HCI Lecture 6:

Formal models I: GOMS

Barbara Webb

Key points:

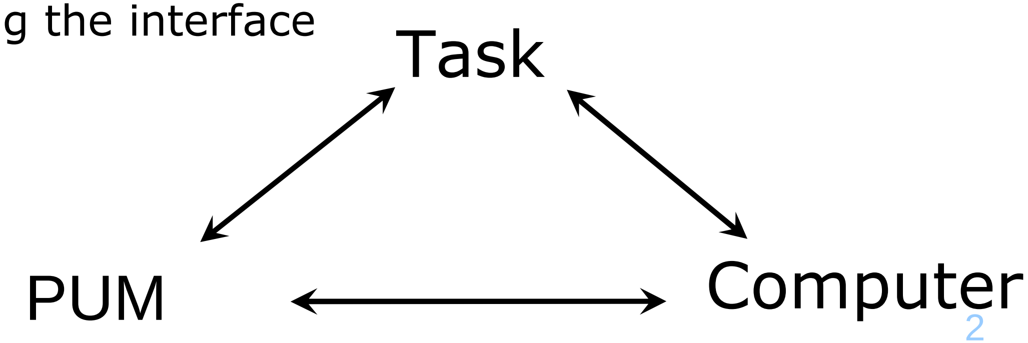
- Motivation
- GOMS – Goals, Operators, Methods, Selection rules
- KLM – Keystroke Level Model
- CPM-GOMS – Cognitive, Perceptual, Motor
- EPIC
- Limitations

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

- 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

Note: some operators are for task flow handling

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

GOMS

- 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

Operators

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*

Keystroke Level Model (KLM)

- 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?

- Alternative methods to delete a word:

DEL-KEY-METHOD	**TIME**	***MENU-METHOD***	**TIME**
Mentally prepare		Mentally prepare	
Reach for mouse		Reach for mouse	
Move cursor to end of word		Move cursor to front of word	
Click mouse (down-up)		Mouse down	
Move finger to delete key		Drag to end of word	
Press delete key n times		Mouse up	
TOTAL=		Recall menu location	
		Move to open menu	
		Mouse down	
		Scroll to 'delete'	
		Mouse up	
		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 ?OBJECT IS-ABOVE ?NEXT-OBJECT)
 (NOT (VISUAL ?OBJECT IS-ABOVE NOTHING))
 (MOTOR OCULAR PROCESSOR FREE)
 (VISUAL ?OBJECT LABEL ?NT)
 (NOT (WM TARGET-TEXT IS ?NT)))
THEN
((DELDB (WM CURRENT-ITEM IS ?OBJECT))
 (ADDDB (WM CURRENT-ITEM IS ?NEXT-OBJECT))
 (SEND-TO-MOTOR OCULAR MOVE ?NEXT-OBJECT)))

(TARGET-IS-LOCATED-BEGIN-MOVING-MOUSE
IF
((GOAL DO MENU TASK)
 (STEP VISUAL-SEARCH)
 (WM TARGET-TEXT IS ?T)
 (VISUAL ?TARGET-OBJECT LABEL ?T)
 (WM CURSOR IS ?CURSOR-OBJECT)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((DELDB (STEP VISUAL-SEARCH))
 (ADDDB (STEP MAKE RESPONSE))
 (SEND-TO-MOTOR MANUAL PERFORM PLY MOUSE
       RIGHT ZERO-ORDER-CONTROL ?CURSOR-OBJECT ?TARGET-OBJECT)))
```

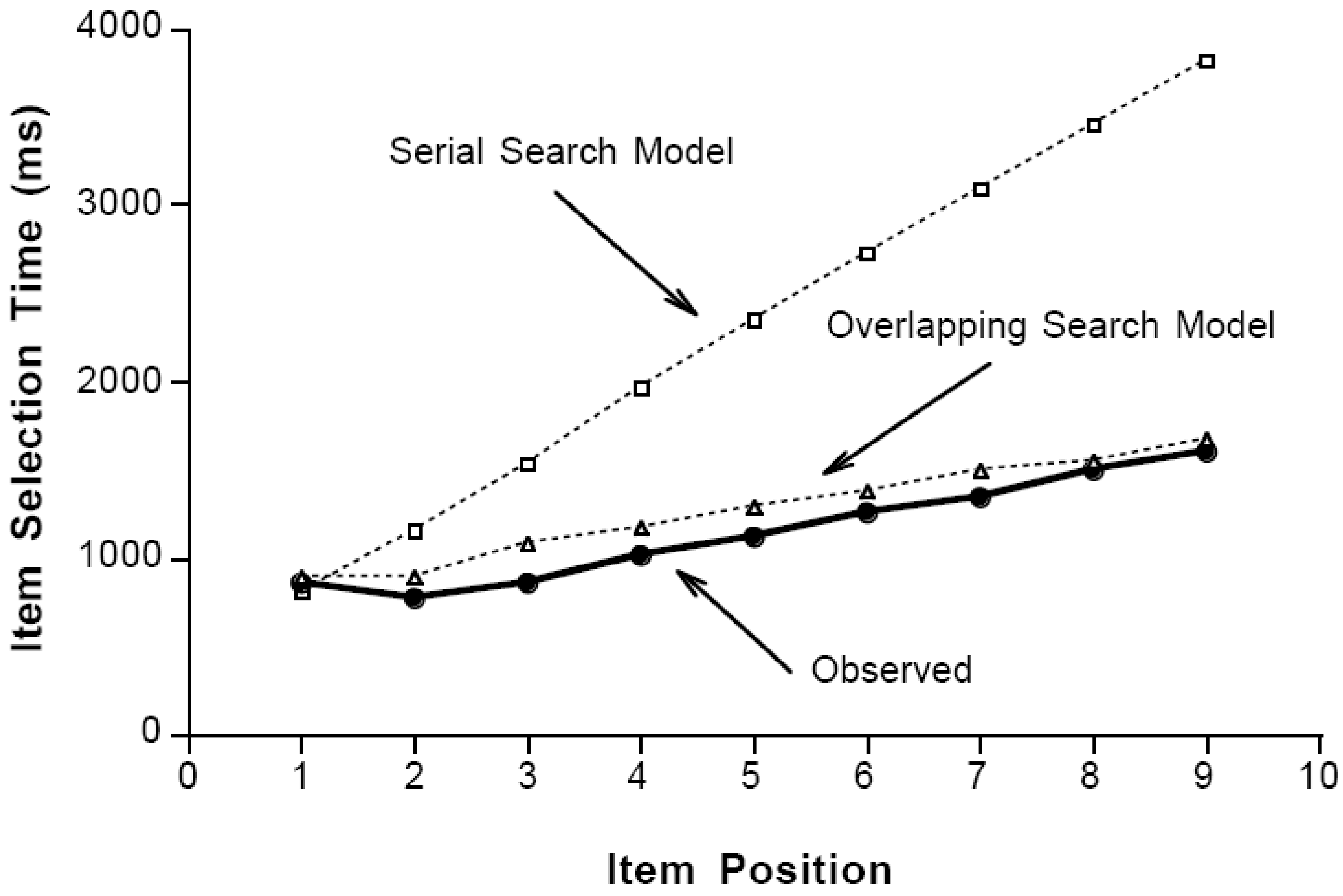
Example: applied to menu search - overlapping

```
(SACCADE-ONE-ITEM
IF
((GOAL DO MENU TASK)
 (STEP VISUAL-SWEEP)
 (WM CURRENT-ITEM IS ?OBJECT)
 (VISUAL ?OBJECT IS-ABOVE ?NEXT-OBJECT)
 (NOT (VISUAL ?OBJECT IS-ABOVE NOTHING))
 (MOTOR OCULAR PROCESSOR FREE)
 )
THEN
((DELDB (WM CURRENT-ITEM IS ?OBJECT))
 (ADDDB (WM CURRENT-ITEM IS ?NEXT-OBJECT))
 (SEND-TO-MOTOR OCULAR MOVE ?NEXT-OBJECT)))
```

```
(STOP-SCANNING
IF
((GOAL DO MENU TASK)
 (STEP VISUAL-SWEEP)
 (WM TARGET-TEXT IS ?T)
 (VISUAL ?TARGET-OBJECT LABEL ?T))
THEN
((DELDB (STEP VISUAL-SWEEP))
 (ADDDB (STEP MOVE-GAZE-AND-CURSOR-TO-TARGET))
 (ADDDB (WM TARGET-OBJECT IS ?TARGET-OBJECT)))
```

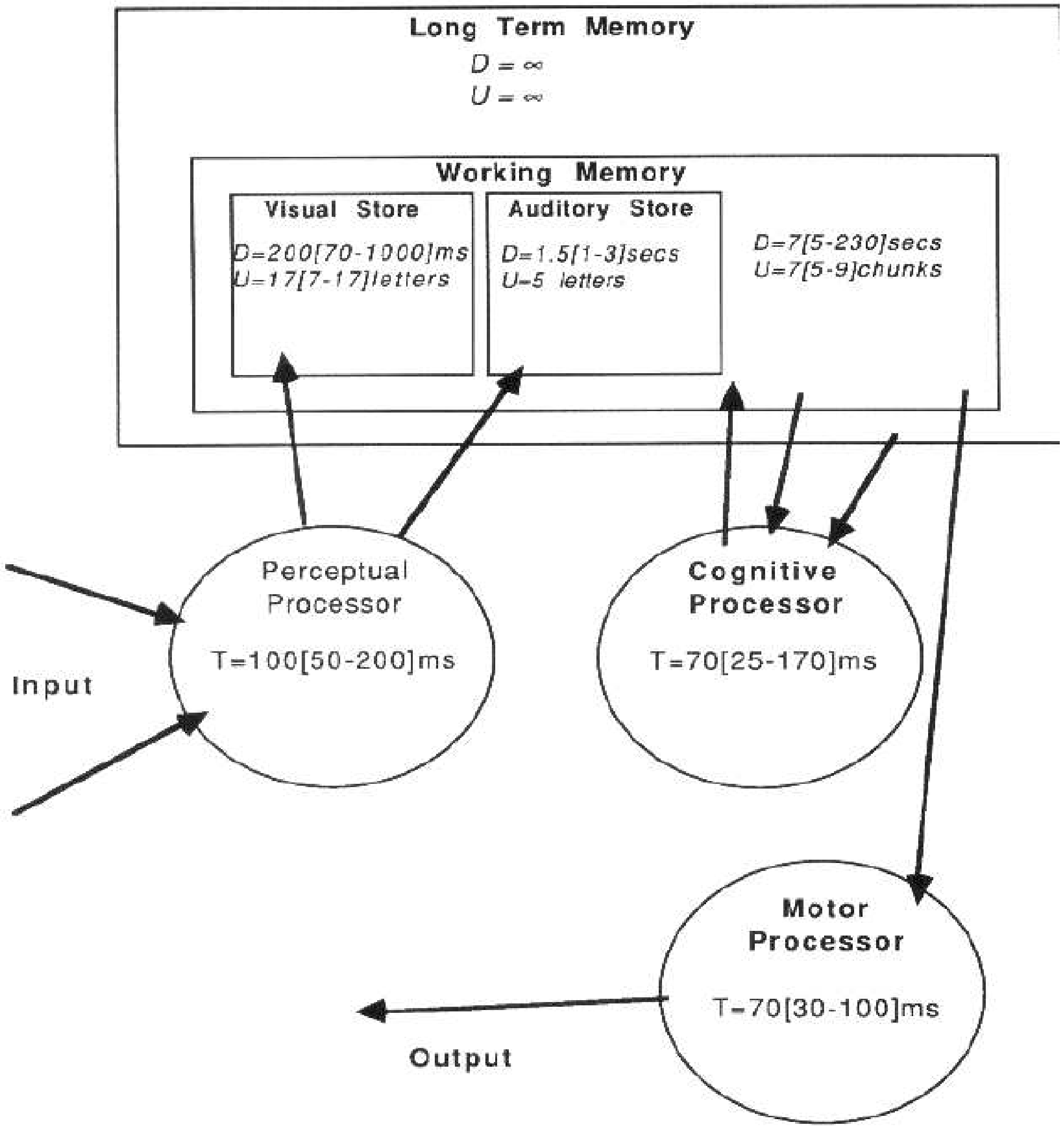
```
(MOVE-GAZE-AND-CURSOR-TO-TARGET
IF
((GOAL DO MENU TASK)
 (STEP MOVE-GAZE-AND-CURSOR-TO-TARGET)
 (WM TARGET-OBJECT IS ?TARGET-OBJECT)
 (WM CURSOR IS ?CURSOR-OBJECT)
 (MOTOR OCULAR PROCESSOR FREE)
 (MOTOR MANUAL MODALITY FREE))
THEN
((DELDB (STEP MOVE-GAZE-AND-CURSOR-TO-TARGET))
 (ADDDB (STEP MAKE RESPONSE))
 (SEND-TO-MOTOR OCULAR MOVE ?TARGET-OBJECT)
 (SEND-TO-MOTOR MANUAL PERFORM PLY MOUSE
      RIGHT ZERO-ORDER-CONTROL ?CURSOR-OBJECT ?TARGET-OBJECT))
```

Example: applied to menu search – results



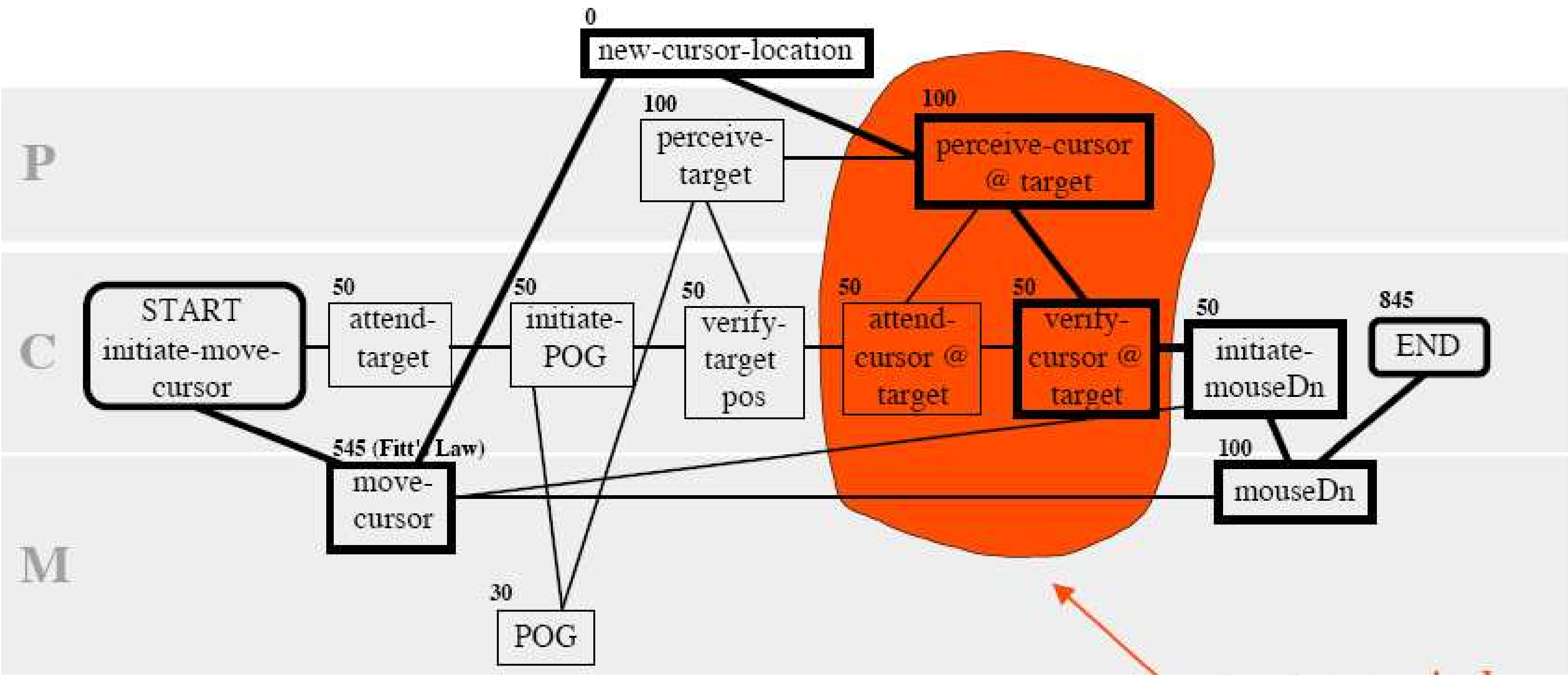
CPM-GOMS

- 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

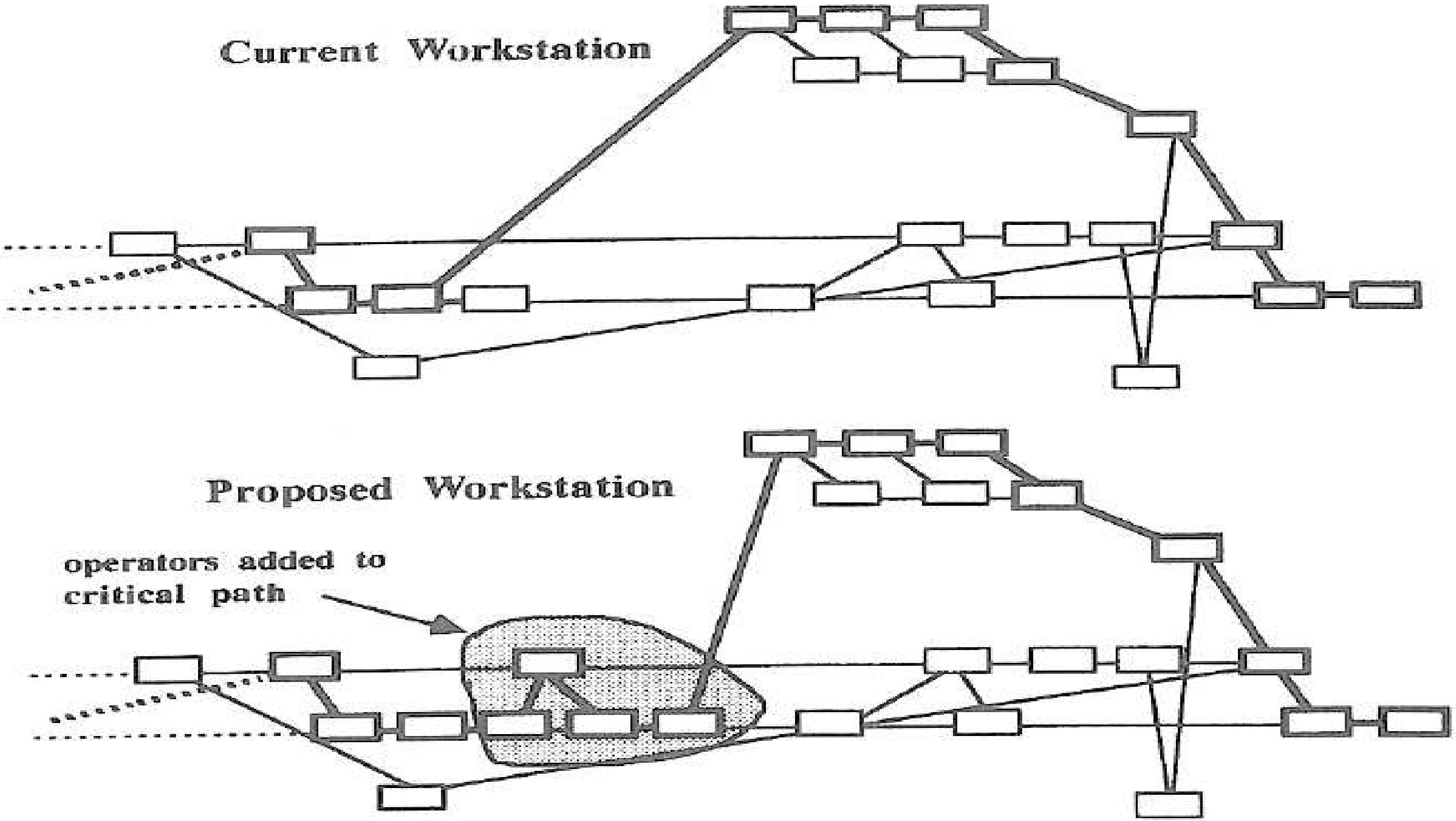
- Expressed as a PERT chart, e.g. “slow move click”



- from *Gray and Boehm-Davis (2001)*

CPM-GOMS

- 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'



CPM-GOMS + APEX

- 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)

```
(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 (mouse-up ?target )                        (waitfor ?m3))
 (step t1  (terminate)                               (waitfor ?m4)) )
```

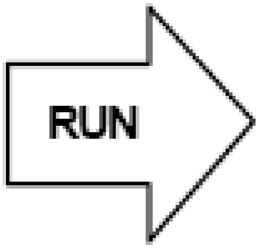
Example: ATM withdrawal

```
(procedure
 (index (do banking))
 (step s1 (initiate session) (priority 300))
 (step s2 (do transaction) (priority 200))
 (step s3 (end session) (priority 100))
 (step t (terminate) (waitfor ?s3  ?s2 ?s1)))

(procedure
 (index (do transaction))
 (step s1 (choose withdraw) (priority 240))
 (step s2 (choose account) (priority 230))
 (step s3 (enter amount) (priority 220))
 (step s4 (retrieve money) (priority 210))
 (step s5 (terminate) (waitfor ?s4 ?s3  ?s2
?s1)))

(procedure
 (index (enter amount))
 (step s1 (enter-number 8-key) (priority 223))
 (step s2 (enter-number 0-key) (priority 222))
 (step s3 (enter-CORRECT) (priority 221))
 (step s4 (terminate)
(waitfor ?s2 ?s1 ?s3)))

(procedure
 (index (enter-number ?number))
 (step s1 (fast-move-click ?number))
 (step s2 (terminate) (waitfor ?s1)))
```



Example: ATM withdrawal

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

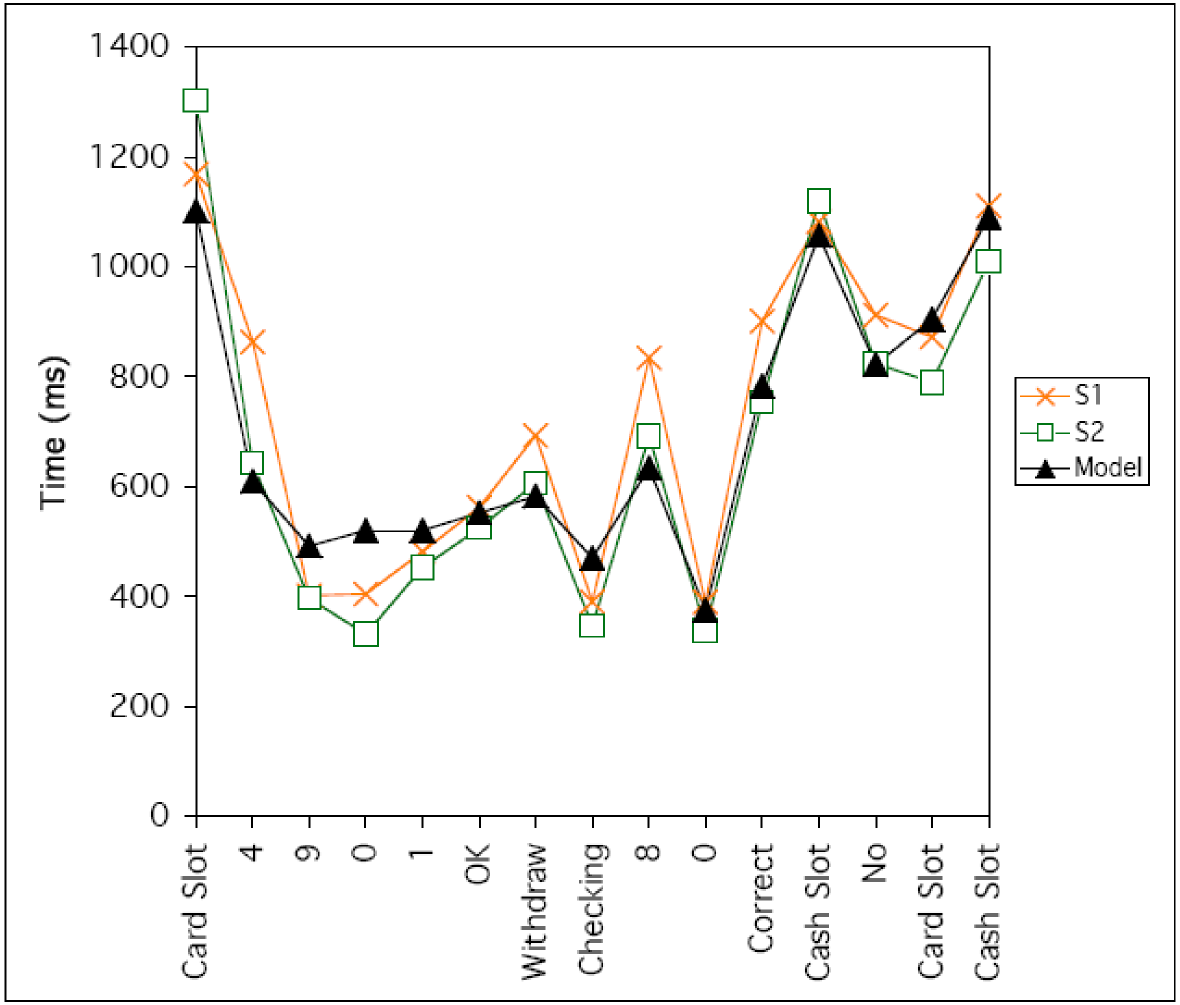
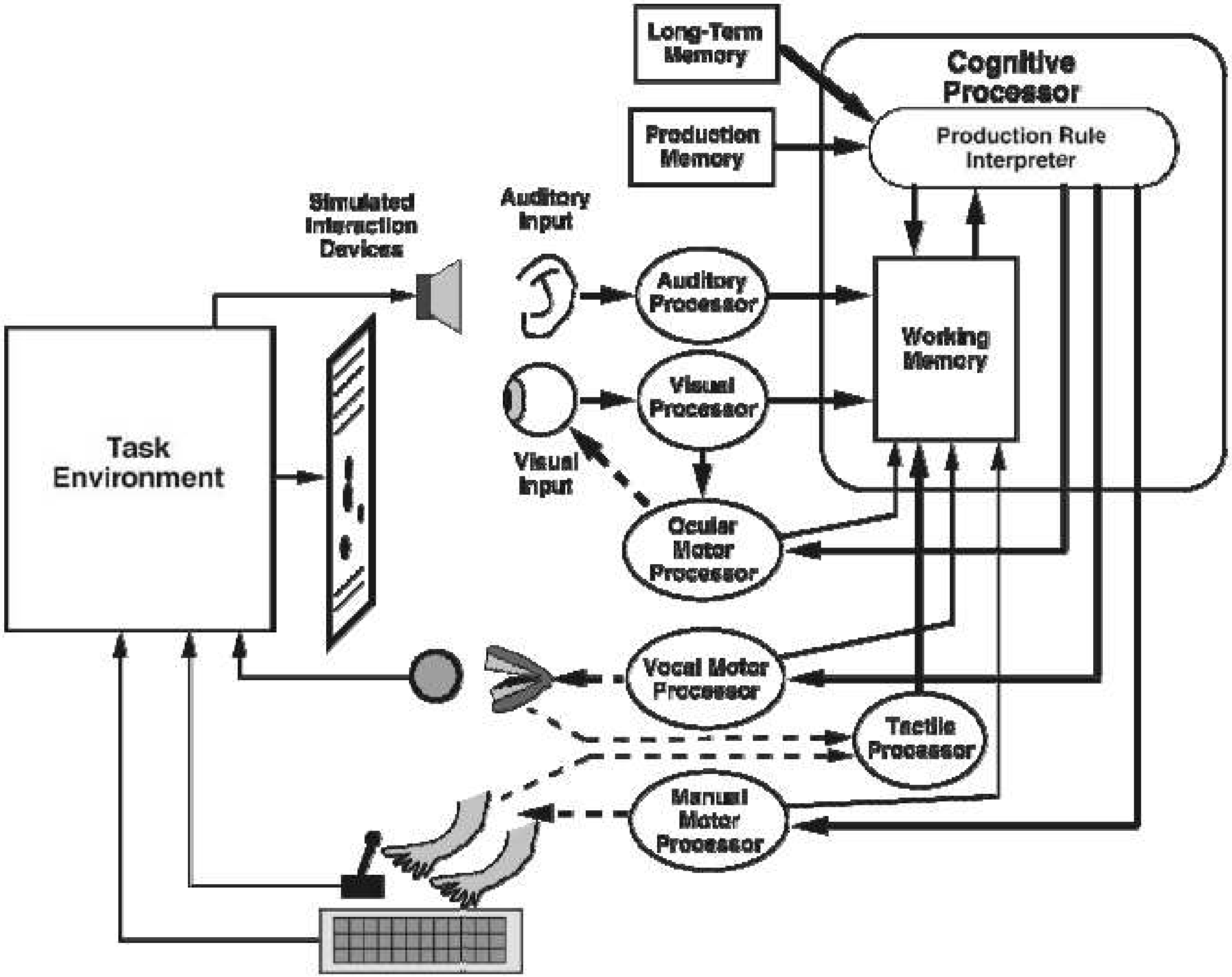


Figure 4: Model predictions and user results

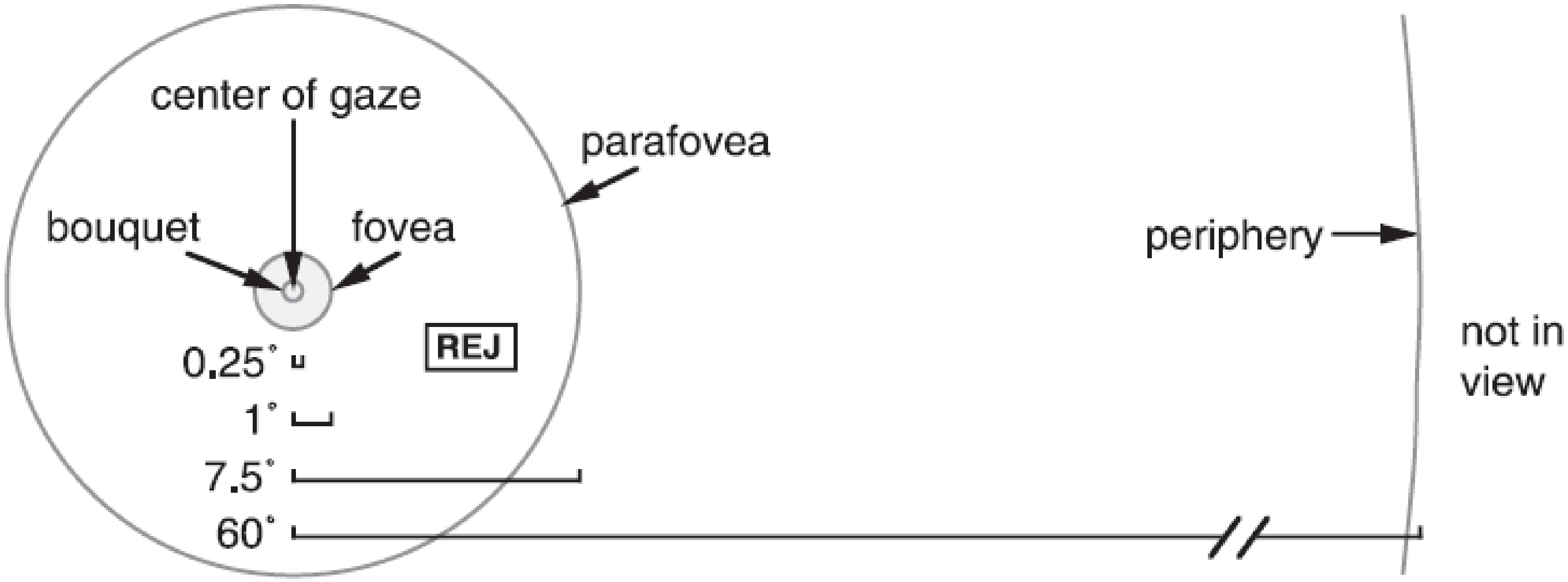
EPIC

- EPIC architecture includes more sensorimotor detail



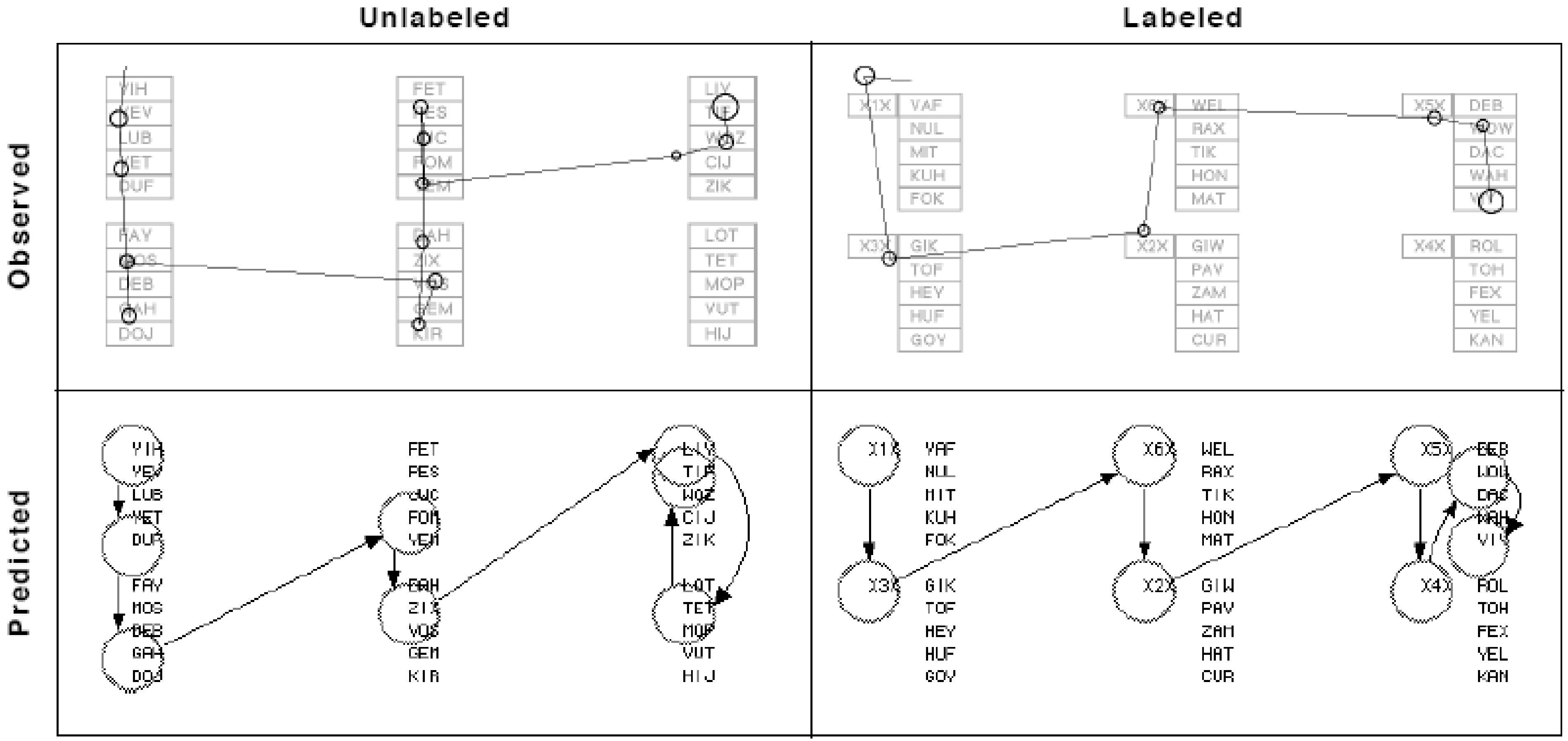
EPIC

- 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)



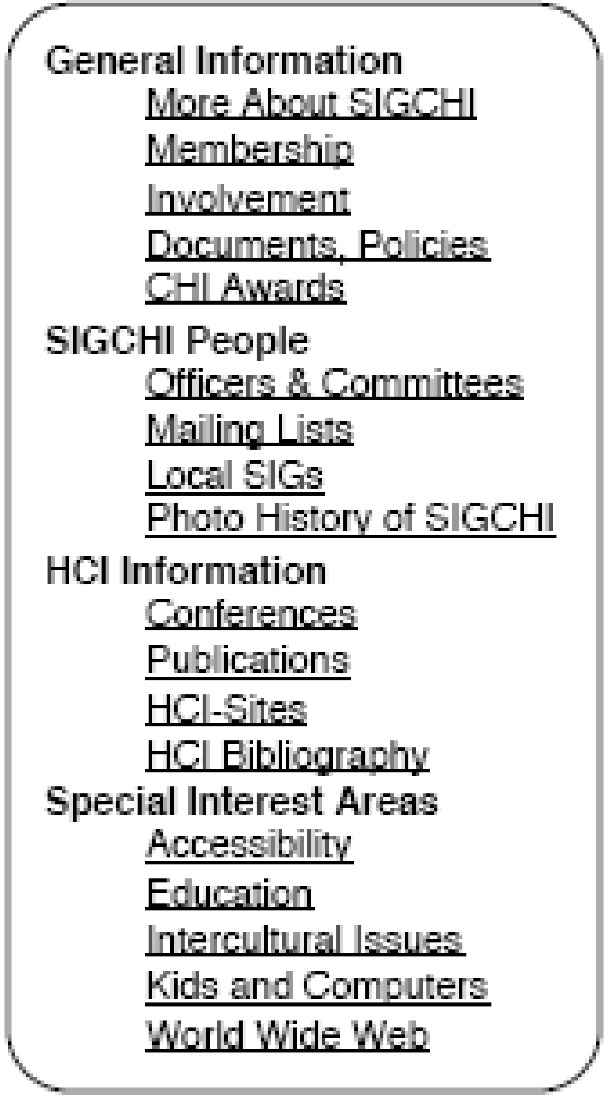
EPIC

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



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.



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"?

References

Young, R.M. and Green, T.R.G (1989) *Programmable user models for evaluation of interface designs.* Proceedings of CHI'89, pp15-19

Card, S.K., Moran, T.P. and Newell, A. (1983) *The Psychology of Human Computer Interaction*. Lawrence Earlbaum Ass., Hillsdale NJ

Kieras, D.E. and Meyer, D.E. (1995) *An Overview of the EPIC Architecture for Cognition and Performance with Application to Human-Computer Interaction*. University of Michigan EPIC Report No. 5 (TR-95/ONR-EPIC-5) ftp://www.eecs.umich.edu/people/kieras/EPIC/TR-EPIC-5.pdf

Gray & Boehm-Davis (2000) Milliseconds matter: an introduction to microstrategies and to their use in describing and predicting interactive behavior. Journal of Experimental Psychology: Applied, 6: 322-335

John, B.E. et al. (2002) Automating CPM-GOMS. Proceedings of CHI 2002.

- See also:

Dix et. al. sections 12.1, 12.2, 12.5