chapter 12

cognitive models

Cognitive models

- goal and task hierarchies
- linguistic
- physical and device

Cognitive models

- They model aspects of user:
  - understanding
  - knowledge
  - intentions
  - processing

- Computational flavour
  - A bit like a program for using the interface

Goal and task hierarchies

- Mental processing as divide-and-conquer
- Example: sales report of HCI textbooks:

produce report
gather data
. find book names
. . do keywords search of names database
. . . … *further sub-goals*
. . sift through names and abstracts by hand
. . . … *further sub-goals*
. search sales database - further sub-goals
layout tables and histograms - further sub-goals
write description - further sub-goals

- Issue – how much detail? (“granularity”)

goals vs. tasks

- goals – intentions

  what you would like to be true
- tasks – actions

  how to achieve it

- Different methods may emphasize one or the others (e.g., “G” in GOMS is for Goals)

Issues for goal hierarchies

- Granularity
  - Where do we start?
    - Goal (cook eggs, or make breakfast, or eat, or live?)
  - Where do we stop? (how detailed to get - go down to individual hand and eye movements?!)
- Model routine learned behaviour, not problem solving
  - The 'unit task' is something the user will (supposedly) know how to do
- More than one way to achieve a goal
  - Some model this more explicitly ("S" of GOMS)
- Error
  - Generally not good predictors of imperfect use of the system (although CCT can be used to examine 'bugs' in use)

Techniques

- Goals, Operators, Methods and Selection (GOMS)

- Cognitive Complexity Theory (CCT)

- Hierarchical Task Analysis (HTA) - Chapter 15

GOMS

Goals
- what the user wants to achieve

Operators
- basic actions user performs

Methods
- decomposition of a goal into subgoals/operators

Selection
- means of choosing between competing methods

GOMS example

```
GOAL: CLOSE-WINDOW
.   [select GOAL: USE-MENU-METHOD
.               .   MOVE-MOUSE-TO-FILE-MENU
.               .   PULL-DOWN-FILE-MENU
.               .   CLICK-OVER-CLOSE-OPTION
                GOAL: USE-CTRL-W-METHOD
.               .   PRESS-CONTROL-W-KEYS]

For a particular user, U1:

    Rule 1: Select USE-MENU-METHOD unless another
            rule applies
    Rule 2: If the application is GAME,
            select CTRL-W-METHOD
```

So here we have one Goal with either of two Methods, one of which requires a sequence of three Operators, the other requires just one Operator; for U1 we have 2 Selection rules

GOMS exercise

Delete a file using Windows Explorer

Three alternative methods: drag-to-trash, delete-key, or right-click

Within the delete-key method we have alternative sub-goals: confirm with keyboard or confirm with mouse

GOMS exercise answer

```
GOAL:    DELETE-FILE
.        LOCATE-FILE
.        MOVE-CURSOR-OVER-FILE
.        [SELECT GOAL:    DRAG-TO-TRASH-METHOD
.                .        HOLD-MOUSE-BUTTON-DOWN
.                .        LOCATE-TRASH-ICON
.                .        MOVE-CURSOR-TO-TRASH-ICON
.                .        VERIFY-TRASH-IS-REVERSE-VIDEO
.                .        RELEASE-MOUSE-BUTTON
.                GOAL:    USE-DELETE-KEY-METHOD
.                .        CLICK-ON-FILE
.                .        PRESS-DELETE-KEY
.                .        LOCATE-CONFIRM-YES
.                .        [SELECT  GOAL:    CONFIRM-YES-KEYBOARD-METHOD
.                .                 .        PRESS-Y-KEY
.                .                 GOAL:    CONFIRM-YES-MOUSE-METHOD
.                .                 .        MOVE-CURSOR-OVER-YES-BUTTON
.                .                 .        CLICK-ON-YES-BUTTON]
.                GOAL:    USE-RIGHT-CLICK-OPTION-METHOD
.                .        RIGHT-CLICK-ON-FILE-AND-HOLD-DOWN
.                .        LOCATE-DELETE-OPTION
.                .        MOVE-CURSOR-OVER-DELETE-OPTION
.                .        RELEASE-MOUSE-BUTTON
.                .        LOCATE-CONFIRM-YES
.                .        …
```

Cognitive Complexity Theory

- Two parallel descriptions:
  - User: User production rules (uses LISP-like syntax [prefix notation])
  - System: Device generalised transition networks
- Production rules are of the form:
  - if condition then action
- Transition networks covered under dialogue models (subsequent lecture)

Some production rules for editing with vi

```
(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-MOVE-FIRST
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-DOIT
IF (AND  (TEST-GOAL insert space)
         (TEST-CURSOR %LINE %COL) )
THEN (   (DO-KEYSTROKE ‘I’)
         (DO-KEYSTROKE SPACE)
         (DO-KEYSTROKE ESC)
         (DELETE-GOAL insert space) ))
(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) ))
```

Using the rules for editing with vi

- Production rules are in long-term memory
- Model working memory as attribute-value mapping:

  (GOAL perform unit task)

  (TEXT task is insert space)

  (TEXT task is at 5 23)

  (CURSOR 8 7)

- Rules are pattern-matched to working memory,

  e.g., LOOK-TEXT task is at %LINE %COLUMN is true, with LINE = 5 COLUMN = 23.

Rules with matched patterns are 'fired' to update working memory

Active rules:

```
SELECT-INSERT-SPACE
INSERT-SPACE-MOVE-FIRST
INSERT-SPACE-DOIT
INSERT-SPACE-DONE
```

New working memory

```
(GOAL insert space)
(NOTE executing insert space)
(LINE 5) (COLUMN 23)
```

SELECT-INSERT-SPACE
matches current working memory

```
(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 %COLUMN)))
```

Notes on CCT

- Rulebase can get quite complex
- Rules are not executed in order (it's less like a conventional program, more like a knowledge base in AI expert systems)
- Can represent novice versus expert style behaviour (different rules)
- Error behaviour can be represented
- Measures usability
  - depth of goal structure
  - number of rules

Problems with goal hierarchies

- a post hoc technique
  - When designed after the interface has been built, often model the interface dialog very closely
- expert versus novice
  - Tends to assume user knows just what to do
- How cognitive are they?
  - Not much model of user finding and recognizing things on the screen – just about acting

Linguistic notations

- Understanding the user's behaviour and cognitive difficulty based on analysis of language between user and system.
- Similar in emphasis to dialogue models (chapter 16, subsequent lecture)

- We'll look at Backus–Naur Form (BNF)
- Task–Action Grammar (TAG) is another

Backus-Naur Form (BNF)

- Very common notation from computer science for command syntax
- A purely syntactic view of the dialogue
- Terminals
  - lowest level of user behaviour
  - e.g. CLICK-MOUSE, MOVE-MOUSE
- Nonterminals
  - ordering of terminals
  - higher level of abstraction
  - e.g. select-menu, position-mouse

Example of BNF

- Basic syntax:
  - nonterminal ::= expression
- An expression
  - contains terminals and nonterminals
  - combined in sequence (+) or as alternatives (|)

```
draw-line      ::= select-line + choose-points + last-point
select-line    ::= pos-mouse + CLICK-MOUSE
choose-points  ::= choose-one   |   choose-one + choose-points
choose-one     ::= pos-mouse + CLICK-MOUSE
last-point     ::= pos-mouse + DBL-CLICK-MOUSE
pos-mouse      ::= NULL  |  MOVE-MOUSE + pos-mouse
```

BNF exercise answer

Deleting a file with Windows Explorer…

```
delete-file    ::= pos-mouse + select-delete
select-delete  ::= drag-delete | key-delete | button-delete
drag-delete    ::= HOLD-MOUSE-DOWN + pos-mouse + RELEASE-MOUSE
key-delete     ::= CLICK-MOUSE + PRESS-DELETE + confirm-yes
button-delete  ::= HOLD-MOUSE-DOWN-RIGHT + pos-mouse + RELEASE-MOUSE
                       + confirm-yes
confirm-yes    ::= PRESS-Y | pos-mouse + CLICK-MOUSE
pos-mouse      ::= NULL | MOVE-MOUSE + pos-mouse
```

Compare to the earlier GOMS specification

Measurements with BNF

- Number of rules
  - not so good because we can write more or less complex rules
- Number of + and | operators
- Limitation
  - no reflection of user's perception

Physical and device models

- We’ll look at the Keystroke Level Model (KLM)
- Another is Buxton's 3-state model

- Based on empirical knowledge of human motor system
- User's task: acquisition then execution
  - these only address execution
- Complementary with goal hierarchies

Keystroke Level Model (KLM)

- lowest level of (original) GOMS
- seven execution phase operators
  - Physical motor: K - keystroking
    B – mouse button
    P - pointing
    H - homing
    D - drawing
  - Mental M - mental preparation
  - System R - response
- times are empirically determined.

$$T_{execute} = T_K + T_B + T_P + T_H + T_D + T_M + T_R$$

KLM times (Card, Moran & Newell)

K	Press key	
	• Good typist (90 wpm)	0.12
	• Poor typist (40 wpm)	0.28
	• Non-typist	1.20
B	Mouse button press	
	• Down or up	0.10
	• Click	0.20
P	Point with mouse	
	• Fitts' law	$0.1 \log_2(D/S + 0.5)$
	• Average movement	1.10
H	Hands to/from keyboard	0.40
D	Drawing	Domain dependent
M	Mentally prepare	1.35
R	Response from system	Measure

KLM example

```
GOAL: ICONISE-WINDOW
[select
   GOAL: USE-CLOSE-METHOD
   .      MOVE-MOUSE-TO- FILE-MENU
   .      PULL-DOWN-FILE-MENU
   .      CLICK-OVER-CLOSE-OPTION
   GOAL: USE-CTRL-W-METHOD
          PRESS-CONTROL-W-KEY]
```

- compare alternatives:
  - USE-CTRL-W-METHOD vs.
  - USE-CLOSE-METHOD
- assume hand starts on mouse

USE-CTRL-W-METHOD		**USE-CLOSE-METHOD**	
H[to kbd]	0.40	P[to menu]	1.1
M	1.35	B[LEFT down]	0.1
K[ctrlW key]	0.28	M	1.35
		P[to option]	1.1
		B[LEFT up]	0.1
Total	**2.03 s**	**Total**	**3.75 s**

KLM exercise

- Delete a file using drag to trash method

- Delete a file using delete key method

KLM exercise answer

<u>Drag to trash</u>

P[to file]	1.1
B[LEFT down]	0.1
M	1.35
P[to trash]	1.1
B[LEFT up]	0.1
	===
	3.75 s

<u>Delete key</u>

P[to file]	1.1
B[click]	0.2
H[to keyboard]	0.4
M	1.35
K[Delete key]	0.28
M	1.35
H[to mouse]*	0.4
M	1.35
P[to Yes button]	1.1
B[click]	0.2
	===
	7.73 s

* using the mouse for the Yes (confirm)

Assume that the user's hand starts on the mouse. Also assume that the trash icon is visible at the time the user wishes to delete the file.

Rules for Placing Mental (M) Operators

Use Rule 0 to place candidate M's and then cycle through Rules 1 to 4 for each M to see whether it should be deleted

- **Rule 0** Insert M's in front of all K's and B's that are not part of text or numeric argument strings proper (e.g., text or numbers). Place M's in front of all P's that select commands (not arguments).
- **Rule 1** If an operator following an M is fully anticipated in an operator just previous to M, then delete the M.
  - E.g., point with mouse then click PMB -> PB
- **Rule 2** If a string of MK's belongs to a cognitive unit (e.g., the name of a command) then delete all M's but the first.
- **Rule 3** If a K is a redundant terminator (e.g., the terminator of a command immediately following the terminator of its argument) then delete the M in front of it.
  - E.g., terminate argument and then command MKMK -> MKK
- **Rule 4** If a K terminates a constant string (e.g., a command name) then delete the M in front of it; but if the K terminates a variable string (e.g., an argument string) then keep the M in front of it.

Cognitive architectures

- All of these cognitive models make assumptions about the architecture of the human mind
  - E.g., Long-term/Short-term memory (well, this is well supported by research)
- Problem space model
  - Problem solving is a search for a goal using available operators (like GOMS)
  - This theory was important to AI in the 70's
- Interacting Cognitive Subsystems
  - A competing model where the problem-solving emerges through various subsystems

Display-based interaction

- Most cognitive models do not deal well with user observation and perception
- Some techniques have been extended to handle system output
  (e.g., BNF with sensing terminals, Display-TAG)
  but problems persist
- There's a tension in the theory of 'exploratory interaction' versus the planning emphasis of GOMS

Conclusion

- Cognitive Models
  - Are like a program for using the system
  - Allow you to assess and measure the usability of the system without experimenting on humans
    - Which is really nifty as an alternative or complementary approach to usability assessment
  - Provide only a limited model of visual display characteristics