MODEL USER DALAM DESAIN

Tujuan: Menerangkan beberapa model yang dapat digunakan selama proses desain interface

Dalam berbagai disiplin ilmu, model sering digunakan dalam proses desain. Model dapat bersifat:

- **Evaluative (mengevaluasi desain yang ada)**
- **Generative (mempunyai kontribusi pada proses desain)**

Pada prakteknya, model yang sering digunakan adalah yang bersifat generative.

MODEL KOGNITIF

Presentasi model kognitif dibagi dalam kategori:

- **Representasi hirarki tugas (task) user dan struktur goal, formulasi goal dan tugas**
- **Model linguistik dan gramatik**
  Grammar dari translasi artikulasi dan bagaimana pemahamannya oleh user
- **Model tingkat device dan fisik (artikulasi pada tingkat motorik manusia)**
  artikulasi tingkat motorik manusia dan bukan tingkat pemahaman manusia

Hirarki Tugas dan Goal

Banyak model menggunakan model pemrosesan mental dimana user mencapai goal dengan menemukan sub-goal dengan cara divide-and-conquer.

Model yang akan dibahas:

- **GOMS(Goals, Operators, Methods and Selections)**
- **CCT (Cognitive Complexity Theory)**

Contoh: Membuat laporan penjualan Buku IMK

Procedure report
    Gather data
        Find book names
            Do keywords search of name database
                <<further sub-goals>>
            shift through names & abstracts by hand
                <<further sub-goals>>
        search sales database
            <<further sub-goals>>
    layout tables and histograms
        <<further sub-goals>>
    Write description
        <<further sub-goals>>

Beberapa isu penting :

- **Dimana kita berhenti (dlm mendekomposisi tugas)**
- **Dimana kita memulai (analisa pada hirarki goal tertentu)**
- **Apa yang harus dilakukan, ketika ada beberapa solusi**
- **Apa yang harus dilakukan terhadap error yang terjadi**

GOMS

- **Goal; goal apa yang ingin dicapai oleh user**
- **Operator; level terendah analisa, tindakan dasar yang harus dilakukan user dalam menggunakan sistem**
- **Methods; Ada beberapa cara yang dilakukan dimana memisahkan kedalam beberapa subgoals**
  Contoh: pada window manager, perintah CLOSE dapat dilakukan dengan menggunakan popup menu atau hotkey
- **Selection; Pilihan terhadap metode yang ada**

Analisa GOMS umumnya terdiri dari single high-level goal, kemudian didekomposisi menjadi deretan unit task, selanjutnya dapat didekomposisi lagi sampai pada level operator dasar

Dimana dalam membuat dekomposisi tugas digunakan *hierarchical task analysis* (HTA).

Analisa struktur goal GOMS dapat digunakan untuk mengukur kinerja. Kedalaman tumpukan struktur goal dapat digunakan untuk mengestimasi kebutuhan memori jangka-pendek.

Cognitive Complexity Theory (CCT)

- **CCT (Kieras dan Polson) dimulai dengan premis dasar dekomposisi goal dari GOMS dan menyempurnakan model untuk menghasilkan kekuatan yang lebih terprediksi.**

- **Deskripsi goal user berdasarkan hirarki goal mirip-GOMS, tetapi diekspresikan terutama menggunakan *production rules* yang merupakan urutan rules:**

  If *kondisi* then *aksi*

Dimana kondisi adalah pernyataan tentang isi dari memori kerja. Aksi dapat terdiri satu atau lebih aksi elementary.

Contoh:
Tugas editing menggunakan editor 'vi' UNIX.
Tugasnya mengoreksi spasi antar kata.

Production rules CCT:

```
(SELECT-INSERT-SPACE
IF (AND (TEST-GOAL perform unit task)
        (TEST-TEXT task is insert space)
        (NOT (TEST-GOAL insert space))
        (NOT (TEST-NOTE executing insert space)))
THEN (  (ADD-GOAL insert space)
        (ADD-NOTE executing insert space)
        (LOOK-TEXT task is at %LINE %COL)))

(INSERT-SPACE-DONE
IF (AND (TEST-GOAL perform unit task)
        (TEST-NOTE executing insert space)
        (NOT (TEST-GOAL insert space)))
THEN (  (DELETE-NOTE executing insert space)
        (DELETE-GOAL perform unit task)
        (UNBIND %LINE %COL))

(INSERT SPACE-1
IF (AND (TEST-GOAL insert space)
        (NOT (TEST-GOAL move cursor))
        (NOT (TEST-CURSOR %LINE %COL)))
THEN (  (ADD-GOAL move cursor to %LINE %COL)))

(INSERT SPACE-2
IF (AND (TEST-GOAL insert space)
        (TEST-CURSOR %LINE %COL))
THEN (  (DO-KEYSTROKE 'I')
        (DO-KEYSTROKE  SPACE)
        (DO-KEYSTROKE  ESC)
        (DELETE-GOAL insert space)))
```

Untuk mengetahui cara kerja rules, anggap user baru saja melihat ketikan yang salah dan isi dari memori kerja adalah :

```
(GOAL perform unit task)
(TEXT task is insert space)
(TEXT task is at 5 23)
(CURSOR 8 7)
```

Isi memori kerja setelah rule SELECT-INSERT-SPACE di fire :

```
(GOAL perform unit task)
(TEXT task is insert space)
(TEXT task is at 5 23)
(NOTE executing insert space)
(GOAL insert space)
```

(LINE 5)
(COL 23)
(CURSOR 8 7)

- **Rule dalam CCT dapat digunakan untuk menerangkan fenomena error, tetapi tidak dapat memprediksi**
  Contoh: rule untuk menginsert space tidak mengecek modus editor yang digunakan
- **Semakin banyak production rules dalam CCT semakin sulit suatu interface untuk dipelajari**

Problem CCT:
- **Semakin detail deskripsinya, size deskripsi dapat menjadi sangat besar**
- **Pemilihan notasi yang digunakan**
  Contoh: pada deskripsi sebelumnya (NOTE executing insert space) hanya digunakan untuk membuat rule INSERT-SPACE-DONE di fire pada waktu yang tepat. Di sini tidak jelas sama sekali signifikansi kognitifnya
- **CCT adalah engineering tool dengan pengukuran singkat learnability dan difficulty digabung dengan dekripsi detail dari user behaviour.**

MODEL LINGUISTIK

Interaksi user dengan komputer dapat dipandang dari segi language, beberapa formalisasi model menggunakan konsep ini. Grammar BNF paling sering digunakan untuk melakukan dialog.

Backus-Naur Form (BNF)
- **Memandang dialog pada level sintaksis, mengabaikan semantik dari bahasa tersebut.**

Contoh: Fungsi menggambar garis pada sistem grafik

draw-line	::=	select-line + choose-points + last-point
select-line	::=	position-mouse + CLICK-MOUSE
choose-points	::=	choose-one | choose-one + choose-points
choose-one	::=	position-mouse + CLICK-MOUSE
last-point	::=	position-mouse + DOUBLE-CLICK-MOUSE
position-mouse	::=	empty | MOVE-MOUSE + position-mouse

- **Non-terminals (huruf kecil) adalah abstraksi level tinggi dimana dapat terdiri dari non-terminal lainnya dan terminal dalam format:**
  name ::= expression
- **Terminals (huruf besar), merepresentasikan level terendah dari user behaviour**
- **Operator '+' adalah sequence, '|' adalah choice**

- **Deskripsi BNF dapat dianalisa dengan mengukur jumlah rules dan operatornya**
- **Pengukuran kompleksitas untuk bahasa secara keseluruhan, BNF dapat digunakan untuk menentukan berapa banyak tindakan dasar yang dibutuhkan dalam tugas tertentu, dan mendapatkan estimasi kasar kesulitan (difficulty) dari tugas**

Task-Action Grammar (TAG)

- **BNF mengabaikan kelebihan konsistensi dalam struktur language dan dalam menggunakan nama perintah**

Contoh:
3 UNIX command:

copy ::= ‘cp’ + filename + filename | ‘cp’ + filename + directory
move ::= ‘mv’ + filename + filename | ‘mv’ + filename + directory
link ::= ‘ln’ + filename + filename | ‘ln’ + filename + directory

BNF tidak dapat membedakan konsistensi dan inkonsistensi command (misal: ln mengambil argumen direktori lebih dahulu). Dengan TAG dapat diatasi dengan mengubah deskripsinya :

File-op [Op] := command-op[Op]+ filename + filename
| command-op[Op] + filename + directory
command-op[Op=copy] := ‘cp’
command-op[Op=move] := ‘mv’
command-op[Op=link] := ‘ln’

- **TAG mengatasi masalah ini dengan menyertakan parametrized grammar rules untuk konsistensi dan pengetahuan umum user (seperti atas lawan dari bawah)**

Contoh: Dua command line interface untuk menggerakkan robot di atas lantai

Command interface 1

movement [Direction] := command[Direction] + distance + RETURN
command[Direction=forward] := ‘go 395’
command[Direction=backward] := ‘go 013’
command[Direction=left] := ‘go 712’
command[Direction=right] := ‘go 956’

Command interface 2

movement [Direction] := command[Direction] + distance + RETURN
command[Direction=forward] := ‘FORWARD’
command[Direction=backward] := ‘BACKWARD’
command[Direction=left] := ‘LEFT’
command[Direction=right] := ‘RIGHT’

Interface kedua lebih komunikatif. TAG menambahkan form khusus known-item yang digunakan untuk menginformasikan ke user bahwa inputnya sudah diketahui secara umum. Interface kedua dapat ditulis ulang sebagai berikut:

Command interface 2

movement [Direction] := command[Direction] + distance + RETURN
command[Direction] := known-item[Type = word, Direction]
command[Direction=forward] := 'FORWARD'
command[Direction=backward] := 'BACKWARD'
command[Direction=left] := 'LEFT'
command[Direction=right] := 'RIGHT'

MODEL FISIK DAN DEVICE

Keystroke Level Model (KLM)

- **Tugas dapat didekomposisi menjadi dua fase:**
  - **Akuisisi tugas, ketika user membangun representasi mental dari tugas**
  - **Execution tugas menggunakan fasilitas sistem**
- **KLM hanya memberikan prediksi untuk kegiatan pada tahap berikutnya**
- **KLM merupakan bentuk model GOMS tingkat terendah**
- **Model mendekomposisi fase eksekusi menjadi operator motor-fisik, operator mental dan operator respons**
  - **K keystroking**
  - **B menekan tombol mouse**
  - **P pointing, menggerakkan mouse (atau sejenis) ke target**
  - **H Homing, perpindahan tangan antar mouse dan keyboard**
  - **D menggambar garis dengan mouse**
  - **M persiapan mental untuk tindakan fisik**
  - **R respon sistem, dapat diabaikan jika user tidak perlu menunggu untuk itu**

Contoh : Mengedit karakter tunggal yang salah

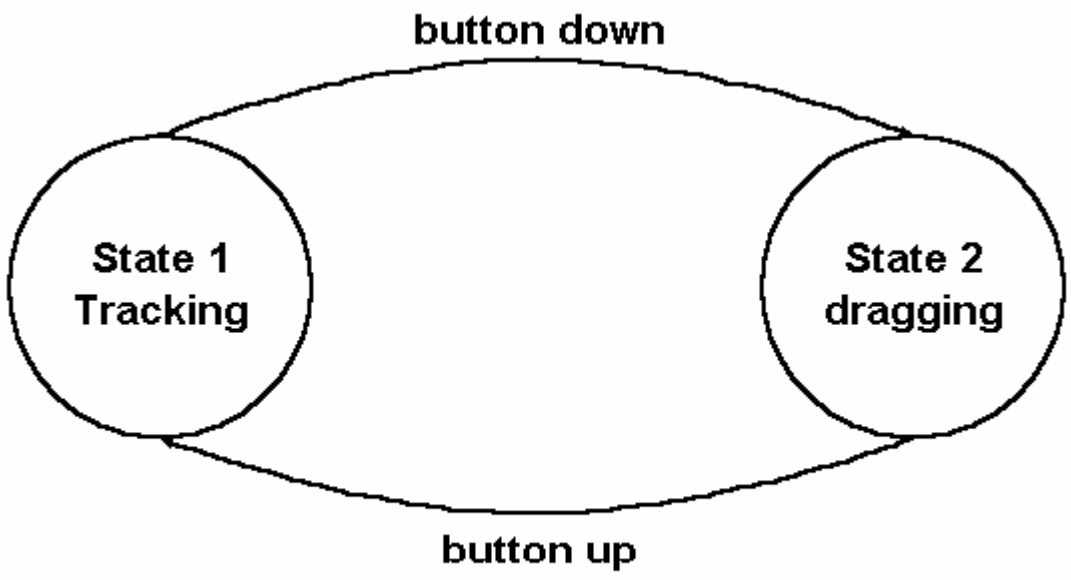
1. **memindahkan tangan ke mouse H[mouse]**
2. **Meletakkan cursor setelah karakter yang salah PB[LEFT]**
3. **Kembali ke keyboard H[keyboard]**
4. **Hapus kerakter MK[DELETE]**
5. **Ketik koreksi K[char]**
6. **Mereposisi ke insertion point H[mouse]MPB[LEFT]**

Waktu yang dibutuhkan:

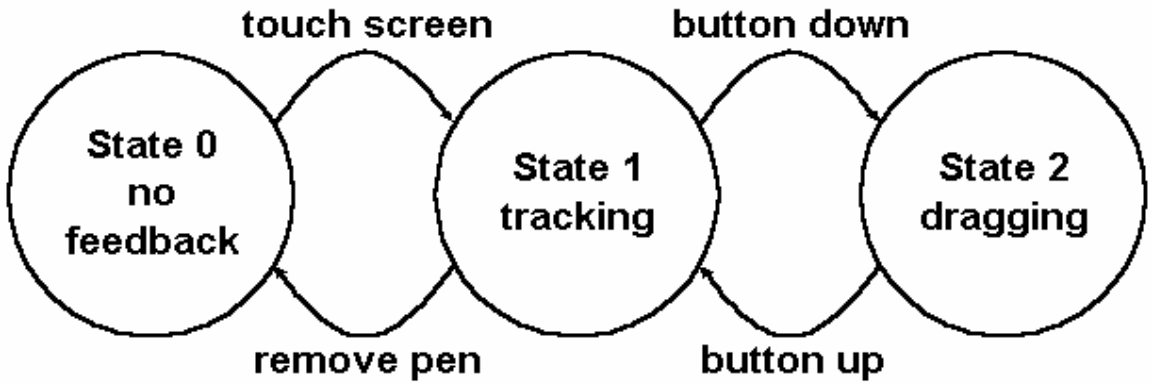
$$T_{execute} = T_K + T_B + T_P + T_H + T_D + T_M + T_R$$
$$= 2t_K + 2t_B + t_P + 3t_H + 0 + t_M + 0$$

Three-State Model

Ada berbagai macam device penunjuk yang digunakan selain mouse. Device biasanya dapat dinyatakan equivalen secara logika (dilihat dari level aplikasi), tetapi dilihat dari karakteristik motor-sensor fisiknya berbeda. Oleh karena itu three-state model dibuat untuk mewakili device tersebut



Mouse transitions: state 1 and 2



Lightpen transitions: three states

ARSITEKTUR KOGNITIF

Asumsi arsitektural yang mendasari permodelan kognitif

Problem Space Model

Dalam ilmu komputer, problem biasanya dijabarkan sebagai pencarian ke setiap state yang memungkinkan dari beberapa state awal ke state goal, keseluruhan state ini berikut transisinya biasa juga disebut state space. Proses pencarian solusi biasanya disebut Problem space.
Setelah problem diidentifikasi dan sampai pada solusi (algoritma), programmer kemudian merepresentasikan problem dan algoritma ke dalam bahasa pmrograman yang dapat dieksekusi pada mesin untuk mencapai state yang diinginkan.

Interactive Cognitive Sub-systems (ICS)

ICS membentuk sebuah model dari persepsi kognitif dan aksi. ICS memandang user sebagai mesin pemroses informasi. Penekanannya dalam menentukan kemudahan melaksanakan prosedur tindakan tertentu dengan membuatnya lebih mudah dilaksanakan di dalam user itu sendiri.

ICS menggunakan dua tradisi psikologi yang berbeda didalam satu arsitektur kognitif. Pertama pendekatan arsitektural dan general-purpose information processing, kedua, karakteristik pendekatan komputasional dan representasional.

Arsitektur ICS dibangun dengan mengkoordinasikan sembilan sub-system yang lebih kecil: lima sub-system periferal yang berkontak langsung secara fisik dan empat adalah sentral, yang menyangkut pemrosesan mental.