Human Computer Interaction

UNIT-2

Lecture 3:

Model-based Design

Mr. Nachiket Sainis / Reena Saini

B K Birla Institute of Engineering & Technology, Pilani

Lecture 3:
Keystroke Level Model - II

Objective

- In the previous lecture, we learned about the keystroke level model (KLM)
- There, we break down a complex cognitive task into a series of keystroke level (elementary) cognitive operations, called operators
  - Each operator has its own pre-determined execution time (empirically derived)

- Although there are a total of seven original operators, two (D and R) are not used much nowadays

- Any task execution with an interactive system is converted to a list of operators.

- When we add the execution times of the operators in the list, we get an estimate of the task execution time (by a user) with the particular system, thereby getting some idea about the system performance from user's point of view

  - The choice of the task is important and should represent the typical usage scenario

An KLM Example

- Suppose a user is writing some text using a text editor program. At some instant, the user notices a single character error (i.e., a wrong character is typed) in the text. In order to rectify it, the user moves the cursor to the location of the character (using mouse), deletes the character and retypes the correct character. Afterwards, the user returns to the original typing position (by repositioning the cursor using mouse). Calculate the time taken to perform this task (error rectification) following a KLM analysis.

Building KLM for the Task

- To compute task execution time, we first need to build KLM for the task
  - That means, listing of the operator sequence required to perform the task
- Let us try to do that step-by-step

Execution time of Operator (KLM)

Operator	Description	Time (second)
K (The key press operator)	Time to perform K for a good (expert) typist	0.12
	Time to perform K by a poor typist	0.28
	Time to perform K by a non-typist	1.20
B (The mouse- button press/release operator)	Time to press or release a mouse-button	0.10
	Time to perform a mouse click (involving one press followed by one release)	2×0.10 = 0.20
P (The pointing operator)	Time to perform a pointing task with mouse	1.10
H (the homing operator)	Time to move hand from/to keyboard to/from mouse	0.40
M (The mental operator)	Time to mentally prepare for a physical action	1.35

Building KLM for the Task

- Step 1: user brings cursor to the error location
  - To carry out step 1, user moves mouse to the location and ‘clicks’ to place the cursor there
- Operator level task sequence

Description	Operator
Move hand to mouse	H
Point mouse to the location of the erroneous character	P
Place cursor at the pointed location with mouse click	BB

Building KLM for the Task

- Step 2: user deletes the erroneous character
  - Switches to keyboard (from mouse) and presses a key (say “Del” key)
- Operator level task sequence

Description	Operator
Return to keyboard	H
Press the “Del” key	K

Building KLM for the Task

- Step 3: user types the correct character
  - Presses the corresponding character key
- Operator level task sequence

Description	Operator
Press the correct character key	K

Building KLM for the Task

- Step 4: user returns to previous typing place
  - Moves hand to mouse (from keyboard), brings the mouse pointer to the previous point of typing and places the cursor there with mouse click
- Operator level task sequence

Description	Operator
Move hand to mouse	H
Point mouse to the previous point of typing	P
Place cursor at the pointed location with mouse click	BB

Building KLM for the Task

- Total execution time (T) = the sum of all the operator times in the component activities

T = HPBBHKKHPBB = 6.20 seconds

Step	Activities	Operator Sequence	Execution Time (sec)
1	Point at the error	HPBB	0.40+1.10+0.20 = 1.70
2	Delete character	HK	0.40+1.20 = 1.60
3	Insert right character	K	1.20
4	Return to the previous typing point	HPBB	1.70

Something Missing!!

- What about M (mental operator) – where to place them in the list?
- It is usually difficult to identify the correct position of M
  - However, we can use some guidelines and heuristics

General Guidelines

- Place an M whenever a task is initiated
- M should be placed before executing a strategy decision
  - If there is more than one way to proceed, and the decision is not obvious or well practiced, but is important, the user has to stop and think
- M is required for retrieving a chunk from memory
  - A chunk is a familiar unit such as a file name, command name or abbreviation
  - Example - the user wants to list the contents of directory *foo;* it needs to retrieve two chunks - *dir* (command name) and *foo* (file name), each of which takes an M

General Guidelines

- Other situations where M is required
  - Trying to locate something on the screen (e.g., looking for an image, a word)
  - Verifying the result of an action (e.g., checking if the cursor appears after clicking at a location in a text editor)
- Consistency – be sure to place M in alternative designs following a consistent policy
  - Number of Ms – total number of M is more important than their exact position
  - Explore different ways of placing M and count total M in each possibility.

General Guidelines

- Apples & oranges – don't use same policy to place M in two different *contexts* of interaction
  - Example: don't place M using the same policy while comparing between menu driven word processing (MS Word) vs command driven word processing (Latex).
- Yellow pad heuristics
  - If the alternative designs raises an *apples & oranges* situation then consider removing the mental activities from action sequence and assume that the user has the results of such activities easily available, as if they were written on a yellow pad in front of them

M Placement Heuristics

Rule 0: initial insertion of candidate Ms

- Insert Ms in front of all keystrokes (K).
- Insert Ms in front of all acts of pointing (P) that select commands.
- Do not insert Ms before any P that points to an argument.
- Mouse-operated widgets (like buttons, check boxes, radio buttons, and links) are considered commands.
- Text entry is considered as argument.

M Placement Heuristics

Rule 1: deletion of anticipated Ms

If an operator following an M is fully anticipated in an operator immediately preceding that M, then delete the M.

Example -if user clicks the mouse with the intention of typing at the location, then delete the M inserted as a consequence of rule 0.

–So BBMK becomes BBK

M Placement Heuristics

Rule 2: deletion of Ms within cognitive units

If a string of MKs belongs to a cognitive unit then delete all Ms except the first.

A cognitive unit refers to a chunk of cognitive activities which is predetermined.

Example -if a user is typing “100”, MKMKMK becomes MKKK (since the user decided to type 100 before starts typing, thus typing 100 constitutes a cognitive unit).

M Placement Heuristics

Rule 3: deletion of Ms before consecutive terminators

If a K is a redundant delimiter at the end of a cognitive unit, such as the delimiter of a command immediately following the delimiter of its argument, then delete the M in front of it.

Example: when typing code in Java, we end most lines with a semi-colon, followed by a carriage return. The semi-colon is a terminator, and the carriage return is a redundant terminator, since both serve to end the line of code.

M Placement Heuristics

Rule 4: deletion of Ms that are terminators of commands

If a K is a delimiter that follows a constant string, a command name (like “print”), or something that is the same every time you use it, then delete the M in front of it.

If a K terminates a variable string (e.g., the name of the file to be printed, which is different each time) then leave it.

M Placement Heuristics

Rule 5: deletion of overlapped Ms

Do not count any portion of an M that overlaps a R —a delay, with the user waiting for a response from the computer.

Example: user is waiting for some web page to load (R) while thinking about typing the search string in the web page (M). Then M should not come before typing since it is overlapping with R

KLM Limitations

- Although KLM provides an easy-to-understand-and-apply predictive tool for interactive system design, it has few significant constraints and limitations
  - It can model only “expert” user behavior
  - User errors can not be modeled
  - Analysis should be done for “representative” tasks ; otherwise, the prediction will not be of much use in design. Finding “representative” tasks is not easy