Buruk	Baik
Keringkasan dan kerapian	Baik	Buruk
Kemudahan memanipulasi	Baik	Buruk

Para peneliti juga telah membandingkan tabel keputusan dengan pohon keputusan, lihat Tabel 9-3 di atas. Para pelopor Analisis dan Desain Terstruktur menemukan bahwa tabel keputusan adalah yang terbaik untuk menggambarkan atau memeragakan logika rumit sementara pohon keputusan lebih baik untuk masalah sederhana.

Peneliti lain menemukan bahwa pohon keputusan lebih baik dalam pengarahan pembuatan keputusan dalam praktis. Sebaliknya tabel keputusan punya kelebihan dalam hal keringkasan dan kerapian. Selain itu, dengan tabel keputuan lebih mudah bagi analis untuk memanipulasi tabel, misalnya terdapat tambahan kondisi, aksi, dan aturan lebih mudah diakomodasi. Jika tabel menjadi terlalu besar, ia dapat dengan mudah dibagi ke dalan subtabel-subtabel tanpa memakai simbol koneksi seperti halnya pohon keputusan.

Kesimpulan

Pembuatan model atau peragaan logika merupakan satu kegiatan penting pada strukturisasi kebutuhan dalam fase analisis sistem. Terdapat berbagai teknik yang tersedia untuk pembuatan model logika keputusan dalam proses sistem informasi. Satu metoda, BIT, yaitu bentuk khusus, dari bahasa inggris lisan yang biasa analis gunakan untuk melukiskan logika proses yang tergambar dalam DFD. BIT merupakan teknik komunikasi utama untuk analis dan pemakai. Dialek BIT sangat bervariasi sesui yang analis pakai, tetapi BIT merepresentasikan pernyataan berurut (sekuensial), bersyarat atau kondisional, dan perulangan atau repetisi dalam proses sistem informasi.

Tabel keputusan dan pohon keputusan adalah metoda grafis untuk merepresentasikan logika pemrograman. Dalam tabel keputusan, kondisi didaftar dalam bagian kondisi, aksi didaftar dalam bagian aksi, dan aturan menghubungkan kondisi dengan aksi yang harus dijalankan. Langkah pertama dalam pembuatan tabel keputusan adalah mendaftar semua aturan yang mungkin; semua aturan ini adalah hasil dari semua nilai kondisi yang terdaftar. Langkah

berikutnya adalah mengurangi kompleksitas tabel dengan menghilangkan aturan-aturan yang tidak nalar dan dengan menggabungkan aturan untuk kondisi-kondisi takberbeda.

Logika yang digambarkan dalam tabel keputusan dapat digambarkan juga dalam pohon keputusan. Dalam pohon keputusan, kondisi direpresentasikan oleh titik keputusan dan nilai direpresentasikan oleh jalur antara titik keputusan dan oval yang berisi aksi.

Beberapa penelitian membandingkan BIT, tabel keputusan, dan pohon keputusan sebagai teknik representasi logika keputusan. Banyak penelitian menunjukan bahwa pohon keputusan merupakan metoda terbaik untuk banyak criteria sementara tabel keputusan buruk untuk criteria-kriteria tertentu dan terbaik untuk criteria tertentu. Karena tidak ada teknik terbaik untuk digunakan dalam strukturisasi kebutuhan maka analis harus menguasi ketiga teknik ini untuk digunakan pada situasi-situasi berbeda.

Pertanyaan

1. Definisikan istilah-istilah berikut ini:
   a. Bahasa Inggris Terstruktur
   b. tabel keputusan
   c. pohon keputusan
   c. kejadian
   d. keadaan

2. Apa maksud pembuatan model logika ? Teknik apa yang dipakai untuk membuat model keputusan logika dan teknik apa yang dipakai untuk untuk membuat model logika temporal ?

3. Apa Bahasa Inggris Terstruktur itu ? Bagaimana Bahasa Inggris Terstruktur dapat dipakai untuk merepresentasikan sekuen, pernyataan-pernyataan kondisional, dan repetisi dalam suatu proses sistem informasi ?

4. Apa perbedaan Bahasa Inggris Terstruktur dengan pseudocode ?

5. Sebutkan langkah-langkah dalam pembuatan tabel keputusan ? Bagaimana anda mengurangi besar dan kompleksitas dari suatu tabel keputusan ?

6. Jelaskan struktur dari pohon keputusan ?

7. Bagaimana anda mengetahui kapan menggunakan Bahasa Inggris Terstruktur, tabel keputusan, atau pohon keputusan ? Mana yang terbaik untuk situasi apa ?

8. Kata kerja apa yang dipakai dalam Bahasa Inggris Terstruktur ? Tipe kata-kata apa yang tidak dipakai dalam Bahasa Inggris Terstruktur ?

9. Apa yang dimaksud dengan istilah “entri terbatas” dalam suatu tabel keputusan ?

10. Formula apa yang dipakai untuk menghitung jumlah aturan yang tabel keputusan harus cakup.