PERKULIAHAN KE 6

Tujuan Instruksional Khusus (TIK)

Mahasiswa mampu :

- Menjelaskan apa yang dimaksud dengan dialog
- Menyebutkan notasi dalam desain dialog
- Menjelaskan notasi diagramatik dan notasi tekstual
- Menjelaskan semantik dialog
- Menjelaskan beberapa metode analisis dialog
- Membuat contoh desain dialog

Pokok Bahasan :

- Apakah dialog
- Notasi desain dialog
- Notasi diagramatik
- Notasi dialog tekstual
- Semantik dialog
- Desain dan analisis dialog

Deskripsi Singkat : bahasan ini tentang dialog dalam interaksi manusia dan komputer. Dialog tersebut dapat dinotasikan dalam bentuk diagram maupun secara tekstual. Dialog terkait dengan semantik atau apa yang dikerjakan oleh sistem dan presentasi atau bagaimana tampilan sistem.

Bahan Bacaan : Dix, Alan et.al, ***HUMAN-COMPUTER INTERACTION***, Prentice Hall, Europe, 1993, hal 9-114

Newman, W.M., and Lamming, M.G, **Interactive System Design**, Addison Wesley, Cambrigde, UK, 1995

Johnson, P., ***HUMAN-COMPUTER INTERACTION : Psychology, Task Analysis and Software Engineering***, McGraw-Hill, England UK, 1992

NOTASI DAN PERANCANGAN DIALOG

Tinjauan

Dialog adalah level sintaksis dari interaksi manusia dan komputer

- Notasi dialog dapat berupa
  - Diagramatik, mudah dibaca
  - Tekstual, mudah untuk dilakukan analisis formal

- Dialog berkaitan dengan
  - Semantik sistem, apa yang dilakukan oleh sistem
  - Presentasi, bagaimana tampilan sistem

- Deskripsi formal dapat dianalisis terhadap
  - Aksi yang tidak konsisten
  - Aksi yang sulit dikembalikan
  - Item yang hilang
  - Kesalahan miskeying potensial

Apakah Dialog Itu?

Dialog adalah percakapan antara dua atau lebih pihak. Dialog juga mengimplikasikan kerjasama atau sedikitnya keinginan untuk menyelesaikan konflik. Pada perancangan user interface, dialog memiliki arti yang lebih spesifik yaitu struktur percakapan antara user dan sistem komputer. Bahasa komputer dapat dibagi atas tiga tingkatan :

Leksikal, merupakan tingkat yang paling rendah : bentuk icon pada layar, tombol yang ditekan. Pada bahasa manusia, ekuivalen dengan bunyi dan ejaan suatu kata.

Sintaksis, yaitu urutan dan struktur dari input dan output. Pada bahasa manusia, ekuivalen dengan grammar / tata bahasa suatu kalimat.

Semantik, yaitu arti dari percakapan yang berkaitan dengan pengaruhnya pada struktur data internal komputer dan / atau dunia eksternal. Pada bahasa manusia, ekuivalen dengan arti yang berasal dari para partisipan dalam percakapan

Dalam user interface, istilah dialog umumnya dianggap sinonim dengan tingkat sintaksis. Namun batasan antara leksikal dan sintaksis juga tidak begitu jelas dan pada kenyataannya deskripsi dialog seringkali meliputi sifat-sifat leksikal.

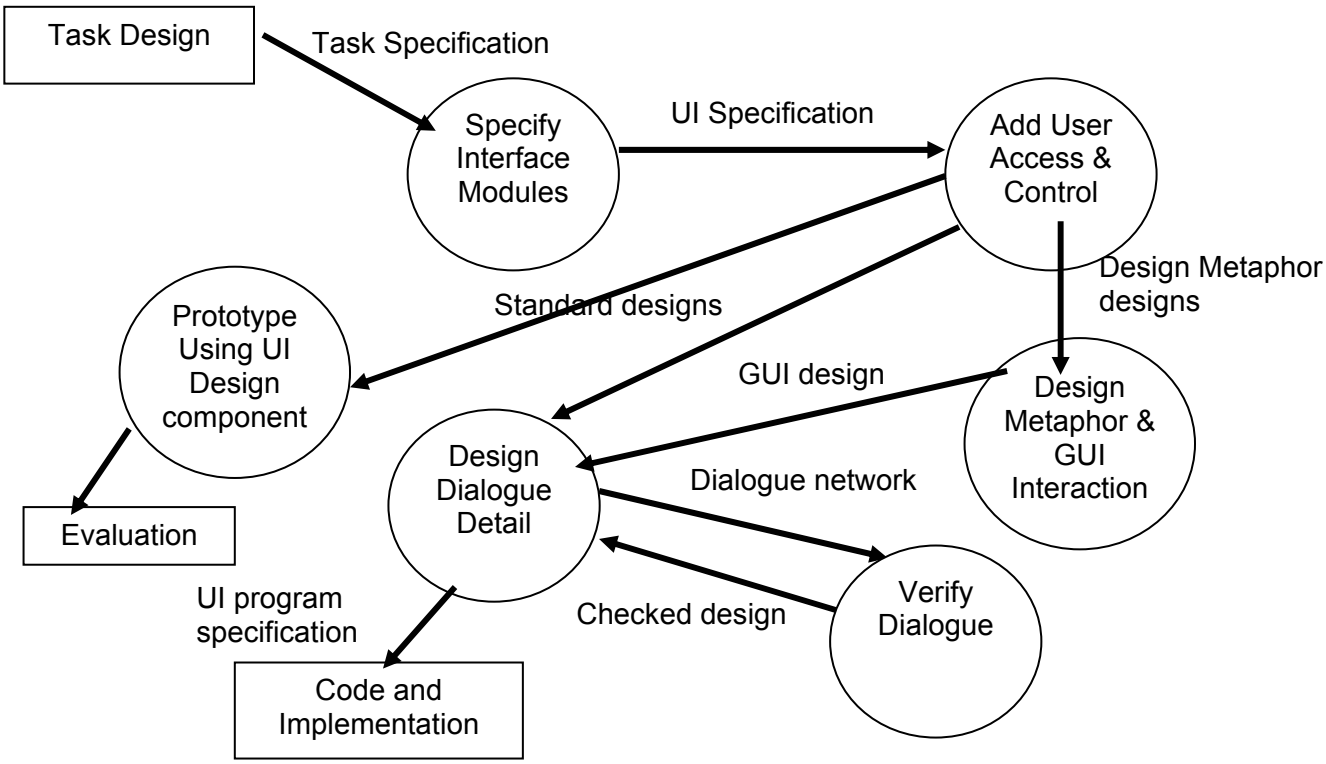
Berbeda dengan dialog antar manusia pada umumnya, dialog dengan komputer biasanya **terstruktur dan terbatas**. Beberapa karakteristik yang dapat ditemui pada sebuah dialog dengan komputer diantaranya adalah :

- Partisipan harus menyebutkan dialognya dalam urutan tertentu.
- Beberapa dialog diantaranya telah ditetapkan sebelumnya.
- Beberapa bagian tertentu dari dialog dilakukan secara bersamaan (*concurrently*).
- Umumnya dialog berikutnya tergantung pada respons dari partisipan
- Dialog dengan komputer mungkin saja tidak mengakomodasi semua kejadian yang mungkin
- Deskripsi dialog biasanya tidak langsung menuju pada arti kata-katanya / semantik tapi pada level sintaksis

Perancangan Dialog

Ada beberapa hal yang perlu diperhatikan dalam perancangan dialog, yaitu :

- Rangkaian Dialog menggambarkan struktur tugas.
- Beberapa rangkaian dialog tambahan digunakan untuk user support, mis. *help system*, *tutorial sub-sytem*.
- Rangkaian dialog diurutkan sesuai dengan struktur tugas

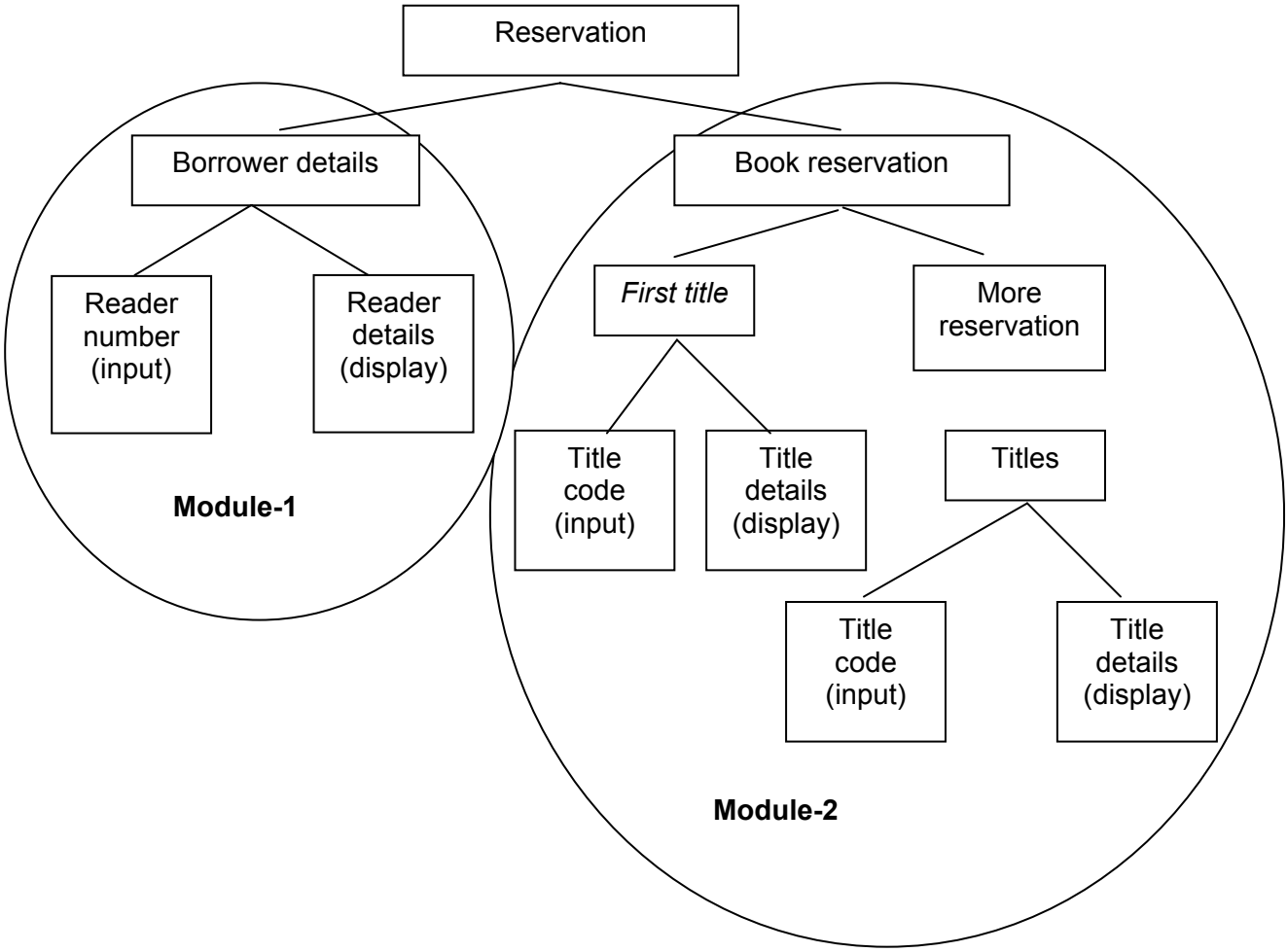


Gambar 6.1 DFD Untuk Desain Dialog

Umumnya *user access* bukan merupakan bagian dari *task description*, tetapi harus disertakan ke dalam sistem yang baru. Empat hal utama dalam desain yang harus diperhatikan dalam *GUI metaphor* :

- Pemilihan dan representasi *conceptual metaphor*
- Representasi obyek interaktif dalam *metaphor*
- Perancangan manipulasi untuk mengimplementasikan aksi user
- Desain *micro-metaphors* untuk kendali aksi (*control action*) dan representasi *command*

Prinsip yang digunakan dalam desain dialog adalah membagi sistem menjadi beberapa bagian yang disebut modul (*module).* Seperti yang dicontohkan pada gambar di bawah ini adalah pembagian modul dalam sebuah sistem pemesanan buku di perpustakaan menjadi 2 bagian.



Gambar 6.2 Pembagian Module Dialog Sistem Pemesanan Buku di Perpustakaan

Dalam mendesain sebuah dialog, diperlukan deskripsi yang terpisah dari program secara keseluruhan. Ada beberapa alasan yang mendasari hal tersebut, yaitu :

- Agar mudah dianalisa
- Pemisahan elemen-elemen interface dari logika program (semantik)
- Apabila notasi dialog ditulis sebelum program dibuat, maka notasi tersebut bisa membantu desainer untuk menganalisis struktur dialog yang diajukan. Bahkan si desainer dapat menggunakan prototyping tool untuk menguji dialog.
- Notasi dialog ini dapat digunakan sebagai salah satu cara bagi anggota tim perancangan untuk mendiskusikan rancangan dialog dan pada akhirnya diberikan kepada programer aplikasi.

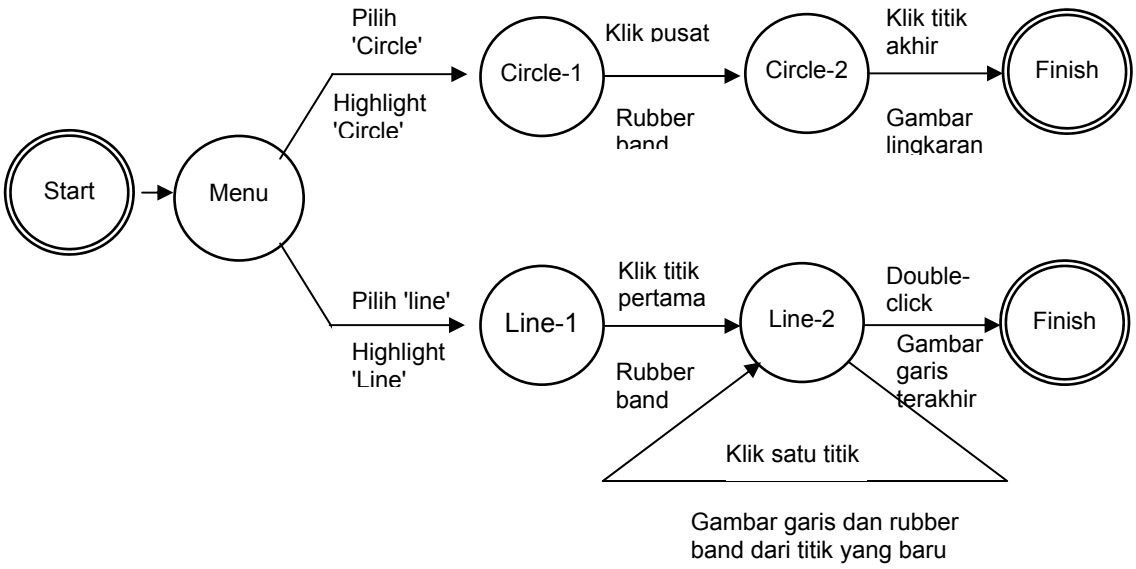
Notasi Diagramatik

Notasi diagramatik merupakan bentuk yang paling sering digunakan dalam desain dialog. Kelebihan dari bentuk ini adalah memungkinkan desainer untuk melihat secara sekilas struktur dialog. Namun kadangkala sulit untuk menjelaskan struktur dialog yang lebih luas dan kompleks. Ada beberapa bentuk notasi diagramatik yang akan dibahas pada bagian ini , yaitu :

- State Transition Networks (STN)
- Hierarchical State Transition Networks
- Harel's State Charts
- Flow Charts
- JSD Diagrams

State Transition Networks (STN)

STN telah lama digunakan untuk mendeskripsikan dialog. Berikut ini dicontohkan sebuah menu dari drawing tool sederhana berbasiskan mouse pada gambar 6.3. Menu tool ini terdiri dari dua pilihan yaitu ‘Circle’ dan ‘Line’. Jika menu ‘Circle’ yang dipilih maka user diperkenankan untuk memilih dua titik pada kertas gambar. Pertama adalah pusat dari lingkaran dan yang lainnya adalah titik kedua sebagai penutup / jarak dari lingkaran. Menu ‘Line’ dipilih jika user akan menggambar polyline. User dapat memilih beberapa titik pada bidang gambar.



Gambar 6.3 State Transition Network Untuk Menu Drawing Tool

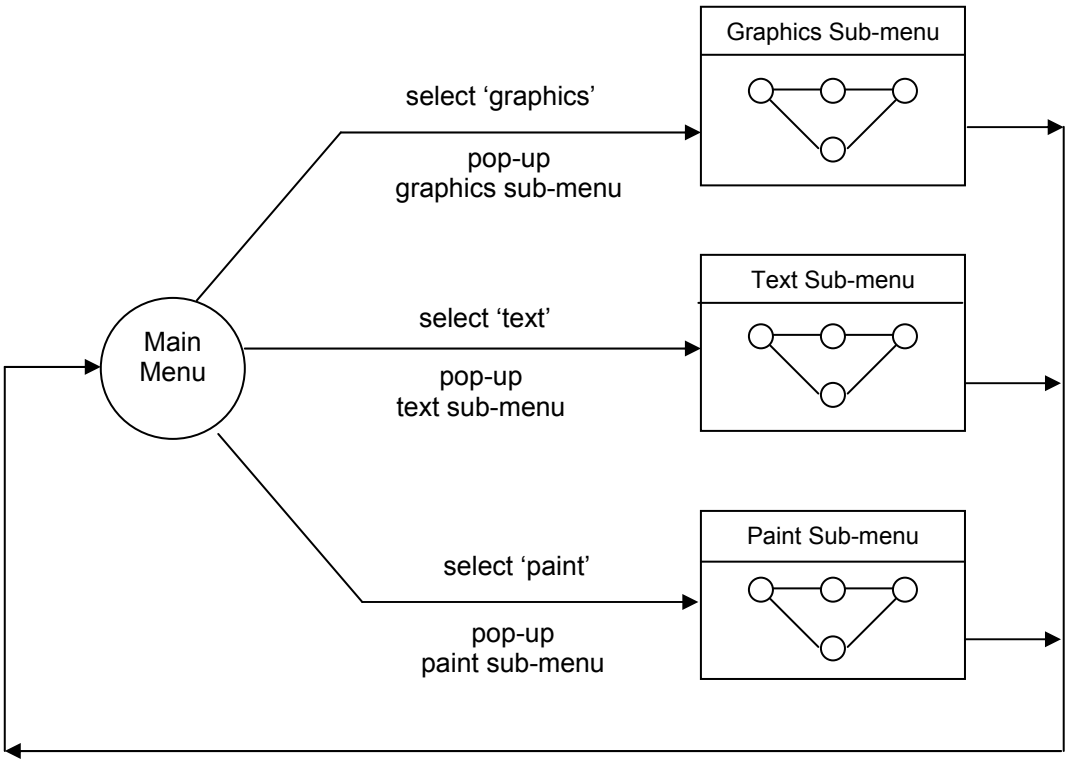
Setiap lingkaran menandakan state / keadaan dari sistem. Misalnya ‘menu’ adalah state sistem menunggu user untuk memilih ‘Circle’ atau ‘Line’. ‘circle-2’ adalah state setelah user memilih sebuah titik sebagai pusat lingkaran dan menunggu user untuk menentukan titik akhir lingkaran. Di antara state-state tersebut terdapat tanda panah yang disebut transisi. Tanda panah ini diberi label yang menjelaskan tentang tindakan user yang menyebabkan transisi perpindahan state dan respon dari sistem. Sebagai contoh, state ‘circle-1’ adalah state sistem menunggu user untuk memilih pusat lingkaran. Jika user telah memilih / meng-klik pusat lingkaran maka state sistem akan bertransisi ke ‘circle-2’ dan direspon oleh sistem dengan menggambar rubber band.

Dari gambar 6.3, kita dapat menyimpulkan bahwa STN dapat merepresentasikan beberapa hal yang terkait dengan dialog yaitu :

- Urutan (*sequence*) dari aksi yang dilakukan user dan respon yang diberikan oleh sistem.
- Pilihan bagi user (*choice*)
  Contoh: dari state Menu, user dapat memilih "Circle" sehingga sistem berpindah ke state Circle-1 dan pilihan "Circle" pada menu di-highlight; alternatif lain, user dapat memilih "Line" sehingga sistem berpindah ke state Line-1.
- Iterasi (*iteration*)
  Contoh: pada state Line-2, transisi dapat kembali ke state Line-2 jika user menambah titik baru pada polyline dan akan berpindah ke state Finish hingga user melakukan double-click.

Hierarchical State Transition Network

Misalkan drawing tool pada bahasan sebelumnya memiliki menu utama yang terdiri dari tiga submenu seperti submenu graphic, text dan paint. Untuk mendeskripsikan sistem yang lengkap dapat digunakan *Hierarchical State Transition Networks* seperti yang ditunjukkan pada gambar 6.4 di bawah ini.



Gambar 6.4 Hierarchical STN dari Sebuah Drawing Tool Lengkap

Struktur *Hierarchical State Transition Networks* mirip dengan STN namun memiliki tambahan berupa gabungan state (*composite state*) yang digambarkan dengan persegi panjang dengan gambar struktur STN berukuran kecil di dalamnya. Masing-masing persegi panjang ini menggambarkan submenu yang berkaitan. Submenu ini dapat dispesifikasikan dengan rinci pada STN tersendiri dengan menaruh label nama submenu yang bersangkutan pada simbol "start"-nya.

Pengunaan hirarki ini tidak mengubah fungsi notasi dasar STN namun seperti menggabungkan beberapa STN ke dalam satu diagram besar sehingga model ini dapat digunakan untuk sistem-sistem yang besar.

Harel's State Charts

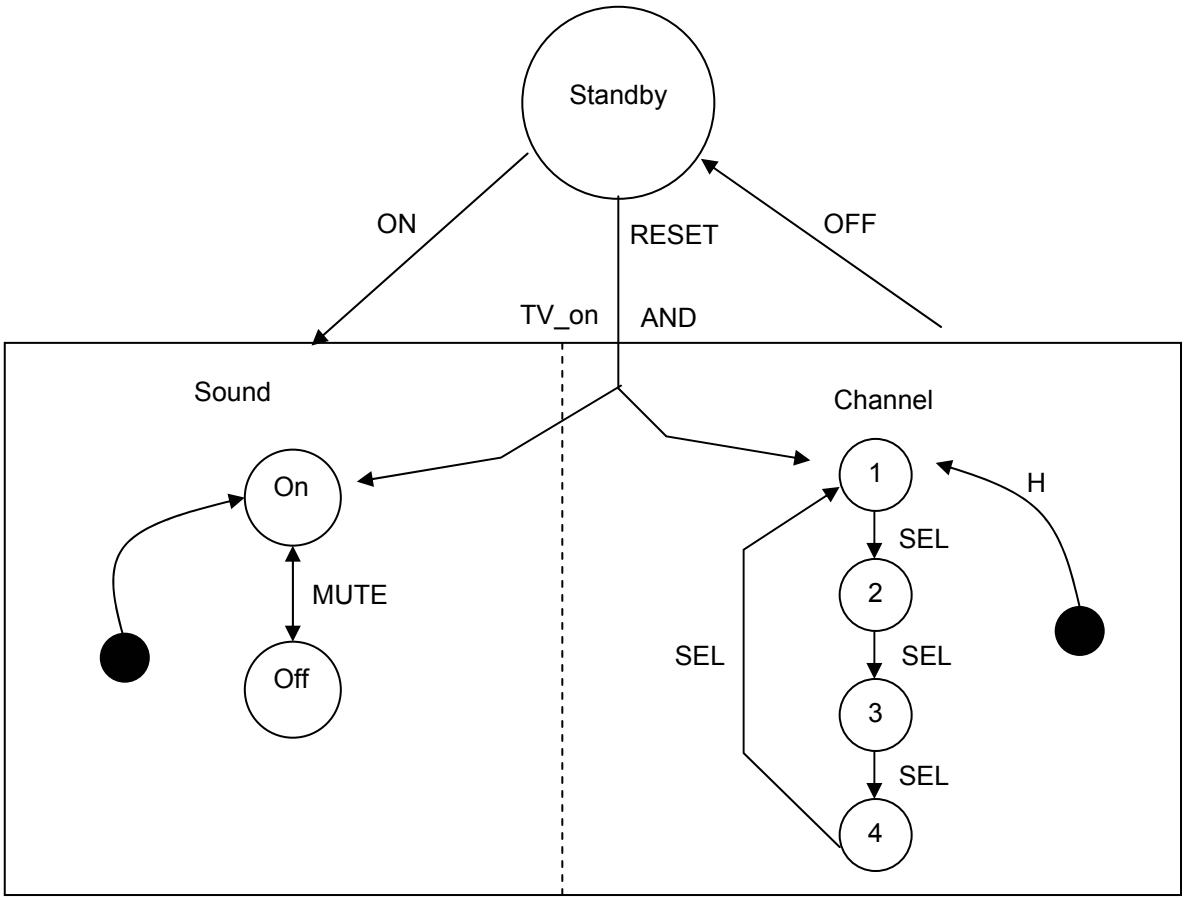
Harel's state chart dapat digolongkan sebagai kelompok STN. Chart ini dibangun untuk menspesifikasikan secara visual sistem reaktif yang kompleks dan mampu mengakomodasi masalah seperti *concurrency* dan *escape*. Chart ini memiliki karakter struktur hirarki dalam satu diagram tunggal yang membagi elemen mana yang merepresentasikan state alternatif dan yang merepresentasikan aktifitas concurrent.

Pada gambar 6.5 berikut ini dicontohkan state chart dari panel kendali televisi. Panel kendali ini memiliki lima tombol berlabel '*ON*', '*OFF*', '*MUTE*', '*SEL*', dan

'*RESET*'. Televisi yang dimaksud hanya dapat berada pada keadaan *ON* atau *standby*. Dimisalkan kita mulai pada keadaan standby. Menekan tombol *ON* atau *RESET* akan menyebabkan televisi menyala (TV_on). Dan pada saat TV menyala, tombol *OFF* akan menyebabkan TV kembali ke keadaan *standby*. Pada saat TV menyala, user dapat mengendalikan suara (*sound*) dengan tombol '*MUTE*' yang mengatur suara menjadi ON atau OF; dan saluran TV (*channel*) dengan tombol '*SEL*' yang dapat memilih diantara empat *channel* yang ada.

Sub-dialog SOUND dan CHANNEL merupakan bagian dari state gabungan (*composite state*) TV_ON. Garis putus-putus diantara kedua sub-dialog dan keyword AND menyatakan bahwa kedua sub-dialog tersebut dapat dijalankan bersama-sama (*concurrent*) dan dalam urutan yang bebas.

Sub-dialog SOUND memiliki lingkaran hitam kecil yang dihubungkan dengan garis lengkung ke state ON yang menunjukkan state awal dan nilai default dari state awal tersebut. Hal ini berarti bahwa setiap kali TV dinyalakan maka state SOUND pun akan ON. Pada sub-dialog CHANNEL sedikit berbeda, yaitu ada penandaan huruf 'H' kependekan dari 'History' pada state awal. Ini mengindikasikan bahwa sub-dialog channel akan mengingat posisi channel TV terakhir yang diaktifkan user. Pada saat awal TV dihidupkan akan dimulai pada channel 1, namun setelah itu akan berada di channel terakhir yang diaktifkan user. RESET akan menghapus default state awal. Jika garis ON menuju hanya ke kotak TV_ON, RESET langsung menuju ke state tertentu pada dialog TV_ON. Hal ini mengakibatkan, setiap kali tombol RESET ditekan dari standby mode maka TV akan hidup dengan setting sound 'ON' dan channel '1'. Tombol 'OFF' juga dapat bertindak sebagai escape pada dialog TV_ON ini.



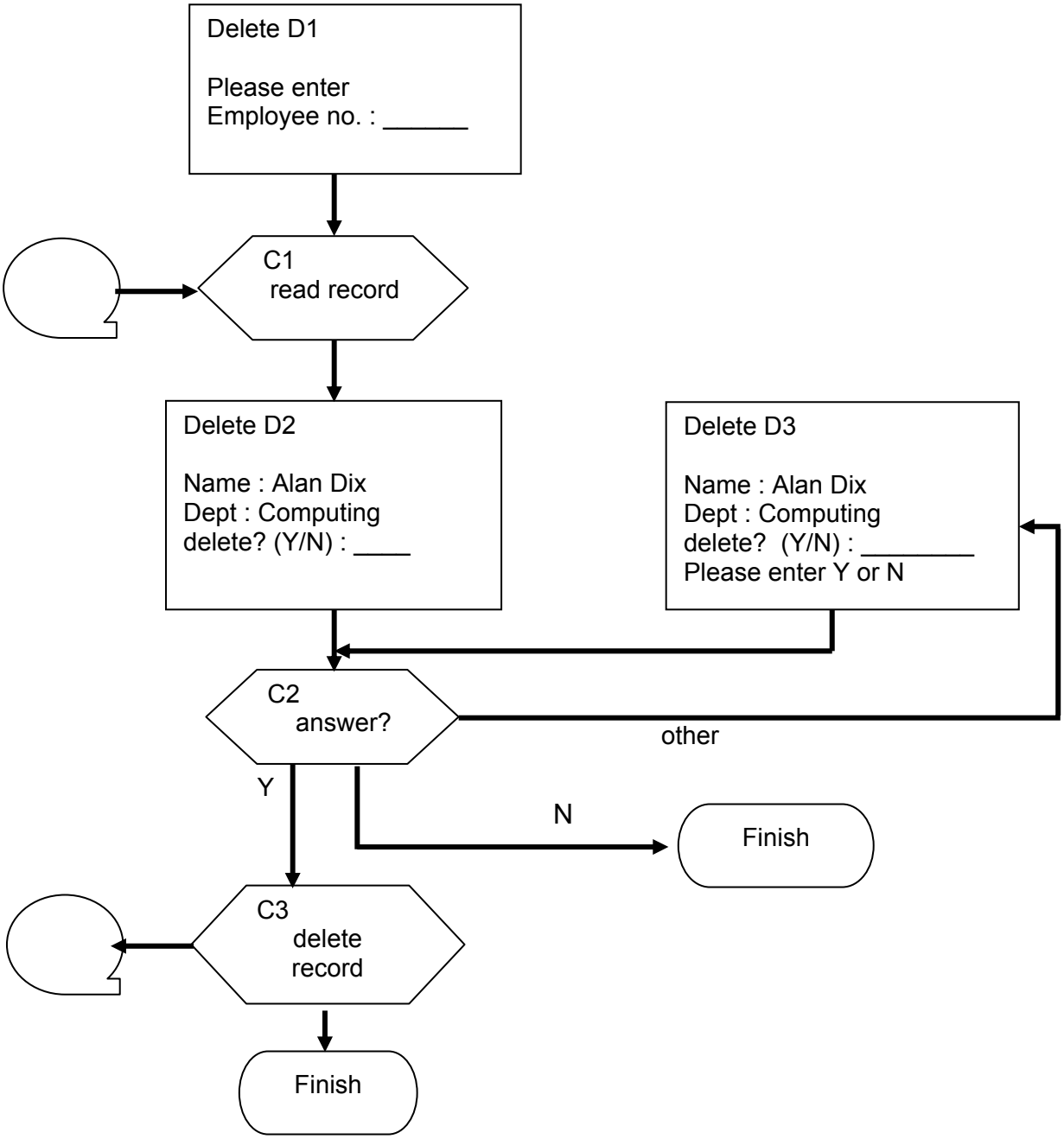
Gambar 6.5 State Chart untuk Panel Kendali Televisi

Flow Chart

Meskipun kelihatannya telah jauh tertinggal, Flow Chart dengan berbagai bentuknya mungkin merupakan notasi diagramatik yang paling banyak digunakan. Flow Chart digunakan untuk mendeskripsikan dialog yang sederhana dan memiliki kelebihan dalam hal kesederhanaannya dan mudah dimengerti.

Kotak pada Flow Chart merepresentasikan proses atau keputusan sehingga tidak ekuivalen dengan state pada STN. Flow Chart menggunakan berbagai jenis kotak untuk merepresentasikan berbagai jenis aktifitas yang berbeda, namun lebih merefleksikan sudut pandang programer dibandingkan user. Pada gambar 6.6 berikut ini adalah flow chart subdialog delete dari sistem update database karyawan. Flow chart terdiri dari dua tipe kotak, yaitu persegi panjang adalah screen yang digunakan untuk berkomunikasi dengan user, dan segi enam adalah proses dan keputusan yang dibuat oleh sistem. Dengan tambahan elips "Finish" yang berarti kembali ke menu utama dan simbol tape yang berarti membaca atau mengubah database.

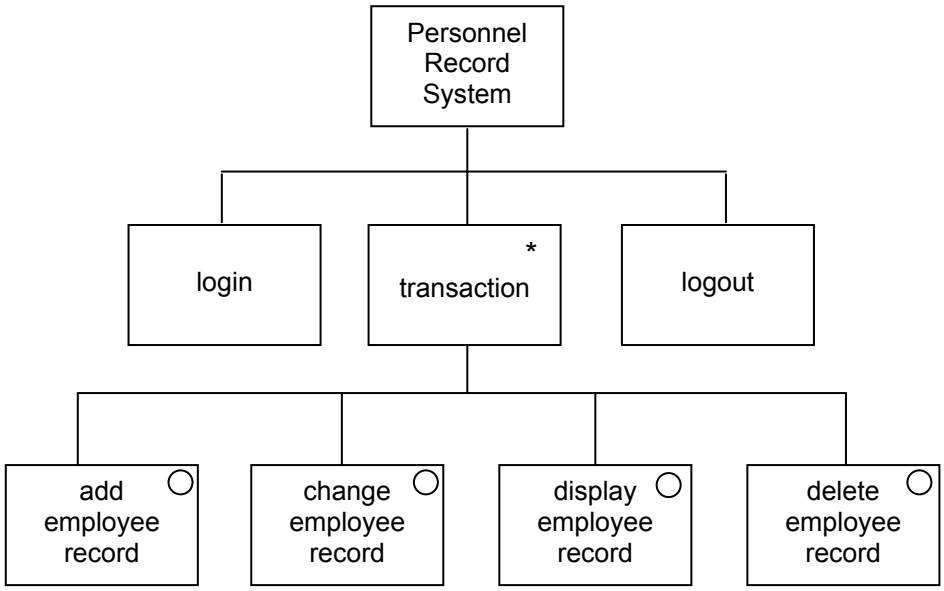
Perbedaan utama antara menggunakan flow chart untuk perancangan dialog dengan pemrograman adalah tingkat detail pada sisi program. Misalkan, jika pembacaan record karyawan melibatkan pencarian secara sequential pada file untuk mencari record tertentu, maka flow chart program akan menyertakan loop pencarian tersebut. Sedangkan pada flow chart dialog, sama sekali tidak akan dicantumkan loop semacam itu.



Gambar 6.6 Flow Chart Sub-dialog Fungsi Delete

JSD Diagram

JSD (*Jackson Structured Design*) diagram telah digunakan untuk berbagai aspek dari analisis tugas dan notasi dialog. Seperti halnya flow chart, JSD memiliki kelebihan dalam hal model ini telah dikenal luas oleh para programer. Gambar 6.7 di bawah ini dicontohkan sebuah JSD Diagram untuk struktur top level dari sistem kepegawaian. Sistem ini dipergunakan untuk meng-update record pegawai dalam berbagai cara, yaitu menambah baru, menampilkan, mengubah dan menghapus.



Gambar 6.7 JSD Diagram untuk Sistem Kepegawaian

Diagram ini mirip dengan diagram hirarki yang digunakan pada analisis tugas yang sudah kita bahas sebelumnya. Diagram JSD sistem kepegawaian ini dibagi menjadi tiga bagian yaitu *login*, *transaction*, dan *logout*. Urutan pengoperasiannya berjalan dari kiri ke kanan, yang berarti seorang user untuk dapat melakukan transaksi harus login terlebih dahulu dan terakhir logout. Tanda asterik (*) merepresentasikan iterasi atau pengulangan. Tanda (o) merepresentasikan pilihan atau opsional. Dengan demikian, dialog ini berarti bahwa operasi dalam sistem kepegawaian terdiri dari sebuah login, diikuti sejumlah transaksi yang terdiri dari empat jenis pilihan yaitu

menambah, mengubah, menampilkan atau menghapus record pegawai, dan akhirnya logout.

Kelas dari dialog yang dapat direpresentasikan dengan JSD adalah sederhana, terbatas dan merupakan sistem informasi yang berbasis menu (*menu-driven information system*).

Notasi Dialog Tekstual

Tata Bahasa (*Grammars*)

Karena diagram yang digunakan untuk menggambarkan dialog komputer juga memiliki sifat linguistik, maka tidak mengherankan jika *formal grammar* juga dipakai sebagai notasi dialog. Salah satu bentuk formal yang sering digunakan untuk notasi dialog tekstual adalah BNF (*Backus-Naur Form*) dan *regular expression*.

BNF dan *regular expression* berfokus pada aksi yang dilakukan user bertolak belakan dengan STN yang berkonsentrasi pada state (kondisi). *Regular expression* lebih banyak digunakan untuk mendeskripsikan kriteria pencarian tekstual yang kompleks dan analisis leksikal bahasa pemrograman. *Regular expression* serupa dengan BNF namun memiliki keterbatasan. Notasi yang digunakan pada *regular expression* berbeda dengan BNF, dan buruknya lagi terdapat beberapa jenis *regular expression* yang memiliki notasi yang berbeda.

Sebagai contoh, proses menggambar garis seperti yang terdapat pada gambar 6.3 dapat direpresentasikan dengan *regular expression* :

```
select-line click click* double-click
```

ekspresi ini berarti bahwa user harus memilih menu line, click posisi awal garis pada di manapun, kemudian posisi garis berikut dapat terdiri dari beberapa titik dengan melakukan click berkali-kali dan diakhiri dengan double-click.

Tanda asteriks (*) pada *regular expression* berarti iterasi atau pengulangan. Pada BNF iterasi ini direpresentasikan dengan sequencing (+) menjadi bentuk :

```
some-thing ::= thing + some-thing
```

Seperti halnya STN, BNF dan *regular expression* tidak dapat digunakan untuk merepresentasikan *concurrency dialog* (dialog yang dilakukan bersamaan). Salah satu kelebihan penggunaan BNF dan *regular expression* adalah kedua notasi mudah diimplementasikan karena telah tersedia tools yang memadai.

Production Rules

Kita telah membahas *production rules* pada saat pembahasan CCT. Seperti telah diketahui, bentuk umum dari production rule adalah :

If *condition* then *action*

Production rule ini juga memiliki beberapa bentuk representasi yang lain yaitu :

condition ➔ *action*
condition: *action*
event: condition ➔ *action*

Semua *rule* umumnya bersifat aktif dan sistem secara konstan mencocokan isi dari bagian *condition* pada *rule* dengan *event* (kejadian) yang dimunculkan oleh user dan isi memori. Jika isi dari *condition* terpenuhi, maka bagian *action* dijalankan. Isi dari *action* dapat berupa respon kepada user atau perubahan terhadap memori sistem. Urutan penulisan *rule* tidak begitu penting, *rule* manapun dapat diaktifkan pada saat kapanpun selama kondisi terpenuhi.

Sistem *production rule* dapat berorientasi pada *event*, *state*, ataupun gabungan dari keduanya. Pertama akan dibahas *production rule* yang berorientasi pada *event*. Sebagai contoh adalah aktifitas menggambar garis / polyline seperti pada gambar 6.3 digambarkan sebagai *production rule* :

Sel-line ➔ start-line <highlight 'line'>
C-point start-line ➔ rest-line <rubber band on>
C-point rest-line ➔ rest-line <draw line>
D-point rest-line ➔ <draw line> <rubber band off>

Pada contoh di atas, bagian *condition* dan *action* menggunakan bentuk sederhana dari nama event. Terdapat tiga jenis event, yaitu :

User events, yang diawali dengan huruf kapital. Pada contoh diatas adalah *Sel-line* (user memilih 'line' pada menu), *C-point* dan *D-point* (merepresentasikan single click dan double click pada bidang gambar yang dilakukan user).

Internal events, yang dimulai dengan huruf kecil. Bagian ini digunakan untuk mencatat sejarah state dialog, sebagai contoh rest-line adalah state setelah titik pertama garis dipilih oleh user.

System response events, yang diapit oleh tanda kurung siku seperti <draw line> merupakan efek sistem yang dapat dilihat atau didengar.

Dialog manager yang mengeksekusi *production rule* memiliki memori yang terdiri dari sekumpulan event. Sebuah rule akan dijalankan jika nama event yang ada di bagian *condition* juga terdapat di memori. Semua interaksi dengan user dimediasi oleh memori event dengan cara : event dari user seperti *mouse-click* ditambahkan ke dalam memori kemudian respon sistem seperti <*draw line*> dihapus dari memori dan dilaksanakan oleh *display controller*.

Contoh : Jika user memilih option *'line'* dari menu, memori sistem akan berisi *Sel-line*. Ini berarti rule yang pertama akan dijalankan. *Sel-line* kemudian dihapus dari memori dan digantikan oleh *start-line* dan <*highlight 'line'*>. Akhirnya *display controller* menghapus <*highlight 'line'*> dan melakukan aksi tersebut. Dengan demikian memori sistem hanya akan berisi *start-line* dan tidak ada rule lagi yang dijalankan hingga user melakukan suatu tindakan tertentu.

Production rule yang berorientasi *state* memiliki perilaku yang berbeda. Memori sistem terdiri dari sekumpulan variabel dengan nilai namun tidak dihapus pada saat rule dijalankan, sebaliknya mereka harus dihapus secara eksplisit oleh isi dari bagian *action*. *Production rule* di CCT beroperasi dengan cara demikian seperti pada contoh *(DELETE-GOAL insert space)*. Contoh dari *state-oriented production rule* yang lain adalah Olsen's *propositional production system* (PPS). Pada PPS, *state* dari sistem dibagi menjadi rangkaian hingga atribut, yang masing-masing memiliki nilai. Sebagian dari atribut merupakan hasil dari aksi yang dilakukan user, dan lainnya adalah efek pada display sistem. Sebagai contoh, menu pada gambar 6.3 dideskripsikan menjadi :

Mouse: {mouse-off, select-line, click-point, double-click}
Line-state: {menu, start-line, rest-line}
Rubber-band: { rubber-band-off, rubber-band-on}
Menu: {highlight-off, highlight-line, highlight-circle}
Draw: {draw-nothing, draw-line}

Atribut yang pertama, mouse diatur secara otomatis setiap kali user melakukan aksi yang relevan, atribut yang kedua digunakan untuk menyimpan sejarah state dan tiga atribut yang terakhir untuk mengendalikan respon sistem. Dengan menggunakan production rule PPS, maka contoh menggambar garis / polyline dideskripsikan menjadi :

select-line ➔ mouse-off start-line highlight-line
click-point start-line ➔ mouse-off rest-line rubber-band-on
click-point rest-line ➔ mouse-off draw-line
double-click rest-line ➔ mouse-off menu draw-line rubber-band-off

CSP dan Event Algebras

STN adalah contoh notasi dialog yang bagus digunakan untuk menunjukan urutan (*sequence*) namun tidak dapat merepresentasikan *concurrency*. Dan sebaliknya *production rule* dapat digunakan untuk *concurrency* namun tidak bagus untuk *sequence*. Masalah sequence dan concurrency adalah masalah yang umum dihadapi dalam area komputasi terutama dalam protokol komunikasi dan *concurrent programming*.

Process algebras adalah sebuah kelas notasi formal yang dibangun untuk menangani situasi yang kompleks. Salah satu contohnya adalah CSP (*Communicating Sequential Processes*) yang dapat menspesifikasikan *concurrency* dan *sequence* sama baiknya dan mudah untuk dibaca. Berikut ini adalah contoh notasi CSP untuk menu Draw pada gambar 6.3 :

```
Draw-menu      = ( select-circle? ➔ Do-circle
                 [] select-line? ➔ Do-line)
Do-circle      = click? ➔ set-centre ➔ click?
                 ➔ draw-circle ➔ skip
Do-line        = Start-line ; Rest-line
Start-line     = click?  ➔ first-point ➔ skip
Rest-line      = (click? ➔ next-point ➔ Rest-line
                 [] double-click? ➔ last-point ➔ skip)
```

Pada deskripsi di atas digunakan beberapa simbol operator, yaitu :

- Simbol (?) adalah event yang berupa aksi mouse yang dilakukan user. Event lain yang tidak diberi simbol merupakan event internal sistem.
- Simbol (=) digunakan untuk membangun deskripsi yang berarti “didefinisikan sebagai”.
- Simbol (➔) berarti urutan (*sequence*) event
- Simbol (**;**) menunjukkan urutan proses
- Simbol [] untuk menunjukkan pilihan (*choice*)
- Semua nama event pada dialog ditulis dalam huruf kecil, sedangkan nama proses diawali dengan huruf besar.

Selain operator di atas, terdapat operator (| |) yang mengindikasikan komposisi paralel. Jika kita memiliki dua buah proses P dan Q, maka P | | Q

merupakan proses paralel P dan Q atau P dan Q dapat dijalankan bergantian (*interleaving*). Contoh lainnya dapat dilihat di bawah ini :

Dialogue-box
= Bold-toggle | | Italic-toggle | | Under-toggle

Bold-toggle = select-bold? ➔ bold-on
➔ select-bold? ➔ bold-off
➔ Bold-toggle
Italic-toggle = select-italic? ➔ italic-on
➔ select-italic? ➔ italic-off
➔ Italic-toggle
Under-toggle = select-under? ➔ under-on
➔ select-under? ➔ under-off
➔ Under-toggle

select-bold? Merepresentasikan user memilih pilihan bold dengan mouse. Proses Bold-toggle mengaktifkan style bold on dan off, demikian pula dengan Italic-toggle untuk style italic dan Under-toggle untuk style underline. Ketiga proses tersebut dapat dijalankan bersama atau bergantian (*interleaving*).

Semantik Dialog

Jika tujuan dari notasi / deskripsi dialog adalah untuk berkomunikasi antar para desainer atau sebagai alat untuk mendeskripsikan pemikiran mengenai dialog, maka perlu ditambahkan catatan pada dialog formal mengenai arti dari suatu aksi tertentu atau dibiarkan user yang membaca menginterpretasikan sendiri. Namun jika notasi dialog berfungsi sebagai prototype atau bagian dari sebuah kontrak maka perlu dideskripsikan pula semantik dari dialog tersebut. Notasi dialog yang telah kita bahas hanya mendeskripsikan strukturnya tidak membahas mengenai arti / semantiknya.

Ada dua aspek dari semantik dialog, yaitu secara internal terhadap aplikasi dan eksternal terhadap presentasi. Semantik dari dialog harus dapat menhubungkan kedua aspek tersebut. Berikut ini akan dibahas tiga pendekatan untuk menghubungkan dilaog dengan semantik, yaitu :

Notation-specific semantics merupakan bentuk semantik dengan tujuan khusus (*special-purpose semantic forms*) yang didesain sebagai bagian dari notasi dialog.

Links to programming languages, dengan menyertakan sebagian koding bahasa pemrograman ke dalam notasi dialog.

Links to specification notations, hampir sama dengan di atas, namun menggunakn notasi spesifikasi formal (*formal specification*).

Notation-specific Semantics

Bentuk semantik ini didesain sebagai bagian dari notasi dialog. Salah satu contohnya adalah *Augmented Transition Networks* (ATN) yang merupakan bentuk STN. Pada ATN, sistem diasumsikan memiliki sekumpulan register, lokasi penyimpanan (*storage*) sebagai tempat bagi STN untuk melakukan pengaturan (*set*) dan pengujian (*test*). Pada STN, tanda panah diberi label event yang mengakibatkan transisi serta respon sistem. Maka pada ATN, label ini diperluas dengan menambahkan kondisi event. Kondisi ini dapat berupa isi register sistem. Respon sistem juga diperluas tidak hanya mencakup feedback dan display namun juga setting dari register. Register ini dapat digunakan untuk untuk mendeskripsikan dialog yang lebih kompleks serta berkomunikasi dengan aplikasi dan menyimpan nilai yang berasal dari mouse.

Links to Programming Languages

Kadangkala notasi dialog ditambahkan ke dalam bahasa pemrograman konvensional. Salah satu contohnya adalah *input tool*, sebuah notasi berbasis regular expression yang digunakan bahasa C untuk mengekspresikan semantik dialog. Deskripsi pada input tool menyertakan regular expression yang dicampur dengan kode bahasa C. Berikut ini adalah contoh dari *input tool* :

```
tool number
{   char buf[80];
    int index;
    int positive;

    input { (digit* + sign; digit; digit*) ; return }

    tool digit
    {    input { key : | key_c >='0' &&  key_c<='9' | }

          if  (index < 79)   /*  append character to string  */
         { buf[index]=key_c;
           index = index + 1;
```

```
            echo (key_c);
          }
      }
      tool sign
      …
      tool return
      {  input { key : \ key_c == '\n' | }
         …
      }
     …
}
```

Keyword *'tool'* mengenalkan tool baru, hal ini similar dengan non-terminal pada grammer BNF. Tool diatur dalam bentuk hirarki yang terdiri dari *'digit'*, *'sign'*, dan *'return'*. *'echo'* menampilkan / memberikan karakter kepada user. *'key'* adalah mekanisme yang digunakan untuk mencocokan karakter tunggal yang dibaca dari keyboard, dan disimpan pada variabel global *key_c*.

Links to Formal Specification

SPI (*Specifying and Prototyping Interaction*) dibagi menjadi dua bagian, yaitu *eventCSP* yang merupakan notasi dialog yang berbasiskan CSP, dan *eventISL* yang mendeskripsikan semantik dialog. CSP telah kita bahas sebelumnya dan untuk setiap event pada CSP memiliki definisi pada eventISL. EventISL bersifat sebagian standar dan sebagian lainnya bebas dari bahasa (*host language*) yang digunakan. Bagian yang bergantung pada *host language* diantaranya adalah klausa variabel global yang digunakan dan di-*update* oleh *event*, *pre-condition* yang mengekspresikan kapan *event* terjadi, serta bagian input dan output.

Berikut ini dicontohkan deskripsi eventCSP dari urutan login :

```
Login    = login-mess ➔ get-name ➔ Passwd
Passwd   = passwd-mess ➔ ( invalid ➔ Login
                         [] valid ➔ Session )
Session  = ( logout ➔ Login
           [] command ➔ execute ➔ Session )
```

Deskripsi eventISL untuk event diatas adalah sebagai berikut :

```
event: login-mess =
        prompt: true
        out: "login:"
```

event: get-name =
uses: input
set: user-id = input

event:passwd-mess =
prompt: invis
out: ‘passwd”

event: valid =
uses: input, user-id, passwd-db =
when: passwd-id = passwd-db(user-id)

event: invalid =
uses: input, user-id, passwd-db =
when: passwd-id ≠ passwd-db(user-id)
out: “Sorry bad user-id / password”

EventISL sangat bergantung pada penggunaan variabel global untuk melewatkan informasi antar event. Variabel global digunakan dan di-update secara eksplisit dan membuat penelusuran kembali menjadi lebih mudah. Notasi dialog menyertakan mekanisme strukturisasi yang efektif namun sayangnya hal ini tidak terdapat pada deskripsi semantiknya.

Desain dan Analisis Dialog

Pada bagian ini akan dibahas beberapa cara untuk menganalisis sebuah dialog untuk menemukan masalah usability yang potensial berkaitan dengan prinsip usabilty yang telah dibahas sebelumnya. Terdapat tiga isu yang berkaitan dengan analisis properti dialog ini, yaitu :

1. Berfokus pada aksi yang dilakukan user, apakah dispesifikasikan dengan cukup dan konsisten.
2. Memperhatikan state dialog, menyangkut state mana yang diinginkan dan yang ingin dihilangkan.
3. Isu presentasi dan leksikal, bagaimana tampilan dan fungsi sebuah tombol (*key*).

Action Properties

Ada tiga karakteristik (*properties*) dialog yang terkait dengan aksi yang dilakukan oleh user, yaitu kelengkapan (*completeness*), deterministik (*determinism*), dan konsistensi (*consistency*). Dalam hal **kelengkapan (*completeness*)**, selain aksi user yang normal / umum, desainer bertanggung jawab untuk melihat jauh ke depan dan mengantisipasi bagaimana perilaku sistem pada kondisi yang tidak diperkirakan atau pada kondisi khusus. Desainer dapat mendaftar semua aksi yang mungkin dan pada setiap state dialog mencari semua aksi yang mungkin terlupakan. Pada setiap state yang diperkirakan akan muncul aksi user yang khusus, desainer harus dapat memutuskan atau paling tidak dicek pada saat pengujian, bagaiman perilaku sistem terhadap kejadian tersebut. Hal yang paling sederhana adalah dengan menetapkan bahwa semua aksi / perilaku khusus tersebut tidak akan memberikan dampak terhadap sistem kecuali dengan mengeluarkan peringatan (*warning*) bagi user. Jika cara ini tidak bisa ditempuh maka perlu dicarikan cara penanganan lain. Misalnya jika pada saat user sedang menggambar garis, sebelum melakukan double-click sebagai tanda dari titik akhir garis, user memilih option lain dari menu. Salah satu pilihan untuk menangani hal ini adalah dengan membatalkan gambar garis yang belum lengkap. Inipun jika memungkinkan, misalnya user baru memilih satu titik saja. Namun jika misalnya user telah memilih 20 titik dengan suatu perhitungan yang cermat, maka yang dapat dilakukan adalah dengan meminta konfirmasi setiap kali user beralih ke option lain pada menu.

Selain hal di atas, kita mungkin akan menemukan kasus beberapa state memiliki sejumlah panah dengan label aksi yang sama. Atau pada sebuah *production rule*, terdapat dua buah rule yang diaktifkan oleh sebuah event. Dalam hal ini dikatakan bahwa aksi tersebut tidak memiliki sifat **deterministik**. Bentuk formal seperti *production rule* memiliki aturan default untuk mengatasi hal tersebut. Salah satunya dengan memperbolehkan keduanya dijalankan atau salah satu yang memiliki prioritas lebih tinggi. Kadangkala ketidak-deterministikan ini merefleksikan keputusan semantik dalam sistem.

Karakteristik yang ketiga adalah **konsistensi (*consistency*)**. Diharapkan aksi yang sama pada situasi yang berbeda akan melakukan hal yang sama pula. Sebagai contoh, user terbiasa dengan tombol ‘tab’ yang menyebabkan kursor bergeser ke kanan sebanyak 8 karakter. Pada penggunaan dialog box, tombol ‘tab’ menyebabkan kursor bergeser ke dialog box berikutnya. Konsistensi ini tidak mudah diperiksa karena dua hal, yaitu pertama, konsistensi melibatkan arti / semantik bahwa aksi yang serupa

(*similar*) melakukan hal yang sama. Kedua, interpretasi serupa (*similar*) tergantung dari pemahaman user mengenai batasan arti *similar*.

State Properties

State pada dialog merepresentasikan titik saat user memperoleh informasi atau sistem telah melakukan suatu hal. Sehingga user menginginkan paling tidak dapat mencapai suatu state yang diinginkan atau idealnya dapat mencapai state tersebut dengan mudah. Karakteristik state ini dikategorikan sebagai ***reachability***. Untuk memastikan karakter ini tercakup pada dialog, pemeriksaan dasar yang dapat dilakukan adalah dengan melihat apakah rangkaian notasi dialog terkoneksi penuh (*fully connected*). Sebuah notasi dialog terkoneksi penuh jika diantara dua buah state terdapat serangkaian aksi yang akan membawa user dari state pertama ke state kedua.

Kasus khusus dari *reachability* adalah *reversability*, yaitu user dapat kembali ke keadaan semula. Bentuk button atau command “undo” mungkin merupakan salah satu metode *reversability* yang paling baik. Satu cara untuk melakukan analisis *reversability* adalah dengan melihat setiap aksi kemudian diberi label dengan jumlah *edge* / garis yang harus dikunjungi untuk kembali ke keadaan semula. Untuk aksi yang memiliki jumlah / biaya kembali (reversing) yang besar mungkin harus didesain ulang dialognya.

Dari keterangan di atas, beberapa state ingin dikunjungi oleh user (*desired state*) dengan mudah, namun ada beberapa state lainnya yang harus dihindari (*dangerous state*) dan harus dibuat dengan sengaja utnuk sulit dijangkau. Namun pengecekan *dangerous state* ini juga tidak dapat dilakukan secara otomatis. Hal ini karena cakupan *dangerous state* bergantung pada semantik sistem dan penilaian desainer. Kita dapat melakukan analisis terhadap seberapa mudahnya *dangerous state* dicapai dengan melabeli seluruh state tersebut dengan warna merah. Jika ditemukan sebuah *dangerous state* sangat mudah untuk dicapai, maka dapat dilakukan suatu aksi untuk mencegah terjadinya kesalahan terutama yang tidak disengaja. Banyak sistem yang melakukan dialog inisiatif mengkonfirmasikan tindakan user jika hal tersebut berakibat fatal, misalnya :

```
C> del test \  * . *
Are you sure (Y/N) ?
```

Perpindahan antara state biasa dengan *dangerous state* harus diperhatikan dan diuji dengan detail oleh desainer. Desainer harus dapat menentukan jalur mana yang dapat mengakibatkan kesalahan dan yang tidak.

Presentation and Lexical Properties

Umumnya perancangan dialog dikatakan harus terpisah (*independent*) dari perancangan detail dari presentasi dan leksikal interface. Seorang desainer harus menentukan fungsi sistem terlebih dahulu kemudian menggunakan model kognitif atau analisis tugas untuk mendesain dialog untuk menjalankan fungsi tersebut. Baru akhirnya mendesain presentasi sistem secara visual dan leksikal interface antara tombol yang ditekan dan mouse yang digerakan dengan aksi dialog yang abstrak.

Pada bagian sebelumnya, kita telah membahas mengenai pelabelan *mode* (situasi - misalnya *silent mode*) dimana aksi user mungkin memiliki interpretasi yang berbeda. Pada prakteknya, akan lebih baik jika dapat meminimalkan jumlah *mode* yang ada. Namun biar bagaimanapun, mode yang ada harus dapat dibedakan secara visual. Hal ini berarti user dapat membedakan pada mode apa dia berada sekarang dari display komputer. Sehingga ada keterkaitan antara presentasi visual dengan dialog. Jika user tidak dapat memantau mode keberadaan user, maka akan muncul kesulitan yang dapat membingungkan user. Hal ini merupakan persyaratan ***visibility***.

Tipe *interface* yang berbeda memiliki tipe dialog yang berbeda pula. Sebagai contoh, umumnya *interface* berbasis *command* akan memiliki gaya *verb-object* seperti “print essay”. Namun sistem berbasis mouse umumnya memiliki gaya sintaks object-verb seperti “select a file icon and then select ‘print’ from a menu”. Meskipun dimungkinkan untuk menggabungkan keduanya. Sehingga dapat disimpulkan bahwa desain dialog detail bergantung pada tipe interface.

Lebih jauh lagi keterbatasan fisik dapat membatasi struktur dialog. Sebagai contoh, dialog untuk *digital watch* yang didesain dengan tiga tombol berbeda dengan dialog untuk on-screen alarm clock yang dapat menggunakan seluruh keyboard. Demikian pula keterbatasan jangkauan output, visual dan aural juga akan membatasi dialog. Jika mode dan state harus dapat dibedakan secara visual, maka display dari device juga harus dapat membedakan mode yang aktif.

Latihan :

Suatu perusahaan minuman akan mengeluarkan mesin penjual minuman yang nantinya akan diletakkan di beberapa tempat umum, seperti bandara, stasiun KA.
Untuk itu dibentuk satu tim yang terdiri dari ahli elektronik untuk menangani masalah hardware dan ahli interface designer untuk menangani masalah interface.
Para interface designer benar-benar dituntut untuk mendesain suatu interface yang baik dan mudah dimengerti karena hasil penjualan sangat tergantung dari hal tersebut.

Konsumen tidak akan membeli minuman lewat mesin tersebut apabila mereka mengalami kesulitan atau kebingungan saat akan melakukan transaksi.
Sekarang anda diminta bertindak sebagai interface designer yang mendesain dialog antara konsumen dan mesin penjual minuman.