Motivation

- From task analysis to formal description and prediction of interaction
- GOMS is a task analytic notation for procedural knowledge
  - Syntax and semantics similar to a programming language
  - Assumes a simplified cognitive architecture (the Model Human Processor from previous lectures)
  - Can be executed in simulation (production rule system)
  - Static and run-time properties can provide quantitative predictions of usability, e.g.:
    - Time to complete task
    - Complexity/difficulty/ knowledge requirements
    - Short term memory load
    - Novice vs. expert behaviour, rate of learning
    - Effects on all these of changing the interface
- “Programmable User Models” (Young et al 1989)

GOMS (Goals, Operators, Methods and Selection rules)

- GOMS model proposed by Card et. al (1983)
- Each user task is described by a goal and a method
- A method is a sequence of steps, each consisting of one or more operators
- Can have more than one method for a task, in which case need a selection rule

Example

GOAL: DELETE SENTENCE

Method for goal: MENU-METHOD-DELETE SENTENCE
  Step 1: HIGHLIGHT SENTENCE
  Step 2: OPEN MENU
  Step 3: SELECT DELETE-COMMAND
  Step 4: Accomplish_goal MENU-METHOD-DELETE SENTENCE

Method for goal: DEL-KEY-METHOD-DELETE SENTENCE
  Step 1: POSITION-CURSOR AT END
  Step 2: PRESS DELETE FOR EACH LETTER
  Step 3: Accomplish_goal DEL-KEY-METHOD-DELETE SENTENCE

Selection rules for goal: DELETE SENTENCE
If [long sentence] Then Accomplish_goal: MENU-METHOD-DELETE SENTENCE
If [short sentence] Then Accomplish_goal: DEL-KEY-METHOD-DELETE SENTENCE

Note: some operators are for task flow handling
Analysis is developed through hierarchical goal decomposition

- Start with highest level goals: what the user is trying to accomplish in the application domain
- Provide a method in terms of high-level operators
- Each operator can be potentially decomposed:
  - Replace operator with equivalent sub-goal, e.g.,
    - Step 2: OPEN MENU → Step 2: Accomplish_goal: OPEN MENU
  - Specify a method for the sub-goal:
    - Method_for_goal: OPEN MENU
      - Step 1: MOVE-CURSOR MENU-BAR
      - Step 2: CLICK MENU-NAME
      - Step 3: Accomplish_goal OPEN MENU

Decomposition can continue until reach primitive operators, which have an associated execution time:

- Primitive physical operators:
  - Keystroke key_name (100-1000ms depending on typing skill)
  - Press mouse_button (100ms)
  - Release mouse_button (100ms)
  - Point_to target (Use Fitt’s law if target size & distance known, else default value of 1100ms)
  - Home_to destination (Moving hands to keyboard or to mouse, 400ms)
  - Look_for object (Visual search...)
- Primitive mental operators: (default assumption 1200ms)
  - Decide conditional
  - Recall item from long term memory
  - Verify selection

A low-level GOMS analysis, down to primitive operators, is sometimes called a ‘Keystroke Level Model’

- Can use directly to estimate the relative execution time for different methods
- Depth of hierarchical goal structure reflects complexity and memory load of task
- Can predict likely sequence of actions (but not mistakes)
- Most appropriate for:
  - Goal-directed interactions (e.g. not browsing)
  - Routine skill performance (e.g. not discovery)
  - Sequential instructional interactions
  - Capturing internal procedural knowledge
- Can use to validate hypotheses about interaction processes
  - e.g., when searching a menu, is each item considered before eyes move to next, or do eyes keep scanning ahead of decision process?

<table>
<thead>
<tr>
<th>DEL-KEY-METHOD</th>
<th>TIME</th>
<th>MENU-METHOD</th>
<th>TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mentally prepare</td>
<td></td>
<td>Mentally prepare</td>
<td></td>
</tr>
<tr>
<td>Reach for mouse</td>
<td></td>
<td>Reach for mouse</td>
<td></td>
</tr>
<tr>
<td>Move cursor to end of word</td>
<td></td>
<td>Move cursor to front of word</td>
<td></td>
</tr>
<tr>
<td>Click mouse (down-up)</td>
<td></td>
<td>Mouse down</td>
<td></td>
</tr>
<tr>
<td>Move finger to delete key</td>
<td></td>
<td>Drag to end of word</td>
<td></td>
</tr>
<tr>
<td>Press delete key n times</td>
<td></td>
<td>Mouse up</td>
<td></td>
</tr>
</tbody>
</table>

TOTAL=

How many letters in a word before it becomes faster to use the menu?
**Example: applied to menu search – serial**

```
(IF NOT-TARGET-THEN-SACCADE-ONE-ITEM
  IF (GOAL-DO-MENU-TASK)
  (STEP-VISUAL-SEARCH)
  (WM CURRENT-ITEM IS OBJECT)
  (VISUAL-VECTOR IS ABOVE TARGET-OBJECT)
  (NOT (VISUAL OBJECT IS ABOVE NOTHING))
  (MOTOR OCULAR PROCESSOR FREE)
  (VISUAL OBJECT LABEL NOT)
  (NOT (WM TARGET-TEXT IS NOT))
  THEN
  (DELAYS (WM CURRENT-ITEM IS OBJECT))
  (ADJUST (WM CURRENT-ITEM IS PRECEDING-OBJECT))
  (SEND-TO-MOTOR OCULAR MOVE PRECEDING-OBJECT))
(TARGET-IS-LOCATED-EDITION-MOVING-MOUSE
  IF (GOAL-DO-MENU-TASK)
  (STEP-VISUAL-SEARCH)
  (WM TARGET-TEXT IS NOT)
  (WM CURSOR IS (CURSOR-OBJECT))
  (MOTOR MANUAL PROCESSOR FREE)
  THEN
  (DELAYS (STEP VISION-SEARCH))
  (ADJUST (STEP NAIVE RESPONSE))
  (SEND-TO-MOTOR MANUAL PERFORM PLUG MOUSE
   RIGHT ZERO-ORDER-CORRELATION CURSOR-OBJECT TARGET-OBJECT)))
```

**Example: applied to menu search – overlapping**

```
(SACCADE-ONE-ITEM
  IF (GOAL-DO-MENU-TASK)
  (STOP-SEARCH)
  (STEP-VISUAL-SEARCH)
  (WM CURRENT-ITEM IS OBJECT)
  (VISUAL-VECTOR IS ABOVE TARGET-OBJECT)
  (NOT (VISUAL OBJECT IS ABOVE NOTHING))
  (MOTOR OCULAR PROCESSOR FREE)
  (VISUAL OBJECT LABEL NOT)
  (NOT (WM TARGET-TEXT IS NOT))
  THEN
  (DELAYS (STEP VISION-SEARCH))
  (ADJUST (WM CURRENT-ITEM IS OBJECT))
  (ADJUST (WM CURRENT-ITEM IS PRECEDING-OBJECT))
  (SEND-TO-MOTOR OCULAR MOVE PRECEDING-OBJECT))
(MOVE-GAZE-AND-CURSOR-TO-TARGET
  IF (GOAL-DO-MENU-TASK)
  (STOP-SEARCH)
  (STEP-VISUAL-SEARCH)
  (WM TARGET-TEXT IS NOT)
  (WM CURSOR IS (CURSOR-OBJECT))
  (MOTOR OCULAR PROCESSOR FREE)
  (MOTOR MANUAL PROCESSOR FREE)
  THEN
  (DELAYS (STEP MOVE-GAZE-AND-CURSOR-TO-TARGET))
  (ADJUST (STEP NAIVE RESPONSE))
  (SEND-TO-MOTOR OCULAR MOVE TARGET-OBJECT)
  (SEND-TO-MOTOR MANUAL PERFORM PLUG MOUSE
   RIGHT ZERO-ORDER-CORRELATION CURSOR-OBJECT TARGET-OBJECT)))
```

---

**CPM-GOMS**

- Recall the ‘Model Human Processor’
- Cognitive, Perceptual, Motor (CPM-) GOMS recognises that the subsystems can work in parallel, subject to constraints of information flow
- Total time to do a task will depend on these interdependencies
CPM-GOMS

- Expressed as a PERT chart, e.g. "slow move click"

![PERT Chart Diagram]

- From Gray and Boehm-Davis (2001)

CPM-GOMS + APEX

- Can store common actions (like mouse click) as templates
- For higher level task, can link together series of templates showing interdependencies
- Can then calculate total time for 'critical path'

![APEX Diagram]

- Still difficult to do by hand, particularly if want to interlace subtasks (i.e. allow later tasks to use free resources)

- Automated approach (John et al 2002) comes from using reactive planner architecture APEX
  - Limited parallel resources (e.g. perceptual, motor, cognitive)
  - Procedure description language (= methods & operations)
  - Selection operator
  - Specify preconditions (task A must complete before task B can start) or priority (if task C and D are competing for same resource)

- APEX then generates the sequencing, with appropriate interleaving
  - Dynamic task scheduling, i.e. can respond to environmental change
  - Precoded low level procedures can be called by high level (the designer does not need to know the psychological details)

<pre>
(procedure
  (index (slow move click ?target))
  (step c1 (initiate move cursor ?target))
  (step m1 (move cursor ?target) (waitfor ?c1))
  (step c2 (attend target ?target))
  (step c3 (initiate eye movement ?target) (waitfor ?c2))
  (step m2 (eye movement ?target) (waitfor ?c3))
  (step p1 (perceive target complex ?target) (waitfor ?m2))
  (step c4 (verify target position ?target) (waitfor ?c3 ?p1))
  (step c5 (attend cursor at target ?target) (waitfor ?c4))
  (step w1 (world new cursor location ?target) (waitfor ?m1))
  (step p2 (perceive cursor at target ?target) (waitfor ?p1 ?c5 ?w1))
  (step c6 (verify cursor at target ?target) (waitfor ?c5 ?p2))
  (step c7 (initiate click ?target) (waitfor ?c6 ?m1))
  (step m3 (mouse down ?target) (waitfor ?m1 ?c7))
  (step m4 (move up ?target) (waitfor ?m3))
  (step t1 (terminate) (waitfor ?m4))
</pre>
Example: ATM withdrawal

Models fits well with actual time data (for practiced performers)

EPIC (Executive Process-Interactive Control) (Kieras and Meyer, 1995)

- EPIC architecture includes more sensorimotor detail

For visual processing, models field of view and eye movements;
Knowledge of different object properties depends on location of target (e.g. only know text content in fovea)
EPIC

Used to model visual search for different layouts (n.b. guided vs. bottom-up attention) and compared to eye-tracking results (Hornof & Halverson, 2003)

Some limitations

- GOMS analysis and production rule systems tend to get very complex even for simple tasks
  - May reflect true complexity of underlying processes, but high effort in construction
  - May need user comparison anyway to fix parameters & verify
  - Some more approximate methods may be more useful
  - Some successful applications but more promise than delivery
- These models are usually not generative
  - Constructed to fit and evaluate a given interface, rather than helping in original design of interface
- The psychological reality of production rules is questionable
  - Creating artificial user in computer’s image – is this really “user centred design”?

EPIC

Results supply concrete justification for a number of layout recommendations:
- People use hierarchy and will focus on one level at a time: so support search with salient labels and ease of eye movements between them, e.g. by use of white space
- Can examine more than one item in a fixation: hence use vertical lists and left justify
- Ocular motor system is primed for (or even anticipates) response to visual onset: added reason why slow response times are annoying for users.

References


See also:
Dix et. al. sections 12.1, 12.2, 12.5